Protecting Cloud Workloads

Protecting Workloads in Google Kubernetes with OSSEC and Google Cloud Armor
Who we are & what we do

• News and information media company
• 140 million unique visitors every month
• 120 markets in the US
• Global reach: Canada, Europe, Brazil, India, Australia
• Augmented Reality (AR) and drones
• Award winning content
• Making a difference in communities
• Protecting our content and delivery
How we deliver content

Surprise, we use the cloud!

• Content delivery requires scalability
• Fastly is our CDN
• Origin is cloud agnostic (move workloads around)
• We use containers & Kubernetes
• We use Google Kubernetes Engine (GKE)
• Why Google
Detect and block bad things

Joe’s challenge: find a solution to monitor & protect containers
Objective

• Detect malicious requests to GKE containers
• Block malicious requests (active-response, CloudArmor)
• Monitor GKE pods for changes (pod restarts, added, deleted)
Everyone has a plan...

- Do not install agent on every container
- Do not monitor containers over ssh
- Unable to install software on Kubernetes worker nodes (managed by Google)
- OSSEC agent wasn’t an option
- OSSEC agentless wasn’t an option
- Leaves us with monitoring Google Cloud Platform (GCP) logs
- Can’t make it easy for Joe
How Joe solved it

You can do this in 2 hours (pretty sure)
Google Kubernetes Engine, Cloud Armor, Stackdriver, Logstash, and OSSEC

**Google Kubernetes Engine** is a managed, production-ready environment for deploying containerized applications.

**CloudArmor**
- Infrastructure DDoS defense
- IP deny list/allow list
- (WAF) Rich language for custom defense **ALPHA**
Getting Logs to OSSEC

GKE Worker Instance

Pod

Nginx Container

/var/log/containers/...

Daemonset

Fluentd Container

OSSec Server

Pub/Sub Topic

Subscription

Logstash Service

Export GKE

Export GCE

Export HTTP LB

/\var/log/gcp.log

OSSec Service

HTTP Load Balancing

GCE Audit Logs

Protecting Workloads in Google Kubernetes with OSSEC and Google Cloud Armor
Getting traffic to our webservice

• HTTP(S) Load Balancer: Layer 7
  – HTTP aware – logging of client ip, url, http status
  – Server sees ip of load balancer, not client ip
  – Original client ip sent to Server in X-Forwarded-For header
  – Automatic health checking
  – Integrates with CloudArmor
    • DDoS protection
    • IP Access control via security policies
  – Logs to Stackdriver
  – Can monitor HTTP requests in HTTP LB or GKE Logs
OSSEC Decoders for Stackdriver

- GCP logs
  - Start with {"insertId":
  - Or start with {"httpRequest":
- GKE logs
  - Container logs in “textPayload” field
- HTTP LB logs
  - Extract remotelp (srcip),requestUrl (url),status (id)
  - Set type to “web-log” – activates built-in web rules
Custom OSSEC Rules

• GCP
  – Ignore Syslog too long messages

• GKE
  – Webserver GET Request
  – Ignore Health Checks

• HTTP LB
  – Custom attack rules
OSSEC Active-Response

• Modified ip-customblock.sh
  – Keep track of blocked ips
  – Call custom script to update CloudArmor if changed

• Custom script update_cloudarmor.pl
  – Retrieve CloudArmor security policy
  – Compare with blocked ips
  – Add rules for newly blocked ips
  – Remove rules for expired ips
Protecting Workloads in Google Kubernetes with OSSEC and Google Cloud Armor

Active-Response

- `/var/log/gcp.log`
- HTTP LB message
  - "httpRequest"...
  - Extract srcip, url, id

- OSSEC decoder
  - "gcp-httplb"
  - "httpRequest"...

- OSSEC rule 100500
  - `<decoded_as>gcp/decoded_as>`
  - `<description>Google Cloud Platform Logs</description>`

- OSSEC rule 100601
  - `<if_sid>100500</if_sid>`
  - `<match>passwd</match>`
  - `<description>Attempt to Download passwd file</description>`

- Active Response
  - `<rules_id>100601</rules_id>`
  - `<command>ip-customblock2</command>`

- ip-customblock2.sh
  - Add file for ip to `/ipblock`
  - Call `update_cloudarmor.pl`

- Update_cloudarmor.pl
  - Export Security Policy From Google Cloud
  - Compare list of ips in `/ipblock` to security policy rules
  - Add/Delete Security Policy Rules
Monitoring for Pod Changes

- **OSSEC Process Monitoring**
  - Run `kubectl` command to retrieve pod status every 2 min
  - Compare with previous output
  - Alert if changed
Putting it all together

Working example
Active-Response Demo
Pod Monitoring Example

** Alert 1550181923.8879914: mail - gcp,

2019 Feb 14 17:05:23 ossec-test->/bin/kubectl --namespace cloud-armor-demo get pods | cut -c 1-60

Rule: 100700 (level 10) -> 'Kubernetes Pods Changed in namespace cloud-armor-demo'

ossec: output: '/bin/kubectl --namespace cloud-armor-demo get pods | cut -c 1-60':

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>my-deployment-65d6c6fd75-cx6vn</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
</tr>
<tr>
<td>my-deployment-65d6c6fd75-prcg2</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
</tr>
</tbody>
</table>

Previous output:

...
Code Examples / Rules

<decoder name="gcp">
  <prematch>^{"insertId|^{"httpRequest":</prematch>
</decoder>

<decoder name="gcp-gke">
  <parent>gcp</parent>
  <prematch>"textPayload":</prematch>
</decoder>
<decoder name="gcp-httplb">
   <parent>gcp</parent>
   <type>web-log</type>
   <prematch>^"httpRequest":</prematch>
   <regex>"remoteIp":"(\S*)",\.*,"requestUrl":"(\S*)"\.*"status":(\d+)</regex>
   <order>srcip,url,id</order>
</decoder>
<rule id="100601" level="10">

<if_sid>100500</if_sid>
<match>passwd</match>
<group>web</group>
<description>GCP HTTP Load Balancer request for passwd file</description>
</rule>
Lessons Learned
Lessons learned

• We were able to detect malicious traffic to GKE containers
  – You’ll want to expand on the ruleset

• We could block malicious requests with active-response & CloudArmor
  – Not perfect, slight delay to apply block

• We can monitor GKE pods for changes
  – Restarts, pods added, deleted
  – Can use same method to monitor other resource types

• With a few tweaks, we can monitor other GCP activity
  – Start, stop, create, delete GCE instances, etc.
Takeaway

You can do this
Takeaway

• Monitor and protect your cloud workloads
  • Use your existing OSSEC implementation to monitor your containers
  • Cloud agnostic container monitoring (GCP, AWS, private cloud, etc)
  • OSSEC Active-Response
  • Google Cloud Armor

• What we do
  • We use containers and managed Kubernetes
  • We saved $$$ by using GKE
  • Invest in people. We have smart people using the latest technologies
THANK YOU.

CONTACT

Ben Auch
@benjaminauch
ben@usatoday.com

Joe Miller
jmmiller@gannett.com

github.com/GannettDigital/ossec-gcp
Are you a builder, a breaker, or defender?

WE ARE HIRING

usatodaynetworkcareers.com