Compliance as Code and at Scale with RHEL

Andrejs Galzons

What we are not going to be talking about?

- Auditing tools themselves – audtictl
- Specific standards – such as HIPPAA, PCI-DSS
- Specific audits or consultants, or anything

What are we actually talking about?

- Problems with audits
- Repeatable work and results
- Additional evidence data
- Reinventing the wheel

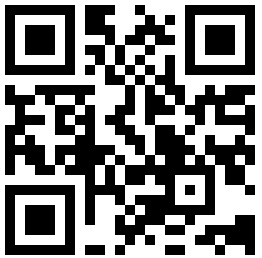
First off components and pieces

SCAP

- Security Content Automation Protocol
- Set of Specification Standard for performing automated security configuration review
- Widely used in the USA Federal governement

OpenSCAP

- Open source framework for implementing configuration compliance verification
- Available on the major Linux distributions – Red Hat Enterprise Linux, Fedora, etc.
- Can perform code execution*



Compliance as Code

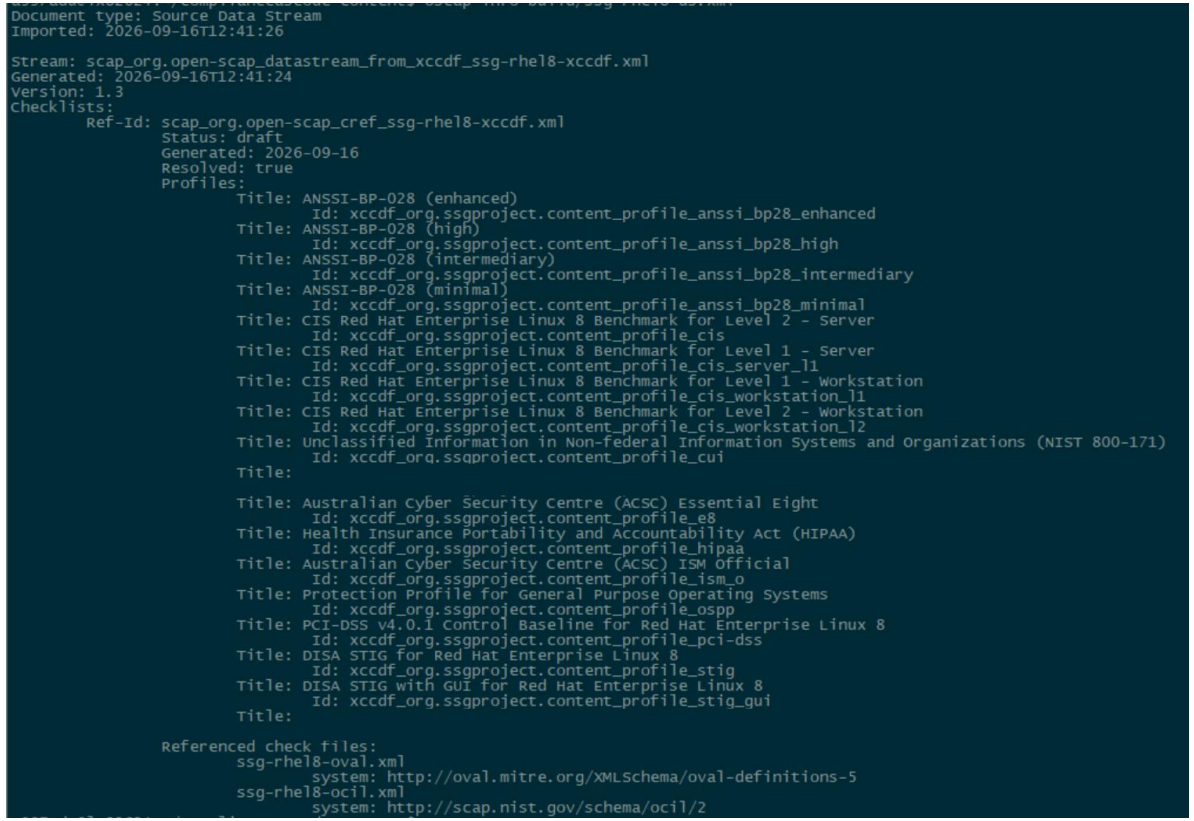
- Support various Linux distributions, as well as products — Firefox,…
- Build, view and modify the benchmarks
- Open Source project used by various distributions, made specifically for humans by humans



Enough selling, let’s see something practical

Checking available benchmarks and metadata

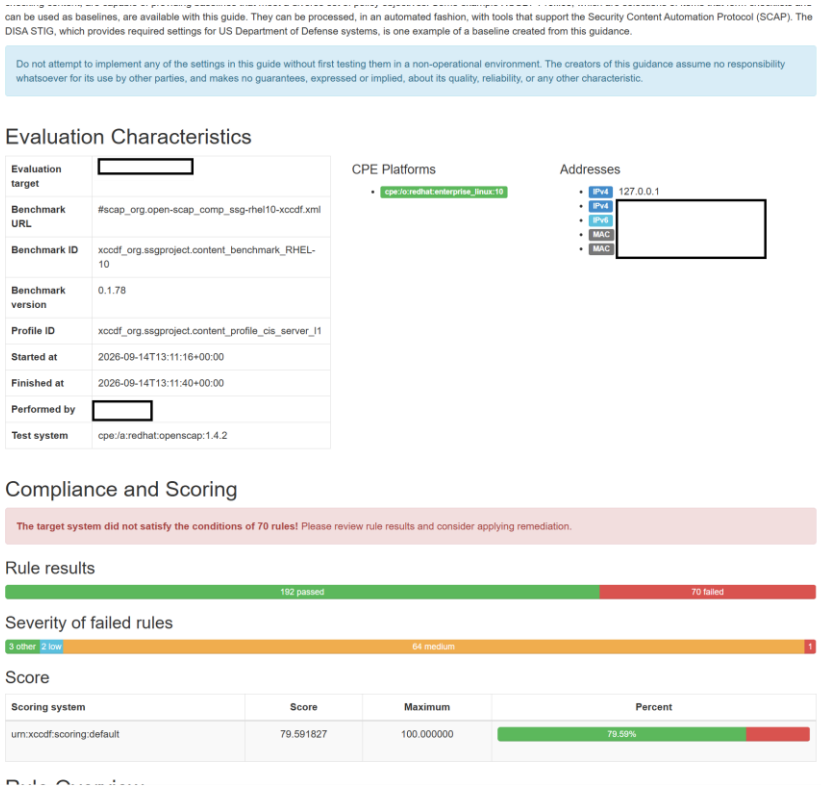
```
oscap info <path to the datastream file>
```



Evaluating system based on a profile/benchmark

oscap xccdf eval --profile <benchmark/profile drom datastream file> --report <path to report to create> <path to datastream file>

What do you get out of it?



can be used as baselines, are available with this guide. They can be processed, in an automated fashion, with tools that support the Security Content Automation Protocol (SCAP). The DISA STIG, which provides required settings for US Department of Defense systems, is one example of a baseline created from this guidance.

Do not attempt to implement any of the settings in this guide without first testing them in a non-operational environment. The creators of this guidance assume no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic.

Evaluation Characteristics

Evaluation target	[redacted]
Benchmark URL	#scap_org.open-scap_comp_ssg-rhel10-xccdf.xml
Benchmark ID	xccdf_org.ssgproject.content_benchmark_RHEL-10
Benchmark version	0.1.78
Profile ID	xccdf_org.ssgproject.content_profile_cis_server_l1
Started at	2026-09-14T13:11:16+00:00
Finished at	2026-09-14T13:11:40+00:00
Performed by	[redacted]
Test system	cpe:/a:redhat:openscap:1.4.2

CPE Platforms

- cpe:/o:redhat:enterprise_linux:10

Addresses

- IPv4 127.0.0.1
- IPv4
- IPv6
- MAC
- MAC

Compliance and Scoring

The target system did not satisfy the conditions of 70 rules! Please review rule results and consider applying remediation.

Rule results

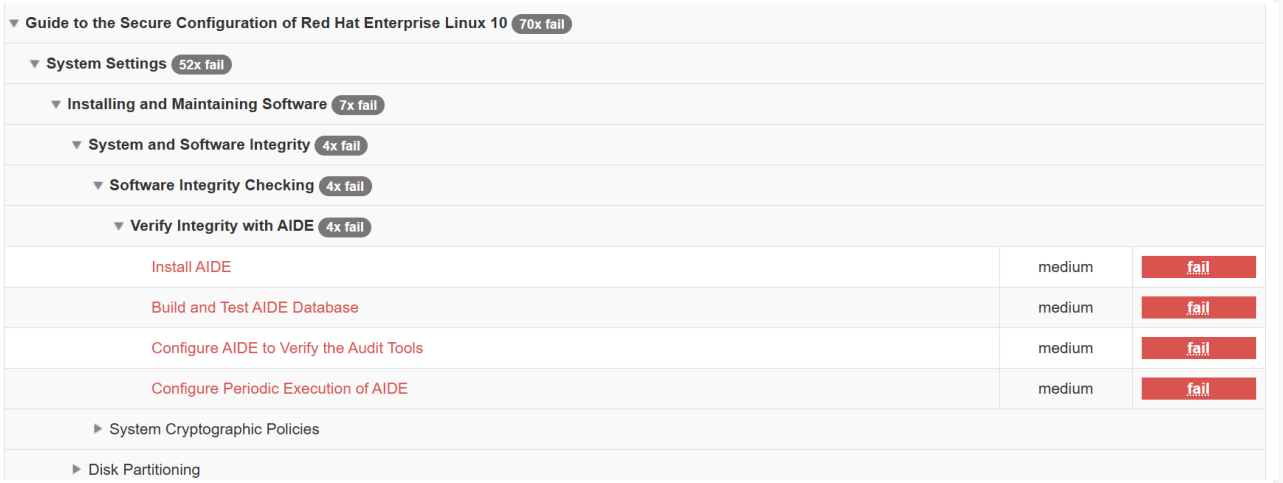
192 passed | 70 failed

Severity of failed rules

3 other | 2 low | 64 medium | 1

Score

Scoring system	Score	Maximum	Percent
urn:xccdf:scoring:default	79.591827	100.000000	79.59%

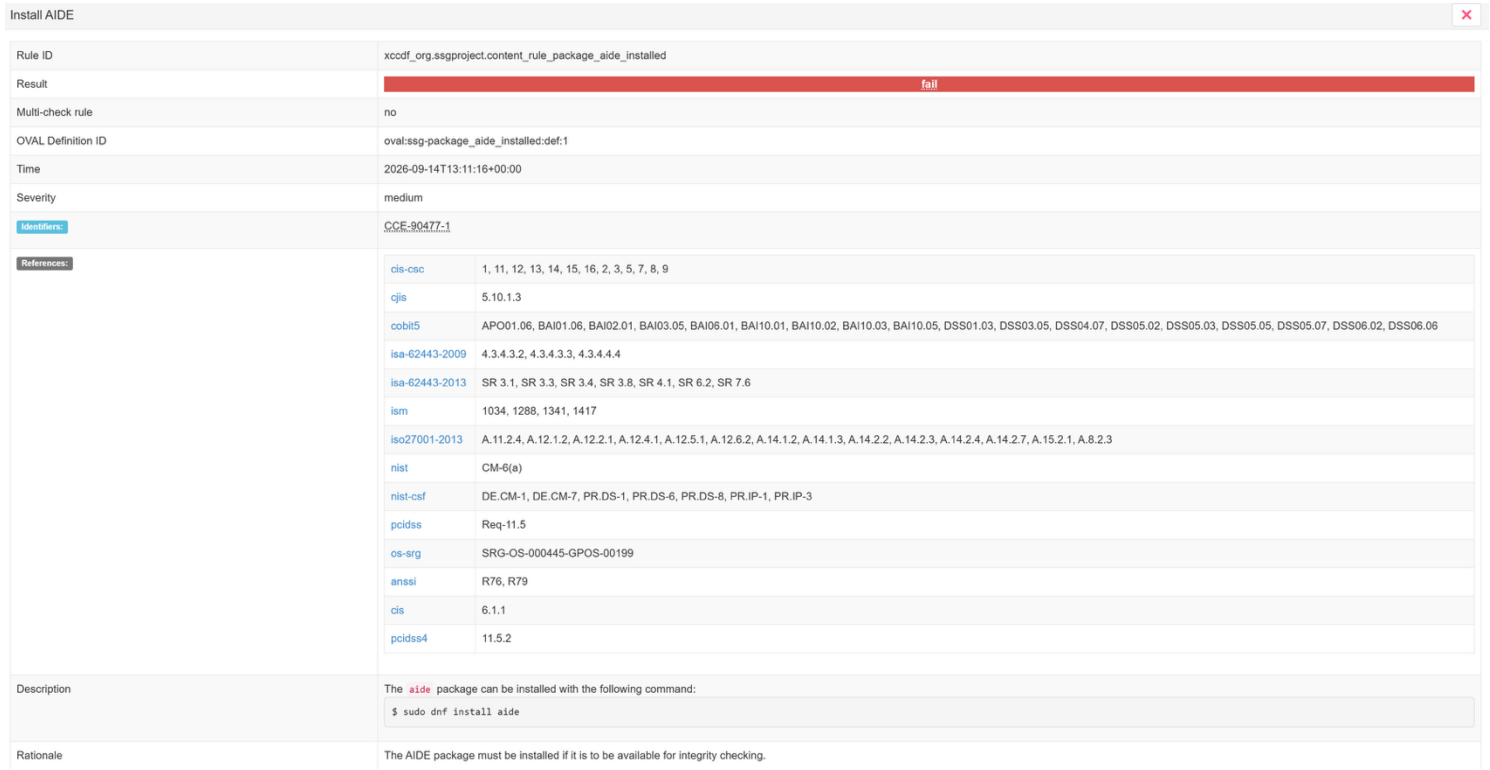


- Guide to the Secure Configuration of Red Hat Enterprise Linux 10 — 70x fail
  - System Settings — 52x fail
    - Installing and Maintaining Software — 7x fail
      - System and Software Integrity — 4x fail
        - Software Integrity Checking — 4x fail
          - Verify Integrity with AIDE — 4x fail

Install AIDE	medium	fail
Build and Test AIDE Database	medium	fail
Configure AIDE to Verify the Audit Tools	medium	fail
Configure Periodic Execution of AIDE	medium	fail

  - ▶ System Cryptographic Policies
  - ▶ Disk Partitioning

Deeper look



Install AIDE

Rule ID	xccdf_org.ssgproject.content_rule_package_aide_installed
Result	fail
Multi-check rule	no
OVAL Definition ID	oval:ssg-package_aide_installed:def:1
Time	2026-09-14T13:11:16+00:00
Severity	medium
Identifiers:	CCE-90477-1
References:	

cis-csc	1, 11, 12, 13, 14, 15, 16, 2, 3, 5, 7, 8, 9
cjis	5.10.1.3
cobit5	APO01.06, BAI01.06, BAI02.01, BAI03.05, BAI06.01, BAI10.01, BAI10.02, BAI10.03, BAI10.05, DSS01.03, DSS03.05, DSS04.07, DSS05.02, DSS05.03, DSS05.05, DSS05.07, DSS06.02, DSS06.06
isa-62443-2009	4.3.4.3.2, 4.3.4.3.3, 4.3.4.4.4
isa-62443-2013	SR 3.1, SR 3.3, SR 3.4, SR 3.8, SR 4.1, SR 6.2, SR 7.6
ism	1034, 1288, 1341, 1417
iso27001-2013	A.11.2.4, A.12.1.2, A.12.2.1, A.12.4.1, A.12.5.1, A.12.6.2, A.14.1.2, A.14.1.3, A.14.2.2, A.14.2.3, A.14.2.4, A.14.2.7, A.15.2.1, A.8.2.3
nist	CM-6(a)
nist-csf	DE.CM-1, DE.CM-7, PR.DS-1, PR.DS-6, PR.DS-8, PR.IP-1, PR.IP-3
pcidss	Req-11.5
os-srg	SRG-OS-000445-GPOS-00199
anssi	R76, R79
cis	6.1.1
pcidss4	11.5.2

Description	The `aide` package can be installed with the following command: `$ sudo dnf install aide`
Rationale	The AIDE package must be installed if it is to be available for integrity checking.

Customization & Creation

Component Creation

- Way to group and organize related controls
- Usually used one component – per system component

components > ! shell-help.yml

```yaml
name: shell-help
packages:
- command-line-assistant
- bash-completion
rules:
- command-line-assistant
- bash-completion
```

components > ! command-line-assistant.yml

```yaml
name: command-line-assistant
packages:
- command-line-assistant
rules:
- command-line-assistant
```

Control Creation

```
linux_os > guide > system > logging > journald > service_systemd-journald_enabled > ! rule.yml
34
35  template:
36      name: service_enabled
37      vars:
38          servicename: systemd-journald
39          packagename: systemd
40
```

- Meat to the actual bone

```
linux_os > guide > system > software > command-line-assistant > ! rule.yml
1   documentation_complete: true
2   title: 'Ensure Command Line Assistant is installed'
3   description: 'Command Line Assistant should be installed and available for our developer friends, who are not dealing with Linux everyday'
4   rationale: |
5     If command line assistant is not present, it can lead to awkward questions, or too much coffee breaks explaining why web servers shouldn't run as root
6   severity: medium
7
8   # This block replaces the need for manual OVAL/Bash/Ansible files
9   template:
10      name: package_installed
11      vars:
12          pkgname: command-line-assistant
13
14  # This determines which products will include the rule
15  platforms:
16    - machine
17
```

Enabling the actual check/control

- Why is it still not executing???

products > rhel10 > profiles > $ cis_server_l1.profile

```
...
---
documentation_complete: true

metadata:
    version: 1.0.0
    SMEs:
        - mab879
        - ggbecker

reference: https://www.cisecurity.org/benchmark/red_hat_linux/

title: 'CIS Red Hat Enterprise Linux 10 Benchmark for Level 1 - Server'

description: |-
    This profile defines a baseline that aligns to the "Level 1 - Server"
    configuration from the Center for Internet Security® Red Hat Enterprise
    Linux 10 Benchmark™, v1.0.0, released 2025-08-27.

    This profile includes Center for Internet Security®
    Red Hat Enterprise Linux 10 CIS Benchmarks™ content.

selections:
    - cis_rhel10:all:l1_server
    - var_authselect_profile=local
    - command-line-assistant
      You, 59 seconds ago • Uncommitted changes
```

Pfffffft, thats only the easy part

Crafting a more complex rule

linux_os > guide > system > software > bingo-bang-bongo > ! rule.yml

```yaml
documentation_complete: true
title: 'Ensure bingo-bang-bongo is scheduled'

description: |-
    The root crontab must contain an active (non-commented) entry for the
    `bingo-bang-bongo` to ensure that manbo is performed on every system

rationale: |-
    All work and no play, makes Andrejs a dull boy

severity: medium

platforms:
  - machine
ocil_clause: 'the bingo-bang-bongo is not scheduled or is commented out'

ocil: |-
    To verify the root crontab entry, run:
    $ sudo crontab -l | grep bingo-bang-bongo
```

Not using templates

linux_os > guide > system > software > bingo-bang-bongo > oval > shared.xml

```
<def-group>
  <definition class="compliance" id="bingo_bango_bongo_scheduled" version="1">
    <metadata>
      <title>Verify bingo-bango-bongo in root crontab</title>
      <affected family="unix">
        <platform>multi_platform_all</platform>
      </affected>
      <description>The root crontab should have an active bingo-bango-bongo entry.</description>
    </metadata>
    <criteria>
      <criterion comment="active bingo-bango-bongo entry found" test_ref="test_cron_bingo-bang-bongo_server" />
    </criteria>
  </definition>

  <ind:textfilecontent54_test check="all" check_existence="at_least_one_exists" id="test_bingo-bango-bongo" version="1" comment="bingo-bango-bongo is sched
    <ind:object object_ref="obj_cron_bingo_bango_bongo" />
  </ind:textfilecontent54_test>

  <ind:textfilecontent54_object id="obj_cron_bingo_bango_bongo" version="1">
    <ind:filepath>/var/spool/cron/root</ind:filepath>
    # Let's check if root crontab contains non-commented out entry of the bingo-bango-bongo
    <ind:pattern operation="pattern match">^[ \t]*[^#\n].*\/opt\/fun\/bingo-bango-bongo.*$</ind:pattern>
    <ind:instance datatype="int">1</ind:instance>
  </ind:textfilecontent54_object>
</def-group>
```

I don’t want to wrap my head arround xml files

Reuse, Reduce and Recycle

Templates

AI (Artifical Intelligence, not Apple Intelligence)

What’s next after the custom benchmark?

Integrations:

- Satellite
- Foreman
- Ansible
- Ansible Automation Platform

Thank you for the attention! Questions?