DevSecCon

Infrastructure Security Assurance with Chef InSpec

MANDI WALLS

Seattle | September 16-17, 2019

HI!

- Mandi Walls
- Developer Advocate at Chef
- mandi@chef.io
- @lnxchk
- https://www.chef.io/
- https://www.inspec.io/

CHEF INSPEC™

EVERY business is a software business

> We’re going to be a software company with airplanes.
>
> – CIO, Alaska Airlines

CHEF INSPEC™

ATTACKS/BREACHES

6/21/2017
05:15 PM

Jai Vijayan
News

Connect Directly

1 COMMENT
COMMENT NOW

Login

WannaCry Forces Honda to Take Production Plant Offline

Work on over 1,000 vehicles affected at automaker's Sayama plant in Japan while systems were restored.

In an example of just how persistent modern cyberthreats can be, automaker Honda Motors had to temporarily stop production at its Sayama plant in Japan this week after being hit by WannaCry, a malware threat the company thought it had mitigated just one month ago.

The nearly 48-hour shutdown impacted production of about 1,000 vehicles at the facility, which does engine production and assembly for a line of vehicles including the Odyssey minivan and the Accord.

A statement from Honda North America said the interruption at the Sayama Auto Plant was caused by the shutdown of several older production-line computers infected with the WannaCry virus.

CHEF INSPEC™

https://www.darkreading.com/attacks-breaches/wannacry-forces-honda-to-take-production-plant-offline-/d/d-id/1329192

MENU DataCenter Knowledge

SEARCH LOG IN REGISTER

RECENT

Why Private Connections Will Soon Carry Most B2B Traffic
SEP 18, 2018

Time to Rebuild Alpine Linux Docker Containers After Package Manager Patch
SEP 17, 2018

Hadoop Architecture: Is the Bloom off the Rose?
SEP 17, 2018

Top 5 Data Center Stories of the Week: September 15, 2018
SEP 15, 2018

SPONSORED CONTENT
Flexible modern data centers demand flexible PDUs

The Providence St. Joseph

SHARED

Botched Server Install Results in $2.14 Million HIPAA Breach Fine

Failure to change a default setting on a new server caused private patient information to be publicly accessible over the Internet for an entire year.

Aldrin Brown | Oct 19, 2016

MSPmentor®
The Ultimate Guide to Managed Services℠

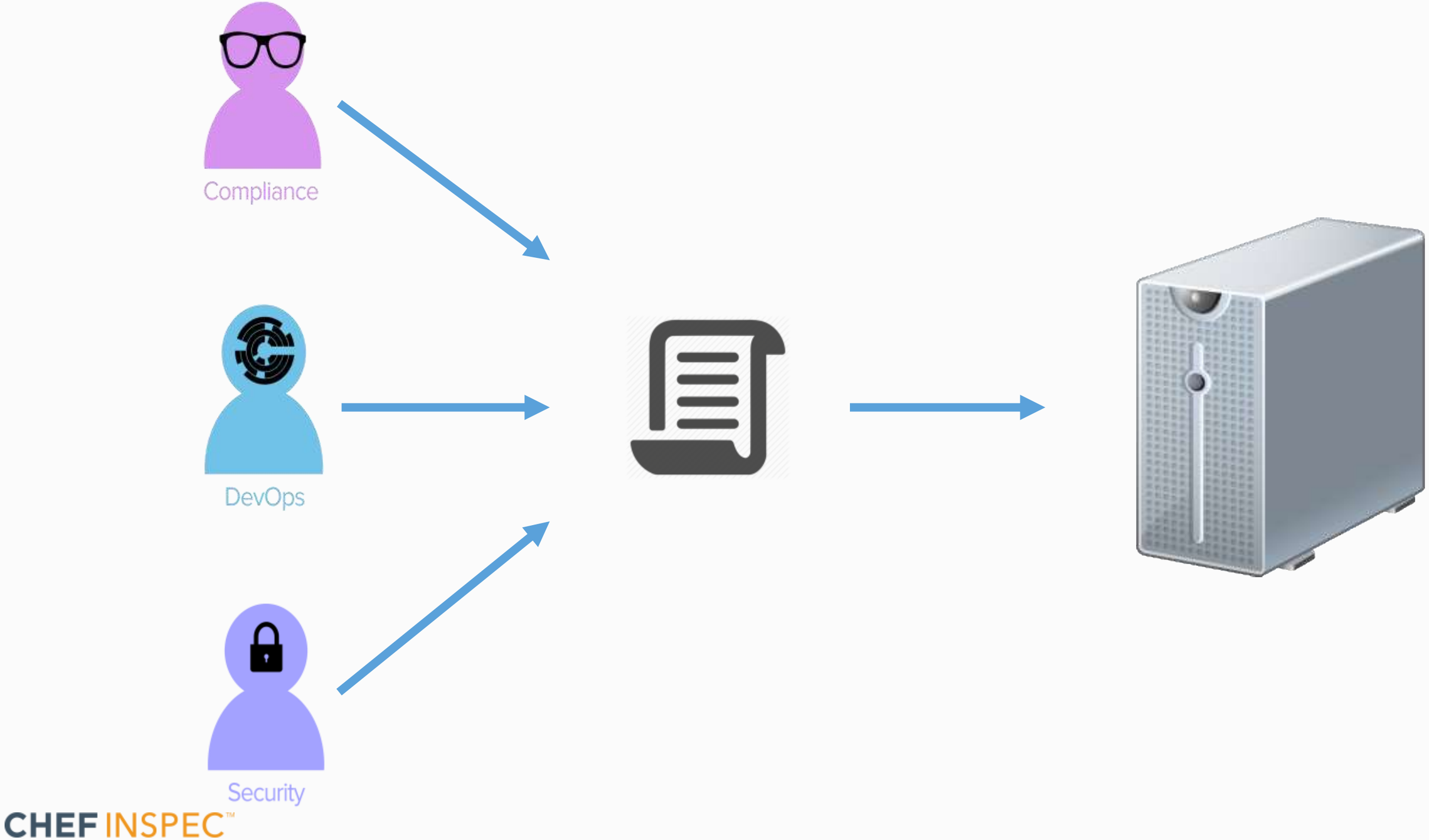
Brought to you by MSPmentor

CHEF INSPEC™

https://www.datacenterknowledge.com/archives/2016/10/19/botched-server-install-results-in-2-14-million-hipaa-breach-fine

Different Sources for the Same Goals

CHEF INSPEC™


Compliance
DevOps
Security
CHEF INSPEC™

Chef InSpec

- Human-readable language for tests related to security and compliance
- Create, share, and reuse complex profiles
- Extensible language - build your own rules
- Command-line tools plug into your existing workflow, build, deploy
- Test early, test often!

CHEF INSPEC™

Create and Consume

- Complex compliance requirements can slow you down
- Different information and expertise live in different teams, but need to be used by many
- Security and compliance personnel can work with operations and development to create comprehensive profiles

Chef InSpec is Code

- Check it into repos, publish as artifacts
- Include InSpec steps before code checkin
- Include InSpec steps in integration and pre-production
- Continue InSpec checks in production to guard against new threats

Network Services

- If your security team sends you a directive:

  Ensure that no legacy network services are installed on all versions of Linux, including inetd, xinetd, telnet, rsh, tftp, and ypserv.

How Do You Go About Checking and Fixing?

- Identify the package names on your systems
- Remove all packages
- What’s the plan for the currently used images?
  - Rebuild for new images?
  - Remediate at launch and hope nothing gets in before the updates?
- Ensure it doesn't get re-installed by accident at some point in the future

Check for `inetd` and `xinetd`

```
control 'package-01' do
      impact 1.0
      title 'Do not run deprecated inetd or xinetd'
      desc 'rhel5-guide-i731.pdf, Chapter 3.2.1'
      describe package('inetd') do
            it { should_not be_installed }
      end
      describe package('xinetd') do
            it { should_not be_installed }
      end
end
```

CHEF INSPEC™

Chef InSpec Components

- Resources
- Resource Characteristics
- Profiles
- Command Line Interface

Resources

- Chef InSpec includes built-in resources for common services, system files, and configurations
- Built-in resources work on several platforms of Linux.
  - There are also Windows-specifics like `registry_key`
- A resource has characteristics that can be verified for your requirements, and Matchers that work with those characteristics

Sample Resources

- System resources:

  `directory`, `file`, `user`, `group`, `crontab`, `service`, `package`
- Specific services:

  `apache, nginx, rabbitmq, postgresql, IIS`
- Programming language components:

  `gem, npm, powershell`
- Network services:

  `port, http, sshd`
- https://www.inspec.io/docs/reference/resources/

Characteristic Tests

- it { should exist } – files, directories, groups that are present
- it { should be_installed } – packages that should be installed
- it { should be_enabled } – services that should be running
- its('max_log_file') { should cmp 6 } – rotate auditd logs

  Check inside a config file for a specific setting
- its('exit_status') { should eq 0 } – run any arbitrary checks

  Remediation scripts from upstream and OS vendors often come as shell

Run Chef InSpec

- InSpec is a command line tool
  - Installs on your workstation as a ruby gem or as part of the ChefWorkstation
- Can be run locally, to test the machine it is executing on
- Or remotely
  - InSpec will log into the target and run the tests for you

Lifecycle – How Often Do You Check Security?

- Single big scan, report mailed out with a “due date”?
  - Considered done, not checked again
- Yearly or twice-yearly massive scans with remediation firedrills?
  - Common audit cycles, large projects around fixing found issues
- Part of the software development lifecycle?
  - “To the left”
  - Regularly part of what is included in builds

Add InSpec to Build and Production Workflows

- Run InSpec on build nodes
  - Ensure they meet your requirements before builds are executed
  - Run smaller targeted profiles on code check-in
- Run InSpec in your integration environments
  - Ensure no new settings, configurations, app features violate your security before they get to prod
- Run InSpec in production
  - Verify your entire fleet on a regular basis – don't wait for the audit!
  - When a new vulnerability is announced, create a test and push to your hosts. Know in minutes how exposed you are
  - Use an agent for regular reporting, or targeted scans for spot-checking

Execute InSpec

```
$ inspec exec ./test.rb
Profile: tests from ./test.rb
Version: (not specified)
Target:  local://
  File /tmp
     ✓  should exist
     ✓  should be directory
     ✓  should be owned by "root"
     ✓  mode should cmp == "01777"

Test Summary: 4 successful, 0 failures, 0 skipped
```

CHEF INSPEC™

Test Any Target

Local: `inspec exec test.rb`

SSH Remote: `inspec exec test.rb -i ~/.aws/mandi_eu.pem -t ssh://ec2-user@54.152.7.203`

WinRM: `inspec exec test.rb -t winrm://Admin@192.168.1.2 --password super`

Docker Container: `inspec exec test.rb -t docker://3dda08e75838`

Profiles

- Collections of InSpec tests
  - Group by team, by application, by platform
- Each profile can have multiple test files included
- Flexible!
  - Create your own profiles for specific software you use
  - Use included matcher libraries or write your own – they live in the profile
- https://dev-sec.io/ for samples

Sample Profile: *linux-baseline*

```
control 'os-02' do
  impact 1.0
  title 'Check owner and permissions for /etc/shadow'
  desc 'Check periodically the owner and permissions for /etc/shadow'
  describe file('/etc/shadow') do
    it { should exist }
    it { should be_file }
    it { should be_owned_by 'root' }
    its('group') { should eq shadow_group }
    it { should_not be_executable }
    it { should be_writable.by('owner') }
...
```

CHEF INSPEC™

Demo

- Basic off-the-shelf CentOS system on AWS
- Install ChefWorkstation and git
- Download and run the *linux-baseline* profile
- Remediate with the corresponding Chef cookbook from https://dev-sec.io

Resources

- https://inspec.io
- https://blog.chef.io/category/inspec
- https://learn.chef.io/
- http://www.anniehedgie.com/inspec-basics-1
- Whitepaper featuring Danske Bank:
  https://www.chef.io/customers/danske-bank/
- Demo script: https://github.com/lnxchk/demos-inpec-2019H2

DevSecCon

Thanks!

More info:

https://chef.io/products/chef-inspec

https://inspec.io

CHEF™

Appendix: Demo Outputs

Select CentOS 7 from the Marketplace

CentOS 7 (x86_64) - with Updates HVM

★★★★☆ (61) | 1901_01 Previous versions | By Centos.org

$0.00/hr for software + AWS usage fees

Linux/Unix, CentOS 7 | 64-bit (x86) Amazon Machine Image (AMI) | Updated: 1/30/19

This is the Official CentOS 7 x86_64 HVM image that has been built with a minimal profile, suitable for use in HVM instance types only. The image contains just enough packages to ...

More info

- Use a small instance - .micro should be fine for this
- Tag X-Contact with your name and X-Customer with something like "InSpec Talk Delete after 7/15/19" or similar

Security Group

Security Groups

Edit security groups

Security Group ID	Name	Description
sg-f4ce8c81	default	default VPC security group

All selected security groups inbound rules

Type	Protocol	Port Range	Source	Description
All traffic	All	All	0.0.0.0/0	
All traffic	All	All	::/0	

- I use the default all-open security group, as there's nothing running but ssh on this machine. If you have another security group that is more locked down, that's fine, too.

Installs

- Install ChefWorkstation from https://downloads.chef.io/chef-workstation/

  curl –o cw.rpm <url>

  sudo rpm -ihv cw.rpm

- Install git via yum

  sudo yum install -y git

Demo stage 1 – Detect with the linux-baseline profile

```
git clone https://github.com/dev-sec/linux-baseline.git
sudo inspec exec linux-baseline/
     <<accept the product license here>>
```

You'll have some number of errors; the default installs will always have too many things installed. This version:

```
Profile Summary: 26 successful controls, 27 control failures, 1 control skipped
Test Summary: 80 successful, 45 failures, 1 skipped
```

CHEF INSPEC™

Demo Stage 2 – Correct with Chef Infrastructure

Download the Chef cookbook that matches the linux-baseline profile via a policyfile workflow

```
chef generate policyfile fix-security
<<accept the license>>
edit fix-security.rb
edit-> run_list 'os-hardening::default'
chef install fix-security.rb
chef export fix-security.rb harden-linux
cd harden-linux
sudo chef-client -z
```

CHEF INSPEC™

Correct with Chef con't

```
...things happening...
Recipe: os-hardening::auditd
  * yum_package[audit] action install (up to date)

Running handlers:
Running handlers complete
Chef Infra Client finished, 141/206 resources updated in 07
seconds
```

CHEF INSPEC™

Demo Stage 3 – Re-check with InSpec

```
cd ..
sudo inspec exec linux-baseline
...
Profile Summary: 52 successful controls, 1 control failure, 1 control skipped
Test Summary: 124 successful, 1 failure, 1 skipped
```

There's almost always at least one failure. Depending on the time you have left, you can work through the next part, creating a wrapper profile and skipping this step, or, conversely, if you audience is already chef-aware, adding an additional recipe to fix whatever it is.

CHEF INSPEC™

The error in this example:

```
× package-08: Install auditd (1 failed)
    ✓ System Package audit should be installed
    ✓ Audit Daemon Config log_file should cmp == "/var/log/audit/audit.log"
    ✓ Audit Daemon Config log_format should cmp == "raw"
    ✓ Audit Daemon Config flush should match
/^incremental|INCREMENTAL|incremental_async|INCREMENTAL_ASYNC$/
    × Audit Daemon Config max_log_file_action should cmp == "keep_logs"

    expected: "keep_logs"
         got: "ROTATE"

    (compared using `cmp` matcher)
```

CHEF INSPEC™

Demo stage 4 – prepare for Automate with wrapper profile

- Create a wrapper profile:

```
inspec init profile my-hardening
```

- Edit my-hardening/inspec.yml

```
depends:
  - name: linux-baseline
    git: https://github.com/dev-sec/linux-baseline
```

- Remove the example

```
rm -f my-hardening/controls/example.rb
```

Stage 4

Create a new control file:

```
$ vi my-hardening/controls/skip-auditd.rb

include_controls 'linux-baseline' do
  skip_control 'package-08'
end
```

Demo Stage 5 – run the wrapper profile

```
sudo inspec exec my-hardening


...
Profile Summary: 52 successful controls, 0 control failures, 1
control skipped
Test Summary: 113 successful, 0 failures, 1 skipped
```

CHEF INSPEC™

Wrapper Profiles

my-app-profile

```
control 'myapp-1'
control 'myapp-2'
control 'myapp-3'

include_controls 'my-baseline' do
  skip_control 'baseline-2'
end
```

my-baseline

control 'baseline-1'
~~control 'baseline-2'~~

CHEF INSPEC™