Working at Booking.com “In God We Trust, all other must bring data.”
Aggressive A/B Testing. Focus on failing fast, understanding the failure, so we can fail again in new way.
From 30,000 Feet

- The Joy in Obsoleting Yourself
- OSSEC Instrumentation
- Instrumentation with OSSEC
- Pitfalls and Caveats
- Q & A
The Joy of Obsoleting Yourself is mostly getting to the pub before everyone else.
This is something that’s understood in the devOps Community
“Write Infrastructure like code, check it in, revert.”
Added benefits of Audit Trails for Compliance Initiatives
Make deploying infrastructure through CM tools easier than manually deploying.
Rinse, Repeat.

I hate doing something more than once and will start writing a script the second time I see the same problem.
Do I need to say more?
Easy Solution

- Server
  - ossec-authd
- Clients
  - agent-auth

Or is it?
Agents still need manual run of agent-auth!
Simply runs the “command” if the “creates” file does not exist. If it changes, “notify” informs the ‘ossec-hids’ service to start or restart based on it’s current status.
“Sure you have OSSEC installed, but how do you know it’s working?”
If you don’t have Graphite/OpenTSDB/InfluxDB to record Security Metrics, drop everything and set that up.
This is how easy it is to record metrics in Graphite.

Simple Graphite Script

```
#!/bin/bash
# Crontab: * * * * * /path/to/this-script.sh
LIST_AGENTS='/var/ossec/bin/list_agents'
CARBON_HOST='graphite.example.com'
CARBON_PORT=2003

prefix="security.ossec.$(hostname -s)"

all=$($LIST_AGENTS -a | wc -l)
connected=$($LIST_AGENTS -c | wc -l)

echo <<EOM
$prefix.available $all $ts
$prefix.connected $connected $ts
EOM | nc $CARBON_HOST $CARBON_PORT
```
So, by measuring and recording metrics, you can alert on them. But be careful to avoid “Alert Fatigue”
If you saw Vic’s presentation this should be pretty easy to recreate.

1. Send OSSEC Events to ElasticSearch
2. ..
3. Profit!
Installing, configuring, tuning, and tweaking ElasticSearch is another talk. For now, see:
https://github.com/reyjrar/es-utils/
Oh, you never did this? You might as well `rm -rf /var/ossec`
Abuses Puppet Facts which are realized **before** the catalog. So if the ossec_server_ip has changed, $::prev_ossec_server and $ossec_server_ip will vary.
1. Stop OSSEC
2. Remove existing keys **and** remove rids **before** running agent-auth
3. The agent-auth exec will run to communicate with the new ossec server!
Now Auto-Distribute!

```
$ossec_servers = exlookup('ossec_servers') # Now an array

file {
  '/etc/factor/facts.d/ossec.txt':
    content => template('ossec/ossec_server_fact.erb');
    require => Service['ossec-hids'];
}

# ossec_server_fact.erb
<%=
  uuid = scope.lookupvar('::uuid');
  # Convert HEX to Integer
  seed = [uuid].pack('H*').unpack('l')[0];
  # Use UUID as Random Seed
  srand(seed);
  # Get seeded random number in range
  idx = rand(0 .. @ossec_servers.length-1);
-%>
pref_ossec_server=<% @ossec_servers[idx] %>
```

Change the ossec_server to an array and use an unchanging UUID to seed a random function. Randomly distribute the clients to the ossec servers.
If these guys had thought of this, the planet may not have been hacked.
Similarities between OSSEC and CM Tools

- Configuration Management has States
- Configuration Files
- Application Versions
- Resource Status and Definitions
- OSSEC has States too!
  - Log data
  - System status
  - Process status
  - Network status

Instrumentation with a Security Tool?
Similarities between OSSEC and CM Tools

- Configuration Management has Actions
- Resource CRUD
- Also “Run this script, kthxbye.”
- OSSEC has Actions too!
- ActiveResponse!!!
- Mostly, “Run this script, kthxbye.”

Instrumentation with a Security Tool?
I hate it, you hate it, it sucks, let’s make it go away.

File Integrity Monitoring

- Noisy, at best
- Things messing with your files are mostly legit:
  - System Updates
  - Configuration Management
  - Software Deployments
  - Sysadmins Saving the Day

I hate it, you hate it, it sucks, let’s make it go away.
This is a daily happening. Commit code, it’s amazing how frequently stuff is needed again.

OSSEC v2.8+
- ActiveResponse passes alert->filename
- Write a script which does your job
- Commit that script
- Deploy that script
- Re-use that script
  - Automatically!
Game Plan

- Demote FIM Alerts to Level 1, disable email
- Fire an ActiveResponse that takes filename
- Emit a new log message in our script
- Decode the new log message
- Alert / Log based on that
- ...
- PROFIT!
Log everything Level 1 and up (to retrieve data).
Stop sending me emails below Level 09 – Error from invalid source
Never ignore FIM events.
And monitor new files added to monitored directories.
This isn’t perfect, but the idea is splay the time the syscheck daemon’s run to prevent message storms to the ossec servers.
Disable all syscheck emails, this isn’t perfect as it impacts more than just FIM events. We use ElasticSearch and OSSEC Reports for other stuff we’re interested in.

no_email_alert makes sure alerts attempting to override email alerting levels STFU.
Yay! or wait.. there’s more.

Problem Solved!
No more emails!
Added in 2.8, send the filename to the Active Response script. Fire on all events that matched our new rule “106002”
Verify Logs

Sep 15 00:26:10 ether ossec-ar-verify: file ok (/etc/cron.d/puppet-job)

Sep 15 00:26:10 ether ossec-ar-verify: file managed by RPM (/etc/mcollective/facts.yaml) changed outside of RPM

Sep 15 00:26:10 ether ossec-ar-verify: file unmanaged (/etc/postfix/aliases.db) changed

This is a sample of the format of the messages the active response verify script will syslog.
Build a simple decoder, we’re not using the “action” element because we can’t in rules, but in theory, we could, one day!
Set the group to verify and be a placeholder. OSSEC rules follow a tree structure if possible. It GREATLY improves performance to use parent rules like this.

Parent Rule

```
<rule id="107000" level="1">
  <decoded_as>ossec-ar-verify</decoded_as>
  <description>Verification</description>
  <group>verify</group>
</rule>
```
03 – Successful/Authorized events – They include successful login attempts, firewall allow events, etc.

Everything is OK

<rule id="107001" level="3">
  <if_sid>107000</if_sid>
  <match>file ok</match>
  <description>File was changed intentionally.</description>
</rule>
Managed File Change

<rule id="107002" level="11">
  <if.sid>107000</if.sid>
  <match>*</match>
  <description>Verified: Unauthorized File Change</description>
</rule>

Level 11 – Integrity checking warning
Level 11 – Integrity checking warning
Does not email, instead we roll these up by filename everyday and investigate how to either move them to a managed system or ignore them.
OSSEC FIM Results

96% Reduction in Alerting

Pretty good, but still too many alerts for humans to make sense of!
Pitfalls and Caveats

- Who controls inputs?
- How resource intensive are your checks?
  - What if 1,000,000 fire simultaneously?
    - On the same server?
  - Think, test, then get some to try to break it.

Input sanitisation
Fencing?
Review scripts in ActiveResponse
An example of how inputs and assumptions can bite you in the ass.

CVE-2014-5284

- host-deny.sh created files in /tmp
- cp /tmp/hosts.$$deny /etc/hosts.deny
- Didn’t properly manage permissions
- Would copy, as root, the contents of that file to /etc
- Moved from /tmp to /var/ossec and added randomness to file name
Find Me

GitHub: https://github.com/rejrar/
Twitter: @rejrar
Blogging: http://edgeofsanity.net
Email: brad.lhotsky@gmail.com