Make Your Own Cloud Security Monitoring Solution
About The Presenter

John Ventura
Security Researcher
Former ISS X-Force Member
Currently Staff Engineer at Datadog*

*These opinions are mine and not my employer’s

@JohnAVentura
Alerting in the Cloud

- Build an alerting system for GCP or AWS
- Associated dangers
- Overcoming these dangers
Why Alerting?

Do you have policies?
Do people make mistakes?
Do you get attacked?
Build An Alerting System

You already have the components!
CIDF defines four categories of components:

- Event generators
- Analyzers
- Databases
- Response units
Actual Response Unit
For this talk, we will focus on the CIDF components provided by AWS and GCP.
Public cloud environments provide some CIDF components as services:

- Easy to configure
- Can feed into 3rd party tools
- Accessible by API Calls
  (EVERYTHING is an API CALL)
Native Event Generators

AWS (CloudTrail)

• Easily configurable
• Logs ALL API calls to S3 or Lambda
• Easily consumable by third party tools
Native Event Generators

AWS (CloudWatch)

- Easily configurable
- Logs filtered API calls to anywhere
- Easily consumable by third party tools
Native Event Generators

Stackdriver (GCP)

- GCP log management platform
- Collects GCP event data
- Includes features that facilitate data management
Native Analyzers

AWS and GCP both provide native analyzers

• Accessible via the GUI
• Accessible via APIs
• Data presentations may vary considerably based on logged calls
GCP’s Native Analyzer

Create a filter!
GCP’s Native Response Unit

Create a policy!
AWS’ Native Analyzer

CloudWatch Management Console

Rules > AlertingCW

Summary

ARN: arn:aws:events:us-east-1: rule/AlertingCW

Event pattern:

```
{
  "detail": {
    "eventName": [
      "AuthorizeSecurityGroupIngress",
      "CreateInstance"
    ],
    "eventSource": [
      "ec2.amazonaws.com"
    ],
    "detail-type": [
      "AWS API Call Tracking"
    ],
    "source": [
      "aws.ec2"
    ]
  }
}
```

Status: Enabled

Description

Monitoring: Show metrics for the rule

Targets

Filter: 

<table>
<thead>
<tr>
<th>Type</th>
<th>Resource name</th>
<th>Input</th>
<th>Role</th>
<th>Additional parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lambda function</td>
<td>AlertingFunction</td>
<td>Matched event</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
AWS Native Response Unit

AWS offers native support for:

• SMS through SNS
• E-mail through Simple Email Service
• Much more...
Simple AWS Alerting

CloudWatch → SNS → SMS → Admins
AWS CIDF Summary

- Configurable
- Consumable
- Spread across multiple services
- Intended for programmatic access
- Log formats have inconsistent schema
GCP CIDF Summary

- Configurable
- Consolidated in single service
- Heavy focus on GUI
- Log formats have inconsistent schema
Incompleteness Theorem

BEWARE!
gcloud logging [metrics | sinks] list

$ gcloud logging metrics list
NAME	DESCRIPTION
ExampleMetric	Example Metric
deleteinstancemetric
deletewithlogname

FILTER
resource.type="gce_instance"
logName="projects/first-vision-211303/logs/cloudaudit.googleapis.com%2Factivity"
protoPayload.methodName="beta.compute.instances.insert"
resource.type="gce_instance"
protoPayload.methodName="v1.compute.instances.delete"
resource.type="gce_instance"
logName="projects/first-vision-211303/logs/cloudaudit.googleapis.com%2Factivity"


gcloud logging metrics update [metric_name] --log-filter = ”SOMETHING”

$ gcloud beta logging metrics update dev-insertinstance-metric --log-filter="resource.type=gce_instance AND logName=projects/cloudland/logs/cloudaudit.googleapis.com%2Factivity"

apis.com%2Factivity AND protoPayload.methodName=beta.compute.instances.delete"
Updated [dev-insertinstance-metric].

Information for Red Teams!
Information for Red Teams!

```
# gcloud pubsub subscriptions pull --auto-ack alertSub
```
Information for Red Teams!

- `aws events list-rules` # for CloudWatch rules
- `aws events [disable-rule | enable-rule]` —name [rule name]
- `aws cloudtrail describe-trails`
Information for Red Teams!
Native Alerting Limitations

- Transparent to attackers
- Easily clobbered by admins (including you)
- Metrics/Exports/Filters can be imprecise
What Can We Do?
Build your own Alerting!

Advantages

• Less transparent to attackers
• Allows for more complicated filtering
• Enabled third-party technology
• Storage and retention...
Case Study

In this scenario

• AWS is our primary cloud

• GCP is our secondary cloud

• Shipping data out of GCP to other CIDF components

• Custom shipper “GoodCoP”
Stackdriver API

- Continuous Polling
- Lose some convenience
- Protobuf support*

*https://github.com/googleapis/googleapis/blob/master/google/cloud/audit/audit_log.proto
from google.cloud.logging import Client, ASCENDING, DESCENDING

client = Client(project = projectName)

while true:
    timeFilter = GetFilter(LastScanTime)
    entries = False
    while not entries:
        entries = client.list_entries(order_by=DESCENDING, filter_ = timeFilter)
    for entry in entries:
        DoSomething(entry.payload)
        UpdateScanTime(entry.timestamp)
GoodCoP Configuration

```yaml
projects:
  - cool-project
  - scorpio
  - london

calls:
  - ALL

alertcycle: 61

sinks:
  logfile:
    filename: /var/log/goodcop.log
  s3:
    bucket: gcp-bucket
    key: stackdriver-raw
```
Flip the Script

What if AWS alerting data flows into GCP?
Considerations for AWS

- **CloudTrail** - Dump all your API event to S3 or Lambda
- **CloudWatch** - Dump filtered API events to Lambda, Kinesis, SQS, SNS, or elsewhere

Consider separate accounts!
Adding Third Parties

Several third party CIDF components are available.
Open Source Searching and Alerting

- Elasticsearch
- Elastalert
- Streamalert
Third Party Response Units

- Slack — https://api.slack.com/
- PagerDuty — https://v2.developer.pagerduty.com/
- SMS — Several available
- SMTP Email — Several available
- Smart bulbs — https://www.developers.meethue.com/philips-hue-api
Simple AWS Alerting

CloudWatch → Kinesis → Elasticsearch → ElastAlert

Admins
Datadog as Datastore
Datadog as Analyzer

1. Define the search query
   - Project: datadog-demodog
   - @methodName: beta.compute.instances.insert

2. Set alert conditions
   - Trigger when the metric is above the threshold during the last 5 minutes
   - Alert threshold: 1
   - Warning threshold: Warning threshold (optional)

3. Say what's happening
   - Example Monitor Name: Number of errors is very high for...
   - Someone just created an instance! STOP THEM! @john.ventura@example.com @slack-alerting-channel @pagerduty-Wake-Me-UP

4. Notify your team
   - @john.ventura@example.com
Datadog Response Units

1. Define the search query
2. Set alert conditions
3. Say what's happening
4. Notify your team
CIDF components are out there
These systems can be fragile
Good luck!
Thank you

- Twitter: @JohnAVentura
- Github: https://github.com/johnaventura/