Security Best Practices As Code

#AssimProj  @OSSAlanR
http://assimproj.org/

Alan Robertson <AlanR@AssimilationSystems.com>
Assimilation Systems Limited
http://AssimilationSystems.com
Biography

- 35+ years in IT/development – 10 years in system management (SysAdmin)
- Founded Linux-HA project - led 1998-2007 – aka “Heartbeat” - now called Pacemaker
- Founded Assimilation Project in 2010
- Founded Assimilation Systems Limited in 2013
- Alumnus of Bell Labs, SuSE, IBM
Assimilation Project Evolution

- Inspired by 2 million core computer (cyclops64)
- Concerns for extreme scale
- Topology aware monitoring
- Topology discovery w/out security issues
  ➞Discovery of everything!
- Take actions based on discovery!
A 6-dimensional overview

1. System Management Suite Overview
2. Basic Technology
3. Best Practice Analyses
4. “Toy” Best Practice Demo
5. Current Status
6. What You Need To Do!
What's in the Suite?

- Graph CMDB
- Exception Monitoring
- Security
- Network Connections
Disturbing Trends...

- 30% of all break-ins come through “lost” systems (Verizon)
- 90% have had failures of unmonitored services (Turnbull)
- 80% are unable to keep systems in compliance (Verizon)
- 30% start monitoring only after a problem (Turnbull)
- 30% of all systems are doing nothing useful (Koomey)
- Larger site admins often don’t know dependencies
Why the Assimilation Suite?

- Provides insight and details through a graph-model CMDB
- Helps you understand and automate your environment
  - Reduce Errors
  - Speed up problem resolution
- Reduces Manual Documentation
- Discovery-driven configuration => near-zero configuration
- Automates Monitoring
- Enhances Security
- Designed for Extreme Scale (≈100K systems)
“Complexity is the enemy of reliability”

- **Complexity likely your single biggest problem**
  - Near-zero configuration reduces complexity
  - Seamless integration reduces complexity
  - Accurate detailed view improves complexity management
Highly Scalable Discovery-Driven Automation

- Continuous extensible discovery (CMDB)
- Extensible exception monitoring
- Discovery Drives (Security) Best Practice Analyses
- All data goes into central graph CMDB
This all sounds unreasonable...

- Huge scalability without complexity.
- Discovery without pings or port scans.

Really?
Simple Scalability

I can explain how we scale so your grandmother would understand...
Massive Scalability – or "I see dead servers in $O(1)$ time"

- Adding systems does not increase the monitoring work on any system
- Each server monitors 2 (or 4) neighbors
- Each server monitors and discovers its own services
- Ring repair and alerting is $O(n)$ – but a very small amount of work

Current Implementation
Service Dependency Graph
Assimilation Architecture

- **Central Collective Management Authority**
  - written in Python
  - delegates most work to nanoprobes
- **Fully distributed “nanoprobe” agents**
  - Simple, policy-free
  - Written in 'C'
  - Invoke scripts for monitoring or discovery
  - Send/receive heartbeats
  - Listen for ARP, LLDP, CDP packets
- **Neo4j graph database**
Switch Discovery Graph from LLDP / CDP
Sample Security Features

- Discovery of “forgotten” IP addresses
- Monitoring of Open Ports and Services
- Collection of network-facing app checksums
- Nmon OS profiling of new MAC addresses
- Checksum outlier analysis
- Security Best Practice Analyses
Best Practice Analyses

This is next major *planned* capability

- Triggered by Discovery Updates
  - Analysis occurs within seconds of change
  - No change => No analysis
- We can analyze anything discovered
- Expect to create alerts and reports
Sample Security Best Practices

- Inappropriate OS settings (hardening)
- Security Patch Coverage
  - OS vendor (RedHat, SuSE, Canonical, etc)
  - Application (Oracle, IBM, WordPress, etc)
- Inappropriate services (telnet, etc)
- Inappropriate Application Settings (hardening)
IT Best Practices Project

ITBestPractices.info

- **IT-Bestpractices** GitHub project
- Apache 2 License
- Initial Sources
  - NIST STIGs
  - Lynis project
  - Individual contributions
IT Best Practices Goals

- Make Best Practice rules available in JSON
  - Curate potentially mechanically-verifiable practices
  - Human-readable descriptions of issues and remedies
  - Multiple language support
  - Not limited to security best practices
  - Eventually available through a web server
  - Source text may change to YAML for readability
The system must limit the ability of processes to have simultaneous write and execute access to memory.
The status of the "kernel.exec-shield" kernel parameter can be queried by running the following command:

```
$ sysctl kernel.exec-shield
```

```
$ grep kernel.exec-shield /etc/sysctl.conf
```

The output of the command should indicate a value of "1". If this value is not the default value, investigate how it could have been adjusted at runtime, and verify it is not set improperly in "/etc/sysctl.conf". If the correct value is not returned, this is a finding.
ExecShield uses the segmentation feature on all x86 systems to prevent execution in memory higher than a certain address. It writes an address as a limit in the code segment descriptor, to control where code can be executed, on a per-process basis. When the kernel places a process's memory regions such as the stack and heap higher than this address, the hardware prevents execution in that address range.
“nist_V-38597”:

{"rule": "EQ($kernel.exec-shield, 1)"}
Assimilation Networking Rule

Buffer bloat prevention

“itbp-0001”:
{
    “rule”: “IN($kernel.core.default_qdisc, fq_codel, codel)”
}

© 2015 Assimilation Systems Limited
“Toy” Best Practices Demo

- Demo is of test code in our source tree
- Test data was discovered from a real system
- This code was written for feasibility testing
Current Status

- 1.0 (Independence Day) release out 4 July 2015
- Security is next future emphasis
- Great unit *and system* tests
- Strongly encrypted communication
- Several discovery methods written
- Extensible Automated Discovery Triggers
- Discovery => Automatic Monitoring + Network-Facing Checksums
- Command Line Queries
- Licenses: Commercial or GPLv3
- Compatible with Nagios remote monitoring agent API
- Best practice analysis work underway
Get Involved!

- Early adopters – customers!
- Contributors
  - Testers, Continuous Integration
  - Best practice experts
  - Designers
  - Developers (C, Python, Shell, PowerShell, JavaScript)
  - Porters (esp Windows)
  - Promoters, Publicists, Packagers, etc.
Assimilation BOF TONIGHT!

- Get All Your Questions Answered
- TONIGHT!
- 7-8 PM (1900-2000)
- Room E145
Resistance Is Futile!

These slides: bit.ly/AssimOSCON15
Mailing List: bit.ly/AssimML
OSSAAlanR
#assimilation on irc.freenode.net

Project Web Site: assimproj.org
Company Web Site: assimilationsystems.com
Download: assimilationsystems.com/download
Minimizing Network Footprint (in roadmap)

- Support diagnosing switch issues
- Minimize network traffic
- Ideal for multi-site arrangements
Discovery / Monitoring Demo

- Demonstrate basic capabilities
  - Discovery
  - Discovery-driven monitoring configuration
  - Discovery-driven 'tripwire-like' checksums
  - Monitoring – failures / successes
  - Host down notification
- No configuration was supplied
  - *everything* comes from discovery

http://assimilationsystems.com/90_second_demo/
Risk Management/Mitigation

- Intrusions
- Vulnerable Software
- Licensed Software
- Audit Risk
- Outages
- System management
Why a graph database? (Neo4j)

- Humans describe systems as graphs
- Dependency & Discovery information: graph
- Speed of graph traversals depends on size of subgraph, not total graph size
- Root cause queries ⇒ graph traversals – notoriously slow in relational databases
- Visualization is Natural
- Schema-less design: good for constantly changing heterogeneous environment

- Graph Model === Object Model
Monitoring Pros and Cons

Pros
Simple & Scalable
Uniform work distribution
No single point of failure
Distinguishes switch vs host failure
Easy on LAN, WAN
Multi-tenant approach

Cons
Active agents
Potential slowness at power-on
Two Schema subgraphs

- Client / server dependency
- Switch interconnect
"sshd": {
  "exe": "/usr/sbin/sshd",
  "cmdline": [ "/usr/sbin/sshd", "-D" ],
  "uid": "root",
  "gid": "root",
  "cwd": "/",
  "listenaddrs": {
    "0.0.0.0:22": {
      "proto": "tcp",
      "addr": "0.0.0.0",
      "port": 22
    }
  }
}
"ssh": {
  "exe": "/usr/sbin/ssh",
  "cmdline": [ "ssh", "servidor" ],
  "uid": "alanr",
  "gid": "alanr",
  "cwd": "/home/alanr/monitor/src",
  "clientaddrs": {
    "10.10.10.5:22": {
      "proto": "tcp",
      "addr": "10.10.10.5",
      "port": 22
    }
  }
}
Problems Addressed

- Discovering and maintaining documentation (CMDB) using continuous discovery
  - Services, Systems, Dependencies, Switches, Interconnects, Configuration
- Monitoring and alerting: services, systems and compliance
- Managing compliance
- Mitigating risk
Why Discovery? (DevOps)

- **Documentation**: incomplete, incorrect
- **Dependencies**: unknown
- **Planning**: Needs accurate data
- **Best Practices**: Verification needs data
- **ITIL CMDB** (Configuration Management Data Base)

Our Discovery: continuous, low-profile
Second Dimension:
Unique Powerful Features

1. Continuous Discovery
2. Discovery: Zero network footprint
3. Centralized graph database
4. We know everything that changes
5. Discover and update dependency information
6. Discovery and monitoring tightly integrated – discovery drives automation
(even more) Features...

7. Discovery and monitoring easily extensible
8. Naturally scalable to > 100K systems
9. Minimal network load
10. Server failures distinguishable from switch failures
11. Best practice and vulnerability alerts
12. Multi-tenant support
Third Dimension: Fully distributed work

Two philosophical underpinnings

1. Monitoring and Discovery are *fully* distributed

2. *Reliable* “no news is good news”

Only *responses* to changes are centralized
Service Monitoring based on HA Technologies

- Well-proven architecture:
  - *reliable “no news is good news”*
- Implements Open Cluster Framework standard (LSB and others – *Nagios coming!*)
- Each system monitors own services
- Can also start, stop, migrate services
How does discovery work?

Nanoprobe scripts perform discovery
- Each discovers one kind of information
- Can take arguments from environment
- Output JSON

CMA stores Discovery Information
- JSON stored in Neo4j database
- CMA discovery plugins => graph nodes and relationships
# A Few Canned Queries

<table>
<thead>
<tr>
<th>Query</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allipports</td>
<td>get all port/ip/service/hosts</td>
</tr>
<tr>
<td>allswitchports</td>
<td>get switch connections</td>
</tr>
<tr>
<td>crashed</td>
<td>get crashed servers</td>
</tr>
<tr>
<td>shutdown</td>
<td>get gracefully shutdown servers</td>
</tr>
<tr>
<td>downservices</td>
<td>get nonworking services</td>
</tr>
<tr>
<td>findip</td>
<td>get system owning IP</td>
</tr>
<tr>
<td>findmac</td>
<td>get system owning MAC</td>
</tr>
<tr>
<td>unknownips</td>
<td>get unknown IP addresses</td>
</tr>
<tr>
<td>unmonitored</td>
<td>get unmonitored services</td>
</tr>
</tbody>
</table>
OS discovery JSON Snippet

```json
{
    "nodename": "alanr-1225B",
    "operating-system": "GNU/Linux",
    "machine": "x86_64",
    "processor": "x86_64",
    "hardware-platform": "x86_64",
    "kernel-name": "Linux",
    "kernel-release": "3.8.0-31-generic",
    "kernel-version":="#46-Ubuntu SMP ...",
    "Distributor ID": "Ubuntu",
    "Description": "Ubuntu 13.04",
    "Release": "13.04",
    "Codename": "raring" 
}
```