Managing Multi-cloud OSSEC Deployments

Andrew Hay, CTO, LEO Cyber Security
+1.415.940.9660
ah@leocybersecurity.com
https://leocybersecurity.com
https://twitter.com/andrewsmhay
About Me

• Co-Founder and Chief Technology Officer (CTO) @ **LEO Cyber Security**

• Former:
  • CISO @ **DataGravity** (now HyTrust)
  • Director of Research @ **OpenDNS** (now Cisco)
  • Chief Evangelist & Director of Research @ **CloudPassage**
  • Senior Security (Industry) Analyst @ **451 Research**
  • Information Security Officer in higher education and financial services
  • Engineering manager @ **Q1 Labs** (now IBM)

• Blogger, author, and rugby coach
OK, maybe you’ve heard of this one...
Introduction

• This session will provide tips and tricks for running OSSEC across
  • Amazon Web Services (AWS),
  • Microsoft Azure, and
  • Google Compute Engine (GCE) instances.
## AWS vs Azure vs Google Comparison

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>AWS</th>
<th>Microsoft Azure</th>
<th>Google Cloud Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Servers</td>
<td>Instances</td>
<td>VMs</td>
<td>VM Instances</td>
</tr>
<tr>
<td>Platform-as-a-Service</td>
<td>Elastic Beanstalk</td>
<td>Cloud Services</td>
<td>App Engine</td>
</tr>
<tr>
<td>Serverless Computing</td>
<td>Lambda</td>
<td>Azure Functions</td>
<td>Cloud Functions</td>
</tr>
<tr>
<td>Docker Management</td>
<td>ECS</td>
<td>Container Service</td>
<td>Container Engine</td>
</tr>
<tr>
<td>Kubernetes Management</td>
<td>EKS</td>
<td>Kubernetes Service</td>
<td>Kubernetes Engine</td>
</tr>
<tr>
<td>Object Storage</td>
<td>S3</td>
<td>Block Blob</td>
<td>Cloud Storage</td>
</tr>
<tr>
<td>Archive Storage</td>
<td>Glacier</td>
<td>Archive Storage</td>
<td>Coldline</td>
</tr>
<tr>
<td>File Storage</td>
<td>EFS</td>
<td>Azure Files</td>
<td>ZFS / Avere</td>
</tr>
<tr>
<td>Global Content Delivery</td>
<td>CloudFront</td>
<td>Delivery Network</td>
<td>Cloud CDN</td>
</tr>
<tr>
<td>Managed Data Warehouse</td>
<td>Redshift</td>
<td>SQL Warehouse</td>
<td>Big Query</td>
</tr>
</tbody>
</table>

Source: [https://www.cloudhealthtech.com/blog/aws-vs-azure-vs-google](https://www.cloudhealthtech.com/blog/aws-vs-azure-vs-google)
• The traditional server-agent model can be followed to monitor the servers with a lightweight agent installed on them.

• All agents report to the central OSSEC manager (Server) to analyze the events and to generate event notifications as appropriate.
Architecture Challenges

• This model works quite well for small and large, fixed setups. However, there are trade-offs:
  • Additional server management overhead,
  • Adding or removing agents overhead, and
  • Limited high availability of OSSEC server.
What Can We Monitor?

- What can we monitor in AWS?
  - AWS CloudTrail
  - AWS Config
  - Amazon VPC
  - Amazon GuardDuty
  - Amazon Macie
  - AWS Key Management Service
  - Amazon Inspector
  - AWS Trusted Advisor
  - More...
What Can We Monitor?

• What can we monitor in Azure?
  • Azure Log Analytics
  • Azure Active Directory Graph
  • Azure Storage
  • More...
What Can We Monitor?

- What can we monitor in Google via Stackdriver Logging?

- All log entries in Stackdriver Logging are represented using a single data type `LogEntry` which defines certain common data for all log entries as well as carries individual payloads.
What Can We Monitor?

• Linux Instances
  • Syslog
  • Apache
  • Cassandra
  • Chef
  • gitlab
  • Jenkins
  • Jetty
  • Joomla
  • Lots more...

• Windows Instances
  • Logging agent messages
  • Windows Event Log
Enter Orchestration…
Chef

- Pick up the OSSEC cookbook at the Chef Supermarket.

- Note, for Wazuh use https://github.com/wazuh/wazuh-chef.

source: https://supermarket.chef.io/cookbooks/ossec#readme
Chef Attributes

node['ossec']['dir']
Installation directory for OSSEC.

node['ossec']['server_role']
When using server/agent setup, this role is used to search for the OSSEC server.

node['ossec']['server_env']
When using server/agent setup, this value will scope the role search to the specified environment.

node['ossec']['agent_server_ip']
The IP of the OSSEC server.

node['ossec']['data_bag']['encrypted']
Boolean value which indicates whether the OSSEC data bag is encrypted.

node['ossec']['data_bag']['name']
The name of the data bag to use.

node['ossec']['data_bag']['ssh']
The name of the data bag item which contains the OSSEC keys.
Chef – ossec.conf

• **ossec.conf** makes little use of XML attributes so you can generally construct nested hashes in the usual fashion.

• Where an attribute is required, you can do it like this:

```perl
default['ossec']['conf']['all']['syscheck']['directories'] = [  { '@check_all' => true, 'content!' => '/bin,/sbin' },  '/etc,/usr/bin,/usr/sbin'
]
```

• Produces:

```xml
<syscheck>
  <directories check_all="yes">/bin,/sbin</directories>
  <directories>/etc,/usr/bin,/usr/sbin</directories>
</syscheck>
```
Chef – agent.conf

• OSSEC servers can also distribute configuration to agents through the centrally managed XM file called `agent.conf`.

• Unlike conventional XML files, `agent.conf` has multiple root nodes so `node['ossec']['agent_conf']` must be treated as an array like so.

```python
default['ossec']['agent_conf'] = [
    {
        'syscheck' => { 'frequency' => 4321 },
        'rootcheck' => { 'disabled' => true }
    },
    {
        '@os' => 'Windows',
        'content!' => {
            'syscheck' => { 'frequency' => 1234 }
        }
    }
]```
Chef – agent.conf

• This produces:

```xml
<agent_config>
  <syscheck>
    <frequency>4321</frequency>
  </syscheck>
  <rootcheck>
    <disabled>yes</disabled>
  </rootcheck>
</agent_config>

<agent_config os="Windows">
  <syscheck>
    <frequency>1234</frequency>
  </syscheck>
</agent_config>
```
Puppet

• The Wazuh folks have spent quite a bit of time building puppet modules for OSSEC.

• Get them at the PuppetForge
  • Modify for your own use.
Puppet *(cont...)*

- Sample puppet manifest
  *(wazuh-manager.pp)*

```puppet
node "server.yourhost.com" {
  class { 'wazuh::server':
    smtp_server => 'localhost',
    ossec_emailto => ['user@mycompany.com'],
  }

  wazuh::command { 'firewallblock':
    command_name => 'firewall-drop',
    command_executable => 'firewall-drop.sh',
    command_expect => 'srcip'
  }

  wazuh::activeresponse { 'blockWebattack':
    command_name => 'firewall-drop',
    ar_level => 9,
    ar_agent_id => 123,
    ar_rules_id => [31153,31151],
    ar_repeated_offenders => '30,60,120'
  }

  wazuh::addlog { 'monitorLogFile':
    logfile => '/var/log/secure',
    logtype => 'syslog'
  }
}
```
Puppet (cont...)

• Download and install the Wazuh module from Puppet Forge:
  
  # puppet module install wazuh-wazuh --version 3.8.2-1

• Install agent via Puppet
  
  # /opt/puppetlabs/bin/puppet agent -t

• More information:
  
  • https://documentation.wazuh.com/current/deploying-with-puppet/index.html
Ansible

• Perhaps the best Ansible playbook I’ve found is the one by *dj-wasabi*.

source: https://galaxy.ansible.com/dj-wasabi
Ansible OSSEC Agent

• OSSEC Agent Role Variables

  ossec_server_ip: This is the ip address of the server running the ossec-server.
  ossec_server_fqdn: This is the fqdn of the server running the ossec-server.
  ossec_server_name: This is the hostname of the server running the ossec-server used for delegate with ansible.
  ossecManaged_server: When set to false, tasks that delegate to ossec server will be skipped.
  ossec_active_response_disabled: Disables active response if set to yes. If this is not defined active response is enabled.
  ossec_disable_public_repos: Disables installation of public repositories if set to "yes".
  ossec_agent_package_name: Default is "ossec-hids-agent". This can be set to a URL or path to a .rpm file or path to a .deb file if the public repositories cannot be used.
  ossec_agent_name: Optional name for the OSSEC agent. Default is to use hostname.
Ansible OSSEC Agent

• Example playbook:

  - hosts: all:!ossec-server.example.com
    roles:
      - { role: dj-wasabi.ossec-agent, ossec_server_ip: 192.168.1.1, ossec_server_name: ossec-server.example.com }

• Install using `ansible-galaxy install`:

  $ ansible-galaxy install dj-wasabi.ossec-agent
Ansible OSSEC Server

- OSSEC Server Example Setup for host variables located in `host_vars/ossec-server`:

```yaml
ossec_server_config:
  mail_to:
    - me@example.com
  mail_smtp_server: mail.example.com
  mail_from: ossec@example.com
  frequency_check: 72000
  ignore_files:
    - /etc/mtab
    - /etc/mnttab
    - /etc/hosts.deny
```

Note, full list here: https://galaxy.ansible.com/dj-wasabi/ossec-server
The UI

- I no longer have to argue with Daniel about the crappy WUI...
The UI

- I no longer have to argue with Daniel about the crappy WUI...
Which Cloud Should I Use For...

- OSSEC Management?
- UI/Elasticsearch?
- Long term storage?
Which Cloud Should I Use For...

- **OSSEC Management?**
  - *Pick highest disk I/O and network throughput*

- **UI/Elasticsearch?**
  - *Pick highest disk I/O and best mem price*

- **Long term storage?**
  - *Pick least expensive storage*
Final Architecture Tips

• Strongly suggest
  • Not sending OSSEC agent-server communications over public Internet - use a VPN between cloud providers,
  • Using a private management network for OSSEC agent-to-server communications within the same cloud provider,
  • Use one OSSEC server per cloud provider and forward alerts to a central repository (e.g. Elasticstack, Logz.io, Splunk, whatever),
  • Using a second private management network for OSSEC server-to-Elasticstack communications within the same cloud provider, and
Final Architecture Tips

• Strongly suggest
  • Not sending OSSEC agent-server communications over public Internet - use a VPN between cloud providers,
  • Using a private management network for OSSEC agent-to-server communications within the same cloud provider,
  • Use one OSSEC server per cloud provider and forward alerts to a central repository (e.g. Elasticstack, Logz.io, Splunk, whatever),
  • Using a second private management network for OSSEC server-to-Elasticstack communications within the same cloud provider, and
• NEVER LET DANIEL FORGET HOW BAD THE FIRST WUI WAS!
References

• How To implement A Host-Based Intrusion Detection System (HIDS) In The Cloud
  • https://www.built.io/blog/how-to-implement-a-host-based-intrusion-detection-system--hids-in-the-cloud

• How to Monitor Host-Based Intrusion Detection System Alerts on Amazon EC2 Instances

• Wazuh documentation
  • https://documentation.wazuh.com/current/index.html

• A Cloud Services Comparison Of The Top Three IaaS Providers
  • https://www.cloudhealthtech.com/blog/cloud-services-comparison
Thank You!

Questions?

Visit Us At:
https://www.leocybersecurity.com