Testing infrastructure using Chef InSpec

Overview

In this tutorial, we will use Chef InSpec to perform testing against our infrastructure.

InSpec can be used to perform local or remote tests and compliance validation for auditing purposes.

Description

The working unit of InSpec is a profile that provides a versioned structure containing checks (i.e. tests); related checks are grouped in "control" blocks. Control is similar to having a high-level rule composed of multiple checks.

In order to check for compliance, the profile along with all respective controls (and corresponding checks) can be run against the local or remote infrastructure; it's also possible to run a single control file.

We'll use the profile dev-sec/linux-baseline that is available in InSpec's supermarket and run it against a remote host.

Please have a look at the control named "os-11".

https://raw.githubusercontent.com/dev-sec/linux-baseline/master/controls/os_spec.rb

```
#
# Copyright 2015, Patrick Muench
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# author: Christoph Hartmann
# author: Dominik Richter
# author: Patrick Muench

login_defs_umask = attribute('login_defs_umask', value: os.redhat? ? '077' : '027', description: 'Default umask
to set in login.defs')

login_defs_passmaxdays = attribute('login_defs_passmaxdays', value: '60', description: 'Default password
maxdays to set in login.defs')
login_defs_passmindays = attribute('login_defs_passmindays', value: '7', description: 'Default password mindays
to set in login.defs')
login_defs_passwarnage = attribute('login_defs_passwarnage', value: '7', description: 'Default password warnage
(days) to set in login.defs')

shadow_group = 'root'
shadow_group = 'shadow' if os.debian? || os.suse? || os.name == 'alpine'
container_execution = begin
                        virtualization.role == 'guest' && virtualization.system =~ /^(lxc|docker)$/
                      rescue NoMethodError
                        false
                      end

blacklist = attribute(
  'blacklist',
  value: suid_blacklist.default,
  description: 'blacklist of suid/sgid program on system'
)

control 'os-01' do
  impact 1.0
```

```
  title 'Trusted hosts login'
  desc "hosts.equiv file is a weak implemenation of authentication. Disabling the hosts.equiv support helps to
prevent users from subverting the system's normal access control mechanisms of the system."
  describe file('/etc/hosts.equiv') do
    it { should_not exist }
  end
end

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_not be_readable.by('other') }
  end
  if os.redhat? || os.name == 'fedora'
    describe file('/etc/shadow') do
      it { should_not be_writable.by('owner') }
      it { should_not be_readable.by('owner') }
    end
  else
    describe file('/etc/shadow') do
      it { should be_writable.by('owner') }
      it { should be_readable.by('owner') }
    end
  end
  if os.debian? || os.suse?
    describe file('/etc/shadow') do
      it { should be_readable.by('group') }
    end
  else
    describe file('/etc/shadow') do
      it { should_not be_readable.by('group') }
    end
  end
end

control 'os-03' do
  impact 1.0
  title 'Check owner and permissions for /etc/passwd'
  desc 'Check periodically the owner and permissions for /etc/passwd'
  describe file('/etc/passwd') do
    it { should exist }
    it { should be_file }
    it { should be_owned_by 'root' }
    its('group') { should eq 'root' }
    it { should_not be_executable }
    it { should be_writable.by('owner') }
    it { should_not be_writable.by('group') }
    it { should_not be_writable.by('other') }
    it { should be_readable.by('owner') }
    it { should be_readable.by('group') }
    it { should be_readable.by('other') }
  end
end

control 'os-04' do
  impact 1.0
  title 'Dot in PATH variable'
  desc 'Do not include the current working directory in PATH variable. This makes it easier for an attacker to
gain extensive rigths by executing a Trojan program'
  describe os_env('PATH') do
    its('split') { should_not include('') }
    its('split') { should_not include('.') }
  end
end
```

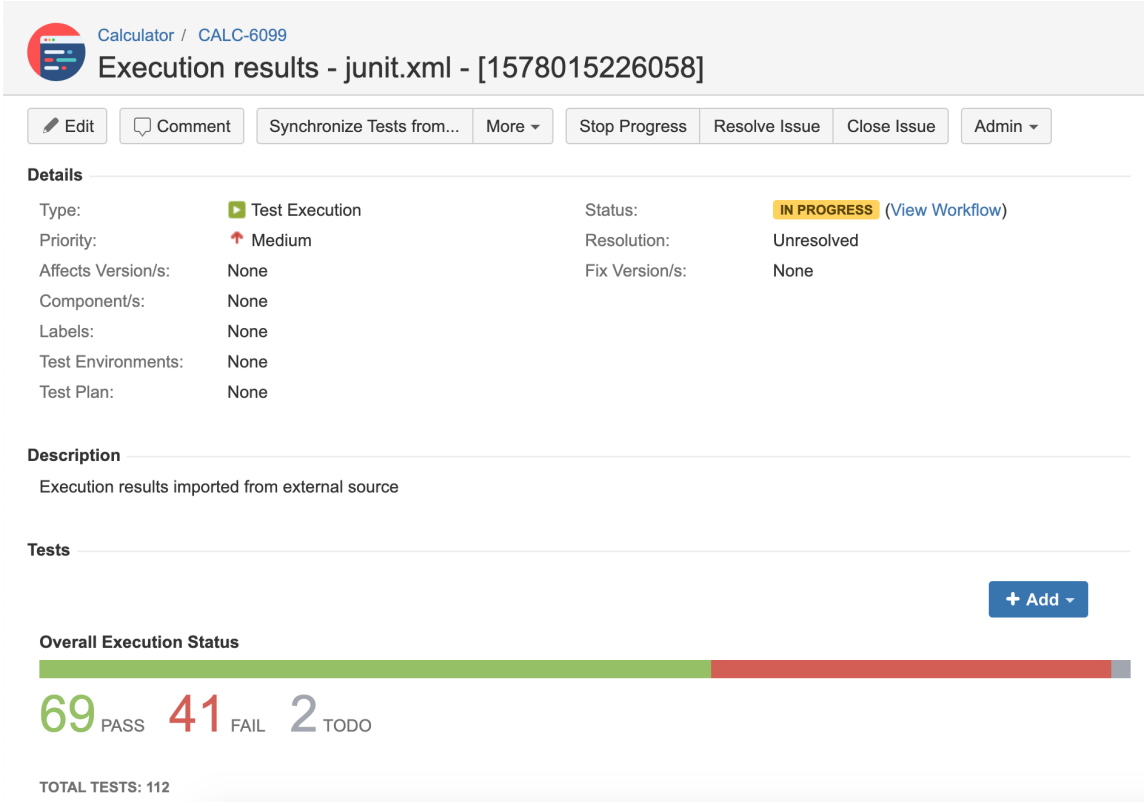
```
control 'os-05' do
  impact 1.0
  title 'Check login.defs'
  desc 'Check owner and permissions for login.defs. Also check the configured PATH variable and umask in login.
defs'
  describe file('/etc/login.defs') do
    it { should exist }
    it { should be_file }
    it { should be_owned_by 'root' }
    its('group') { should eq 'root' }
    it { should_not be_executable }
    it { should be_readable.by('owner') }
    it { should be_readable.by('group') }
    it { should be_readable.by('other') }
  end
  describe login_defs do
    its('ENV_SUPATH') { should include('/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin') }
    its('ENV_PATH') { should include('/usr/local/bin:/usr/bin:/bin') }
    its('UMASK') { should include(login_defs_umask) }
    its('PASS_MAX_DAYS') { should eq login_defs_passmaxdays }
    its('PASS_MIN_DAYS') { should eq login_defs_passmindays }
    its('PASS_WARN_AGE') { should eq login_defs_passwarnage }
    its('LOGIN_RETRIES') { should eq '5' }
    its('LOGIN_TIMEOUT') { should eq '60' }
    its('UID_MIN') { should eq '1000' }
    its('GID_MIN') { should eq '1000' }
  end
end

control 'os-05b' do
  impact 1.0
  title 'Check login.defs - RedHat specific'
  desc 'Check owner and permissions for login.defs. Also check the configured PATH variable and umask in login.
defs'
  describe file('/etc/login.defs') do
    it { should_not be_writable }
  end
  describe login_defs do
    its('SYS_UID_MIN') { should eq '201' }
    its('SYS_UID_MAX') { should eq '999' }
    its('SYS_GID_MIN') { should eq '201' }
    its('SYS_GID_MAX') { should eq '999' }
  end
  only_if { os.redhat? }
end

control 'os-06' do
  impact 1.0
  title 'Check for SUID/ SGID blacklist'
  desc 'Find blacklisted SUID and SGID files to ensure that no rogue SUID and SGID files have been introduced
into the system'

  describe suid_check(blacklist) do
    its('diff') { should be_empty }
  end
end

control 'os-07' do
  impact 1.0
  title 'Unique uid and gid'
  desc 'Check for unique uids gids'
  describe passwd do
    its('uids') { should_not contain_duplicates }
  end
  describe etc_group do
    its('gids') { should_not contain_duplicates }
  end
end

control 'os-08' do
```

```
  impact 1.0
  title 'Entropy'
  desc 'Check system has enough entropy - greater than 1000'
  describe file('/proc/sys/kernel/random/entropy_avail').content.to_i do
    it { should >= 1000 }
  end
end

control 'os-09' do
  impact 1.0
  title 'Check for .rhosts and .netrc file'
  desc 'Find .rhosts and .netrc files - CIS Benchmark 9.2.9-10'
  output = command('find / -maxdepth 3 \( -iname .rhosts -o -iname .netrc \) -print 2>/dev/null | grep -v
\'^find:\'')
  out = output.stdout.split(/\r?\n/)
  describe out do
    it { should be_empty }
  end
end

control 'os-10' do
  impact 1.0
  title 'CIS: Disable unused filesystems'
  desc '1.1.1 Ensure mounting of cramfs, freevxfs, jffs2, hfs, hfsplus, squashfs, udf, FAT'
  only_if { !container_execution }
  efi_dir = inspec.file('/sys/firmware/efi')
  describe file('/etc/modprobe.d/dev-sec.conf') do
    its(:content) { should match 'install cramfs /bin/true' }
    its(:content) { should match 'install freevxfs /bin/true' }
    its(:content) { should match 'install jffs2 /bin/true' }
    its(:content) { should match 'install hfs /bin/true' }
    its(:content) { should match 'install hfsplus /bin/true' }
    its(:content) { should match 'install squashfs /bin/true' }
    its(:content) { should match 'install udf /bin/true' }
    # if efi is active, do not disable vfat. otherwise the system
    # won't boot anymore
    unless efi_dir.exist?
      its(:content) { should match 'install vfat /bin/true' }
    end
  end
end

control 'os-11' do
  impact 1.0
  title 'Protect log-directory'
  desc 'The log-directory /var/log should belong to root'
  describe file('/var/log') do
    it { should be_directory }
    it { should be_owned_by 'root' }
    its(:group) { should match(/^root|syslog$/) }
  end
end
```

To run InSpec against a remote host and produce a JUnit XML report that can be submitted to Xray, we may use the following command.

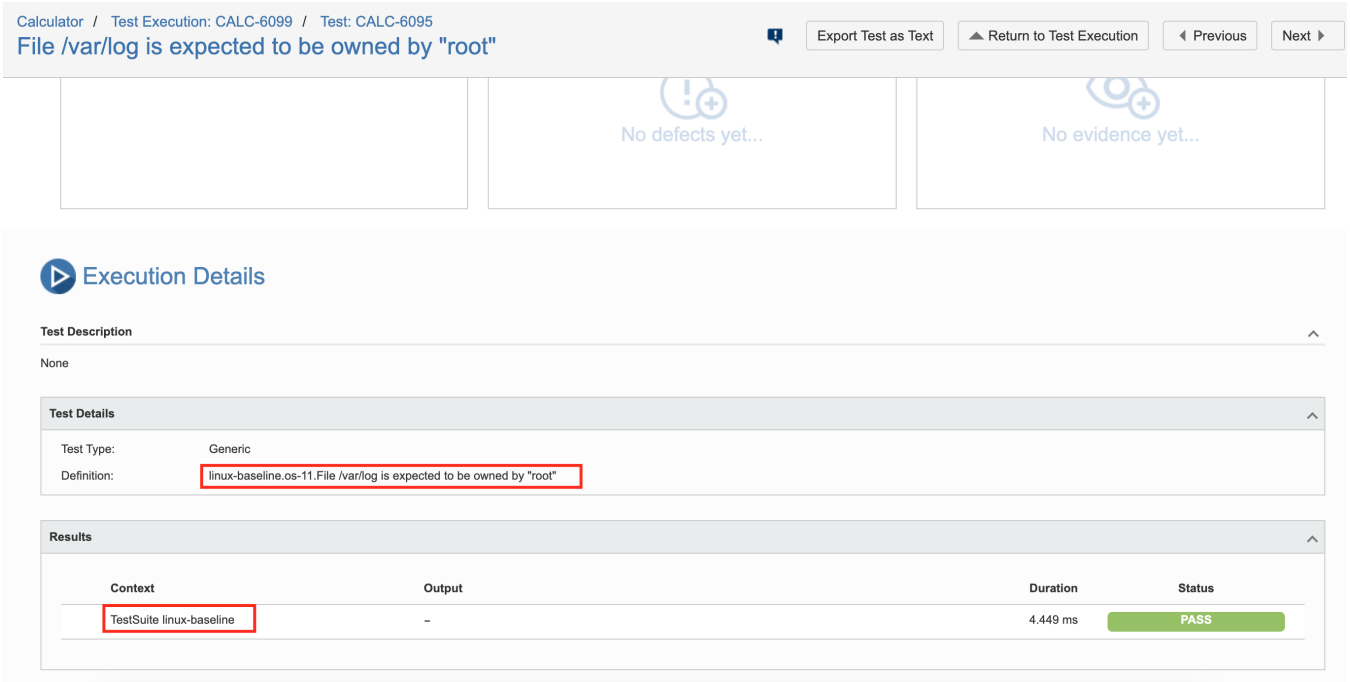
```
inspec supermarket exec dev-sec/linux-baseline -t ssh://jira:jira@192.168.56.102 --reporter junit:junit.xml
```

After successfully running the Test Case and generating the JUnit XML report (e.g., junit.xml), it can be imported to Xray (either by the REST API or by using one of the available CI addons or even through **Import Execution Results** action within the Test Execution).



Each check is mapped to a Generic Test in Jira, and the **Generic Test Definition** field contains the name of the profile followed by the name of the control and some text derived from the assertion being done in the *describe* and inner *it* block.

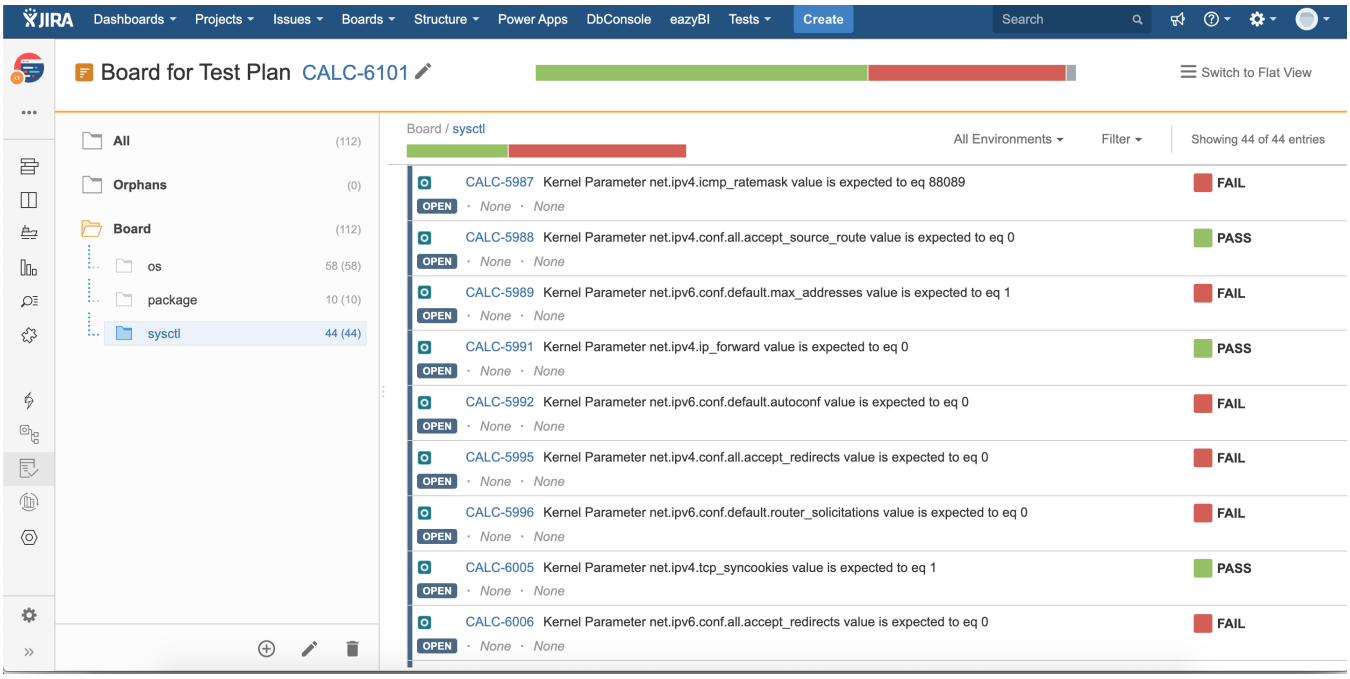
The Context section contains information about the profile name.



Tips

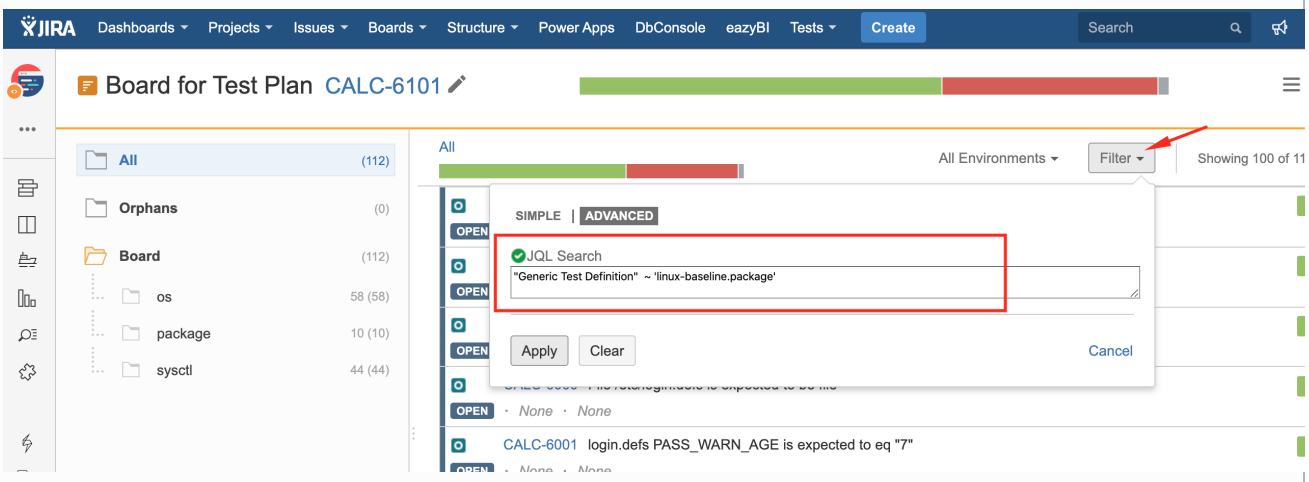
It may be useful to have a Test Plan with all these checks organized hierarchically, so you can track compliance at multiple levels.

1. create an empty Test Plan
2. import the results, so you'll have a Test Execution
3. go to the Test Plan issue screen and add the Test Execution
4. organize the Tests within the Test Plan Board



Learn more

To organize the Tests in the Board of the Test Plan, you may use filters based on the Generic Test Definition field, filtering by the name of the profile plus the name of the control.



Then you can easily move the Tests that matter to the folder that you want.

References

- https://www.inspec.io/
- https://www.inspec.io/docs/
- https://learn.chef.io/modules/try-inspec#/
- http://www.anniehedgie.com/inspec/
- https://medium.com/gsktech/always-on-compliance-with-inspec-e3015a229be4
- https://github.com/dev-sec/linux-baseline