SESSION ID: CSV-R14

FIM AND SYSTEM CALL AUDITING AT SCALE IN A LARGE CONTAINER DEPLOYMENT

Ravi Honnavalli
Staff Engineer
Walmart
Twitter handle: @ravi_honnavalli
NOTE: All content discussed here are out of self learning and not related to the work I do at Walmart.
Ever increasing amount of logs
Overwhelming amount of choices

Too many options!!

Static rules?

SIEM tools

ML?

Event source?

Agent vs Agentless
Flood of OSS tools

- fluentd
- Kolide
- Beats
- Kibana
- Elasticsearch
- go-audit
- SIEM tools
- Logstash
- osquery
- journald
- Audittd
- Doorman
- Zentral
- ElastAlert
- Unstructured datastores
- TensorFlow
GOAL: Demystifying the choices we have

Understanding types of event sources
- Classifying event sources
- Understand event source type
- Understand how they work in containers

Build our own stack based on insight needed
- Understand the insights we are looking for
- Build a stack based on the event classification
- If needed customize existing open source tools
- Build adaptors / tools that join the whole chain

Make an informed decision
- The stacks discussed in this presentation are by no means the only stack available

The stacks discussed in this presentation are by no means the only stack available

Make an informed decision
Possibilities of tools evaluated

**Deployment tools**
- Chef
- Puppet
- Ansible

**Data shippers**
- Logstash
- Filebeat
- Fluentd

**Audit tools**
- auditd
- go-auditd
- go-audit-container
- osquery

**Sink**
- File
- Syslog

**Fleet manager**
- Zentral
- Kolide
- Doorman
- Hand crafted tool

**Unstructured data stores**
- Elasticsearch
- MongoDB

**Graphing and reporting**
- Kibana
- Grafana
- ElastAlert
- Custom tools based on query DSL/Lucene

**Data Preprocessing**
- Custom tool to build a training set

**ML**
- TensorFlow
- Layer depth
- Optimization algorithm
- Learning rate
- Gradient descent mechanisms

**Data Preprocessing**
- Custom tool to build a training set
Classification of event sources

- Event sources
  - Event based
    - syscall
  - Scheduled query
    - inotify
    - agent
Security insight based on event source type

syscall

Looking for specific outliers among mostly normal dynamic events.
- Like identifying outliers
- Monitoring constantly for a specific malicious system call along with other criteria (uid, etc)

inotify

Safe-guarding specific sensitive files / area in the file system
- Watch for CREATE/ACCESS/MODIFY/DELETE events on specific files

agent

Scheduled activities for static information
- OS patch level queries, vulnerable kernel modules, mis-configuration
SYSTEM CALLS
Why syscall?

- Fundamental transit points between user land and kernel
- Every process makes system calls disclosing information of its activity
- Several user space tools that send audit information (auditd, go-audit, go-audit-container)
- Can provide deep insight when aggregated and drilled down
- Ideal candidate to build a machine learning training set as the volume of data is huge
Audit component

Namespacing concern!

Reporting daemon

Netlink socket

Kauditd

Syscall interface

User space application

User land

Kernel land
Container anatomy

Container 1

Container 2

Container 3

Kernel

Audit, syslog, kernel key ring
go-audit-container
Demo: Privilege escalation
Inotify component

- `Inotify_add_watch`
- `Watch list`
- `User space application`
- `Event queue`
- `User land`
- `Kernel land`

*Inotify sysctls are not namespaced!*
Why inotify?

- Lesser CPU consumption on an average
- Missing details in the reports
inotify based stack for FIM

- **Elasticsearch**
- **Grafana**
- **Kibana**
- **Pre-Processsink**
- **TensorFlow**
- **Email**
- **Pagerduty**
- **Slack**

**osquery**

**Register watch**

**Notify event**

**User land**

**Kernel land**

- **Inotify component**
AGENTS
osquery

Distributed Query

Fleet Manager

osquery

OS

osquery

OS

osquery

OS

osquery

OS
Osquery stack to get insights at scale
Compliance query

NIST
Information Technology Laboratory
NATIONAL VULNERABILITY DATABASE

 проблемы
CVE-2016-8610 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

A denial of service flaw was found in OpenSSL 0.9.8, 1.0.1, 1.0.2 through 1.0.2h, and 1.1.0 in the way the TLS/SSL protocol defined processing of ALERT packets during a connection handshake. A remote attacker could use this flaw to make a TLS/SSL server consume an excessive amount of CPU and fail to accept connections from other clients.

Source: MITRE

QUICK INFO

CVE Dictionary Entry:
CVE-2016-8610
NVD Published Date:
11/13/2017
NVD Last Modified:
01/11/2018
East West Threat

Probe process_open_sockets

status: Active

description: select distinct remote_address, remote_port, local_address, local_port, pid from process_open_sockets where remote_address <> '' and local_address <> '' and remote_port <> '0' and remote_address <> '127.0.0.1';

Filters

Add filter -

Inventory

Business unit: demo

Osquery distributed query

Query:

```sql
SELECT DISTINCT remote_address, remote_port, local_address, local_port, pid
```
osquery packs
Learning from using fixed queries in Kibana, Grafana and custom tools

Fixed queries need to be constantly updated for new patterns

Robust rules need a lot of queries

Any small variation of the rules is a false negative

Machine learning helps improve our ability to detect anomalies and broadly classify security posture
MACHINE LEARNING
High level differences

**Unsupervised**
- Elasticsearch ML
- Anomaly detection
- Time series data

**Supervised**
- Pre-processink-osquery
- Explicit labelling and pre-processing
- Explicit data classification on disparate info
Elasticsearch ML: Detecting outliers

Picture credit: https://unsplash.com/@ripato
Demo of ElasticSearch ML
Use case: Supervised Learning

Classifying data from different event sources

Broad classification into RED/YELLOW/GREEN

Classifying to have a big picture of the security posture of the organization
Supervised learning: Building a training set is key

Elasticsearch → Preprocessink-osquery → Labelling → TensorFlow
Pre-processink-osquery stages

**Stage 0**
- Query one probe at a time from ES
- Label into RED/YELLOW/GREEN
- Write to stage_0.csv

**Stage 1**
- Merge into existing stage_1.csv

**Manual Labeling**
- At this stage human administrator can manually label events that were not labeled or which were incorrectly labeled by the automated rules

**Stage 2**
- Transform into numeric values, normalizing and mean centering
ML choices if you are building your own solution

- Activation function
- Optimizers
- Learning rate
- Batch size vs iterations
- Depth of the network
Results of our experiment

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReLU activation</td>
<td></td>
</tr>
<tr>
<td>Batch Size</td>
<td></td>
</tr>
<tr>
<td>Adam optimizer</td>
<td></td>
</tr>
<tr>
<td>Lower the learning rate the better</td>
<td></td>
</tr>
</tbody>
</table>
Lessons learnt

Identify event sources keeping in mind nature of events needed, frequency, memory and CPU footprint of the agents on the application nodes, etc.

Be aware of how and what is namespaced in containers when configuring event sources in containers.

High level classification of ML

Machine learning helps in getting intelligent insights, but needs training and fine tuning.
Start with a simple File Integrity monitoring implementation using inotify log. Observe load of FIM events on the infrastructure.

Grow the solution to more detailed monitoring. Keep in mind load on the infrastructure, when thousands of nodes start sending events.

Try applying ML based on unsupervised learning and then get your hands dirty with training.

Think of possibilities outside of what is discussed here today.
AuditNG suite

https://github.com/auditNG/preprocessink-osquery

https://github.com/auditNG/go-audit-container
Questions?

You can also reach out later:
Twitter handle: @ravi_Honnavalli
LinkedIn:
https://www.linkedin.com/in/ravi-honnavalli-0535163/