

SCAP Security Guide Questions / Answers



...

Contributor WorkShop Volume #2

Ján Lieskovský

January 2016

Agenda

- Introductory Notes
- Source Code / Repository Notes (Moved to Appendix for self-study)
- SCAP Security Guide “shorthand” Format
- Contributing New Rule
- Advanced Topics / Selected Use Cases



Agenda (more detailed)



- Introductory Notes
 - Purpose of the SCAP Security Guide (SSG) Project
 - SCAP Protocol Standards
- Source Code / Repository Notes (Moved to Appendix for self-study)
 - Where To Obtain the Source Code
 - Developer Workflow
 - Pull Request Reviewer Workflow
 - Repository Structure
 - Currently Supported SSG Products
- SCAP Security Guide “shorthand” Format

Agenda (more detailed)



- Contributing New Rule
 - Contributing XCCDF Part
 - Contributing OVAL Part
 - Contributing Remediation Part
- Advanced Topics / Selected Use Cases
 - Purpose of different locations of various OVAL checks
 - How to contribute new rule from scratch
 - How to contribute new rule from **templates**
 - Using SSG **templates** to contribute new OVAL checks
 - How to port existing rules, profiles, benchmarks across different products

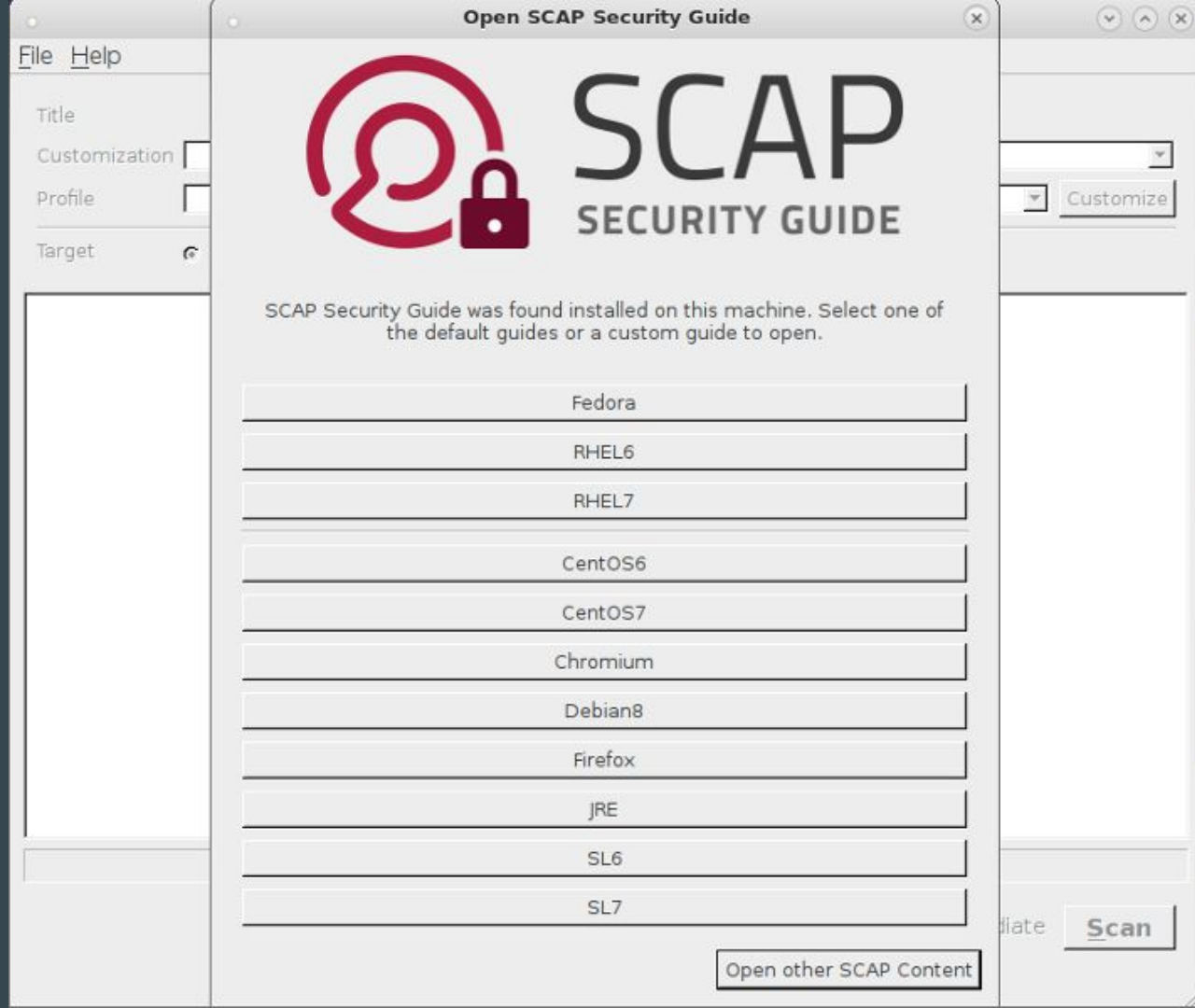


Introductory Notes

Introductory Notes

Benchmarks
available in
SSG-0.1.27:

Note:
SSG-0.1.28 will bring
benchmark for
RHEL Openstack
Platform 7 yet.



Introductory Notes

Currently
available
RHEL-6
profiles:

File Help

Title **Guide to the Secure Configuration of Red Hat Enterprise Linux 6**

Customization (no customization)

Profile Example Server Profile Customize

Target

- C2S for Red Hat Enterprise Linux 6
- Example Server Profile
- CSCF RHEL6 MLS Core Baseline
- Common Profile for General-Purpose Systems**

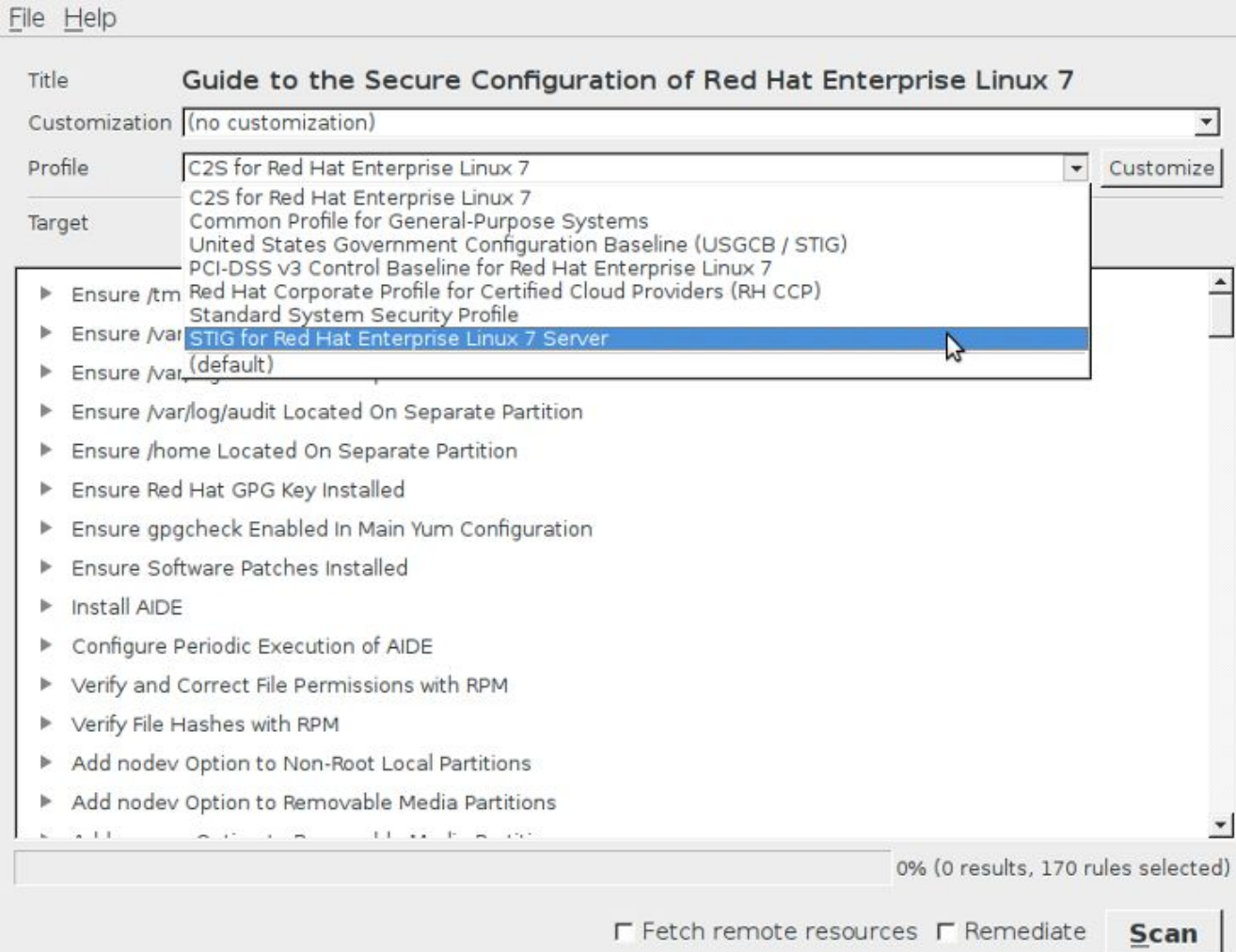
- ▶ Ensure /tmp CNSSI 1253 Low/Low/Low
- ▶ Ensure /var PCI-DSS v3 Control Baseline for Red Hat Enterprise Linux 6
- ▶ Ensure /var Red Hat Corporate Profile for Certified Cloud Providers (RH CCP)
- ▶ Ensure /var Server Baseline
- ▶ Ensure /var Upstream STIG for Red Hat Enterprise Linux 6 Server
- ▶ Ensure /var United States Government Configuration Baseline (USGCB)
- (default)
- ▶ Ensure /home Located On Separate Partition
- ▶ Ensure Red Hat GPG Key Installed
- ▶ Ensure gpgcheck Enabled In Main Yum Configuration
- ▶ Ensure gpgcheck Enabled For All Yum Package Repositories
- ▶ Install AIDE
- ▶ Disable Prelinking
- ▶ Build and Test AIDE Database
- ▶ Verify and Correct File Permissions with RPM
- ▶ Verify File Hashes with RPM
- ▶ Add nodev Option to Removable Media Partitions

0% (0 results, 313 rules selected)

☐ Fetch remote resources ☐ Remediate Scan

Introductory Notes

Currently
available
RHEL-7
profiles:



Introductory Notes - The two `scap-security-guide` RPM packages:

```
File Edit View Search Terminal Help
[iankko@localhost ~]$ rpm -qi scap-security-guide
Name       : scap-security-guide
Version    : 0.1.27
Release    : 1.fc22
Architecture: noarch
Install Date: Sat 09 Jan 2016 02:37:01 PM CET
Group      : Applications/System
Size       : 35229783
License    : Public Domain
Signature  : RSA/SHA256, Sat 12 Dec 2015 07:10:26 PM CET, Key ID 11adc0948e1431d5
Source RPM : scap-security-guide-0.1.27-1.fc22.src.rpm
Build Date  : Fri 11 Dec 2015 09:49:57 PM CET
Build Host  : buildvm-08-nfs.phx2.fedoraproject.org
Relocations : (not relocatable)
Packager    : Fedora Project
Vendor      : Fedora Project
URL         : https://fedorahosted.org/scap-security-guide/
Summary     : Security guidance and baselines in SCAP formats
Description :
The scap-security-guide project provides a guide for configuration of the
system from the final system's security point of view. The guidance is specified
in the Security Content Automation Protocol (SCAP) format and constitutes
a catalog of practical hardening advice, linked to government requirements
where applicable. The project bridges the gap between generalized policy
requirements and specific implementation guidelines. The Fedora system
administrator can use the oscap CLI tool from openscap-scanner package, or the
scap-workbench GUI tool from scap-workbench package to verify that the system
conforms to provided guideline. Refer to scap-security-guide(8) manual page for
further information.
[iankko@localhost ~]$
```



Introductory Notes - The two `scap-security-guide` RPM packages:

```
File Edit View Search Terminal Help
[iankko@localhost ~]$ rpm -qi scap-security-guide-doc
Name       : scap-security-guide-doc
Version    : 0.1.27
Release    : 1.fc22
Architecture: noarch
Install Date: Sat 09 Jan 2016 02:38:35 PM CET
Group      : System Environment/Base
Size       : 42791747
License    : Public Domain
Signature  : RSA/SHA256, Sat 12 Dec 2015 07:08:50 PM CET, Key ID 11adc0948e1431d5
Source RPM : scap-security-guide-0.1.27-1.fc22.src.rpm
Build Date : Fri 11 Dec 2015 09:49:57 PM CET
Build Host : buildvm-08-nfs.phx2.fedoraproject.org
Relocations : (not relocatable)
Packager   : Fedora Project
Vendor     : Fedora Project
URL        : https://fedorahosted.org/scap-security-guide/
Summary    : HTML formatted documents containing security guides generated from
             XCCDF benchmarks.
Description :
The scap-security-guide-doc package contains HTML formatted documents containing
hardening guidances that have been generated from XCCDF benchmarks
present in scap-security-guide package.
[iankko@localhost ~]$
```



Introductory Notes - scap-security-guide RPM content:



File Edit View Search Terminal Help

```
[iankko@localhost ~]$ ls -d $(rpm -ql scap-security-guide)
/usr/share/doc/scap-security-guide/Firefox/DISCLAIMER
/usr/share/doc/scap-security-guide/LICENSE
/usr/share/doc/scap-security-guide/README.md
/usr/share/man/en/man8/scap-security-guide.8.gz
/usr/share/scap-security-guide
/usr/share/scap-security-guide/kickstart
/usr/share/scap-security-guide/kickstart/ssg-rhel6-pci-dss-with-gui-ks.cfg
/usr/share/scap-security-guide/kickstart/ssg-rhel6-stig-ks.cfg
/usr/share/scap-security-guide/kickstart/ssg-rhel6-usgcb-server-with-gui-ks.cfg
/usr/share/scap-security-guide/kickstart/ssg-rhel7-ospp-ks.cfg
/usr/share/scap-security-guide/kickstart/ssg-rhel7-pci-dss-server-with-gui-oaa-ks.cfg
/usr/share/scap-security-guide/remediation_functions
/usr/share/xml/scap
/usr/share/xml/scap/ssg
/usr/share/xml/scap/ssg/content
/usr/share/xml/scap/ssg/content/ssg-centos6-ds.xml
/usr/share/xml/scap/ssg/content/ssg-centos6-xccdf.xml
/usr/share/xml/scap/ssg/content/ssg-centos7-ds.xml
/usr/share/xml/scap/ssg/content/ssg-centos7-xccdf.xml
/usr/share/xml/scap/ssg/content/ssg-chromium-cpe-dictionary.xml
/usr/share/xml/scap/ssg/content/ssg-chromium-cpe-oval.xml
/usr/share/xml/scap/ssg/content/ssg-chromium-ds.xml
/usr/share/xml/scap/ssg/content/ssg-chromium-oval.xml
/usr/share/xml/scap/ssg/content/ssg-chromium-xccdf.xml
/usr/share/xml/scap/ssg/content/ssg-debian8-cpe-dictionary.xml
/usr/share/xml/scap/ssg/content/ssg-debian8-cpe-oval.xml
/usr/share/xml/scap/ssg/content/ssg-debian8-ds.xml
/usr/share/xml/scap/ssg/content/ssg-debian8-oval.xml
/usr/share/xml/scap/ssg/content/ssg-debian8-xccdf.xml
/usr/share/xml/scap/ssg/content/ssg-fedora-ds.xml
/usr/share/xml/scap/ssg/content/ssg-fedora-oval.xml
/usr/share/xml/scap/ssg/content/ssg-fedora-xccdf.xml
/usr/share/xml/scap/ssg/content/ssg-firefox-cpe-dictionary.xml
/usr/share/xml/scap/ssg/content/ssg-firefox-cpe-oval.xml
/usr/share/xml/scap/ssg/content/ssg-firefox-ds.xml
/usr/share/xml/scap/ssg/content/ssg-firefox-oval.xml
/usr/share/xml/scap/ssg/content/ssg-firefox-xccdf.xml
/usr/share/xml/scap/ssg/content/ssg-jre-cpe-dictionary.xml
/usr/share/xml/scap/ssg/content/ssg-jre-cpe-oval.xml
/usr/share/xml/scap/ssg/content/ssg-jre-ds.xml
/usr/share/xml/scap/ssg/content/ssg-jre-oval.xml
/usr/share/xml/scap/ssg/content/ssg-jre-xccdf.xml
/usr/share/xml/scap/ssg/content/ssg-rhel6-cpe-dictionary.xml
/usr/share/xml/scap/ssg/content/ssg-rhel6-cpe-oval.xml
/usr/share/xml/scap/ssg/content/ssg-rhel6-ds.xml
/usr/share/xml/scap/ssg/content/ssg-rhel6-oval.xml
/usr/share/xml/scap/ssg/content/ssg-rhel6-xccdf.xml
/usr/share/xml/scap/ssg/content/ssg-rhel7-cpe-dictionary.xml
/usr/share/xml/scap/ssg/content/ssg-rhel7-cpe-oval.xml
/usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
/usr/share/xml/scap/ssg/content/ssg-rhel7-oval.xml
/usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml
/usr/share/xml/scap/ssg/content/ssg-sl6-ds.xml
/usr/share/xml/scap/ssg/content/ssg-sl6-xccdf.xml
/usr/share/xml/scap/ssg/content/ssg-sl7-ds.xml
/usr/share/xml/scap/ssg/content/ssg-sl7-xccdf.xml
```

Introductory Notes - `scap-security-guide-doc` RPM content: (first 20 lines)



File Edit View Search Terminal Help

```
[iankko@localhost ~]$ rpm -ql scap-security-guide-doc | head -20
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-C2S.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-CS2.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-CSCF-RHEL6-MLS.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-common.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-default.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-index.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-nist-cl-il-al.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-pci-dss.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-rht-ccp.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-server.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-stig-rhel6-server-upstream.html
/usr/share/doc/scap-security-guide/guides/ssg-centos6-guide-usgcb-rhel6-server.html
/usr/share/doc/scap-security-guide/guides/ssg-centos7-guide-C2S.html
/usr/share/doc/scap-security-guide/guides/ssg-centos7-guide-common.html
/usr/share/doc/scap-security-guide/guides/ssg-centos7-guide-default.html
/usr/share/doc/scap-security-guide/guides/ssg-centos7-guide-index.html
/usr/share/doc/scap-security-guide/guides/ssg-centos7-guide-ospp-rhel7-server.html
/usr/share/doc/scap-security-guide/guides/ssg-centos7-guide-pci-dss.html
/usr/share/doc/scap-security-guide/guides/ssg-centos7-guide-rht-ccp.html
/usr/share/doc/scap-security-guide/guides/ssg-centos7-guide-standard.html
[iankko@localhost ~]$
```

Introductory Notes - Purpose of the SCAP Security Guide project:



SCAP Security Guide:

- delivers security guidance, baselines, and associated validation mechanisms using the SCAP protocol

Introductory Notes - Purpose of the SCAP Security Guide project (continued):



SCAP Security Guide:

- links security hardening advice for (Red Hat) products with official requirements (**NIST SP 800-53**, **USGCB ..**)

Introductory Notes - SCAP Security Guide project “links security hardening advice ... with official requirements”



```
File Edit View Search Terminal Help
[iankko@localhost 6]$ pwd
/tmp/github/scap-security-guide/RHEL/6
[iankko@localhost 6]$ make clean >& /dev/null; echo $?
0
[iankko@localhost 6]$ make tables >& /dev/null; echo $?
0
[iankko@localhost 6]$ firefox output/table-rhel6-nistrefs-common.html
[iankko@localhost 6]$
```

Common Profile for General-Purpose Systems

CCE ID	Rule Title	Description	Rationale	Variable Setting	NIST 800-53 Mapping
CCE-26435-8	Ensure /tmp Located On Separate Partition	The /tmp directory is a world-writable directory used for temporary file storage. Ensure it has its own partition or logical volume at installation time, or migrate it using LVM.	The /tmp partition is used as temporary storage by many programs. Placing /tmp in its own partition enables the setting of more restrictive mount options, which can help protect programs which use it.		SC-32
CCE-26639-5	Ensure /var Located On Separate Partition	The /var directory is used by daemons and other system services to store frequently-changing data. Ensure that /var has its own partition or logical volume at installation time, or migrate it using LVM.	Ensuring that /var is mounted on its own partition enables the setting of more restrictive mount options. This helps protect system services such as daemons or other programs which use it. It is not uncommon for the /var directory to contain world-writable directories installed by other software packages.		SC-32

Introductory Notes - Purpose of the SCAP Security Guide project (continued):



SCAP Security Guide:

- bridges the gap between official security requirements and specific security hardening implementation details

Introductory Notes - SCAP Security Guide project

“bridges the gap between official security requirements and specific security hardening implementation details”



For example [Requirement 8.2.3](#) of the [PCI DSS v3 standard](#) requires the following:

8.2.3 Passwords/phrases must meet the following:

- Require a minimum length of at least seven characters.
- Contain both numeric and alphabetic characters.

Alternatively, the passwords/phrases must have complexity and strength at least equivalent to the parameters specified above.

In SSG's [RHEL/7](#) product this requirement maps to the following rules:

- [no_empty_passwords](#)
- [accounts_password_pam_dcredit](#)
- [accounts_password_pam_minlen](#)
- [accounts_password_pam_ucredit](#)
- [accounts_password_pam_lcredit](#)

Introductory Notes - SCAP Security Guide project “bridges the gap between official security requirements and specific security hardening implementation details”



In SSG’s **RHEL/7** product this requirement maps to the following rules:

```
File Edit View Search Terminal Help
[iankko@localhost 7]$ pwd
/tmp/github/scap-security-guide/RHEL/7
[iankko@localhost 7]$ make clean >& /dev/null; echo $?
0
[iankko@localhost 7]$ make tables >& /dev/null; echo $?
0
[iankko@localhost 7]$ firefox output/table-rhel7-pcidss.html
[iankko@localhost 7]$
```

Introductory Notes - SCAP Security Guide project “bridges the gap between official security requirements and specific security hardening implementation details”



In SSG’s [RHEL/7](#) product this requirement maps to the following rules (preview):

Req-8.2.3	Set Password Strength Minimum Digit Characters	The pam_pwquality module’s dcredit parameter controls requirements for usage of digits in a password. When set to a negative number, any password will be required to contain that many digits. When set to a positive number, pam_pwquality will grant +1 additional length credit for each digit. Modify the dcredit setting in /etc/security/pwquality.conf to require the use of a digit in passwords.	Requiring digits makes password guessing attacks more difficult by ensuring a larger search space.
Req-8.2.3	Set Password Minimum Length	The pam_pwquality module’s minlen parameter controls requirements for minimum characters required in a password. Add minlen= after pam_pwquality to set minimum password length requirements.	Password length is one factor of several that helps to determine strength and how long it takes to crack a password. Use of more characters in a password helps to exponentially increase the time and/or resources required to compromise the password.
Req-8.2.3	Set Password Strength Minimum Uppercase Characters	The pam_pwquality module’s ucredit= parameter controls requirements for usage of uppercase letters in a password. When set to a negative number, any password will be required to contain that many uppercase characters. When set to a positive number, pam_pwquality will grant +1 additional length credit for each uppercase character. Modify the ucredit setting in /etc/security/pwquality.conf to require the use of an uppercase character in passwords.	Requiring a minimum number of uppercase characters makes password guessing attacks more difficult by ensuring a larger search space.
Req-8.2.3	Set Password Strength Minimum Lowercase Characters	The pam_pwquality module’s lcredit parameter controls requirements for usage of lowercase letters in a password. When set to a negative number, any password will be required to contain that many lowercase characters. When set to a positive number, pam_pwquality will grant +1 additional length credit for each lowercase character. Modify the lcredit setting in /etc/security/pwquality.conf to require the use of a lowercase character in passwords.	Requiring a minimum number of lowercase characters makes password guessing attacks more difficult by ensuring a larger search space.



The Security Content Automation Protocol (SCAP):

Suite of specifications to:

- standardize both **format** and **nomenclature**,
- used to express both **software flaw** and **security configuration information**,
- in a way easily comprehensible both to **machines** and **humans**



The Security Content Automation Protocol (SCAP):

- Latest version 1.2 (defined by [NIST SP 800-126 Rev. 2](#))
- Upcoming version 1.3 in progress
 - [Tentative plans](#) to:
 - add support for [OVAL-5.11.1](#), [CVSS v3.0](#)
 - Final NIST SP 800-126 Rev.3 needs to clarify on support for e.g. SCAP 1.0, SCAP 1.1, CVSS v2.0 and many more

SCAP 1.2 suite of specifications:



Languages	<u>XCCDF v1.2</u>	<u>OVAL v5.10.1</u>	<u>OCIL v2.0</u>
Reporting Formats	<u>ARF v1.1</u>	<u>AI v1.1</u>	
Enumerations	<u>CCE v5</u>	<u>CPE v2.3</u>	<u>CVE</u>
Measurement / Scoring Systems	<u>CVSS v2.0</u>	<u>CCSS v1.0</u>	
Integrity Model	<u>TMSAD v1.0</u>		

Introductory Notes - SCAP v1.2 standards relevant to SCAP Security Guide



Languages	<u>XCCDF v1.2</u>	<u>OVAL v5.10.1</u>	<u>OCIL v2.0</u>
Reporting Formats	<u>ARF v1.1</u>		
Enumerations	<u>CCE v5</u>	<u>CPE v2.3</u>	<u>CVE</u>

Introductory Notes - SCAP v1.2 standards to be covered within today's session



Languages	<u>XCCDF v1.2</u>	<u>OVAL v5.10.1</u>	
-----------	-----------------------------------	-------------------------------------	--

And a bit of [OVAL v5.11.1](#)

Extensible Configuration Checklist Description Format (XCCDF) Version 1.2

- Organizational layer
 - Structured collection of security configuration rules for some set of target systems

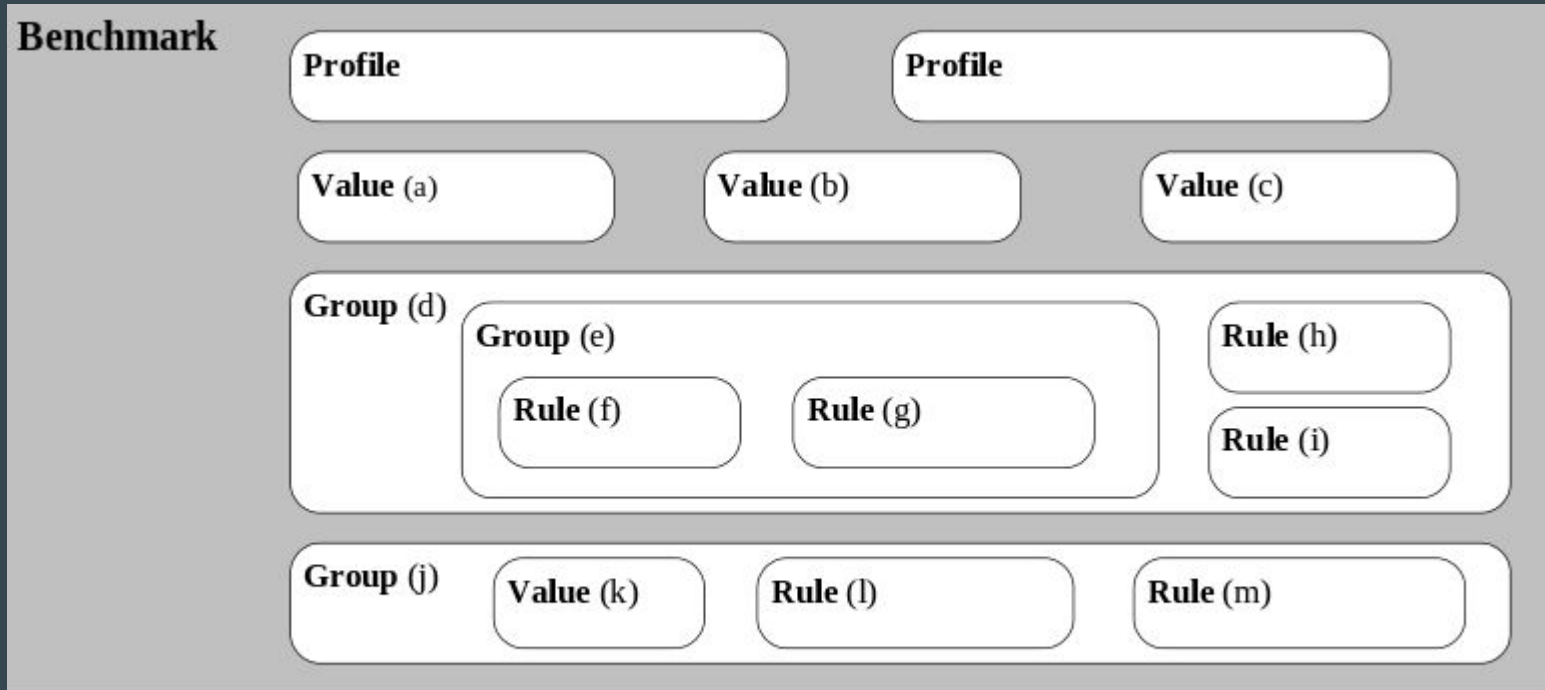


Figure 1: Typical Structure of a Benchmark

Extensible Configuration Checklist Description Format (XCCDF) Version 1.2



- Organizational layer
 - XCCDF benchmark document contains:
 - single root `<xccdf:Benchmark>` element
 - (one or) multiple `<xccdf:Profile>`
 - (one or) multiple `<xccdf:Group>` elements
 - ..
 - `<xccdf:Group>`
 - represents portion of a benchmark
 - contains (multiple) `<xccdf:Rule>`, `<xccdf:Value>`, and other `<xccdf:Group>` elements

Extensible Configuration Checklist Description Format (XCCDF) Version 1.2



- Example `<xccdf:Rule>`

```
<xccdf:Rule id="xccdf_org.example_rule_pwd-perm" selected="1" weight="6.5"
severity="high">
  <xccdf:title>Password File Permission</xccdf:title>
  <xccdf:description>Check the access control on the password file.
    Normal users should not be able to write to it.
  </xccdf:description>
  <xccdf:requires idref="xccdf_org.example_rule_passwd-exists"/>

  <xccdf:fixtext>
    Set permissions on the passwd file to owner-write, world-read
  </xccdf:fixtext>
  <xccdf:fix strategy="restrict" reboot="0" disruption="low">
    chmod 644 /etc/passwd
  </xccdf:fix>
  <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
    <xccdf:check-content-ref href="ovaldefs.xml" name="oval:org.example:def:123"/>
  </xccdf:check>
</xccdf:Rule>
```

Open Vulnerability and Assessment Language (OVAL)



Standardizes three main steps of the assessment process:

- representing configuration information of systems for testing,
- analyzing the system for the presence of the specified machine state (vulnerability, configuration, patch state etc.),
- reporting the results of the assessment



SSG “shorthand” format

SSG “shorthand” format

- A way to abstract away the complexity of XCCDF and OVAL during development



XCCDF case -- for the sake of simplicity we will consider benchmark:

- containing just one profile, two groups and one rule
- the description of the benchmark to be limited to one sentence

Comparison



Contributing New Rule

Contributing New Rule - Contributing XCCDF part - Steps



Yet before we begin -- relevant SSG git repository folder is:

`<product>/input/xccdf`

- 1) Identify the SSG `<product>` (RHEL/6, RHEL/7, ...) the new rule should be inserted into

Contributing New Rule - Contributing XCCDF part (continued) - Steps



2) Provide the data for basic XCCDF rule skeleton:

```
<Rule id="RULE_ID">
  <title>RULE_TITLE</title>
  <description>RULE_DESCRIPTION</description>
  <ocil clause="it is not">OCIL_FOR_RULE</ocil>
  <rationale>RULE_RATIONALE</rationale>
  <ident cce="RULE_CCE_ID" stig="RULE_STIG_ID" />
  <oval id="RULE_ID" />
</Rule>
```

Note: The **XCCDF** RULE_ID needs to match **OVAL** RULE_ID.

Contributing New Rule - Contributing XCCDF part (continued) - Steps

2) Provide the data for basic XCCDF rule skeleton (**result**):



```
<Rule id="require_smb_client_signing">

  <title>Require Client SMB Packet Signing, if using <tt>smbclient</tt></title>
  <description>To require samba clients running <tt>smbclient</tt> to use
  packet signing, add the following to the <tt>[global]</tt> section
  of the Samba configuration file, <tt>/etc/samba/smb.conf</tt>:
  <pre>client signing = mandatory</pre>
  Requiring samba clients such as <tt>smbclient</tt> to use packet
  signing ensures they can only communicate with servers that support
  packet signing.
</description>
<ocil clause="it is not">To verify that Samba clients running smbclient must
  use packet signing, run the following command:
  <pre>$ grep signing /etc/samba/smb.conf</pre>
  The output should show:
  <pre>client signing = mandatory</pre>
</ocil>
<rationale>Packet signing can prevent man-in-the-middle attacks which
  modify SMB packets in transit.
</rationale>
<ident cce="26328-5" stig="RHEL-06-000272" />
<oval id="require_smb_client_signing" />

</Rule>
```

Contributing New Rule - Contributing XCCDF part

(continued) - Steps



- 3) Decide if new rule is **services** or other **system** component related. Based on that select one of **services** or **system** subfolders of particular `<product>/input/xccdf` folder

In this concrete example the “**require_smb_client_signing**” is related with the Samba service => we would place the definition under:

`<product>/input/xccdf/services`

Contributing New Rule - Contributing XCCDF part

(continued) - Steps



- 3) Decide if new rule is **services** or other **system** component related. Based on that select one of **services** or **system** subfolders of particular `<product>/input/xccdf` folder

E.g. rule related with “**network**” system component would go under:

`<product>/input/xccdf/system`

Same case for some “**SELinux**” system component related rule.

Note: The proper decision needs advanced familiarity / experience with SSG git repository content.

Contributing New Rule - Contributing XCCDF part

(continued) - Steps



Intermediary summary:

- we know the product for new XCCDF rule,
- we got the content for the XCCDF rule,
- we got the location for the XCCDF rule.

What remains:

- is to determine the correct `<xccdf:Group>` to place the rule under (or create a new one if suitable group doesn't exist yet),
- instruct the SSG build system to include the newly added rule into the benchmark for the product during the build
- rebuild the benchmark and confirm the rule is included

Contributing New Rule - Contributing XCCDF part (continued) - Steps



- 4) Determine the correct `<xccdf:Group>` to place the rule under (or create a new one if suitable group doesn't exist yet)

```
File Edit View Search Terminal Help
[iankko@localhost services]$ pwd
/tmp/github/scap-security-guide/RHEL/6/input/xccdf/services
[iankko@localhost services]$ grep -rHn '<Group id=\"' * | more
avahi.xml:1:<Group id="avahi">
avahi.xml:10:<Group id="service_avahi-daemon_disabled_group">
avahi.xml:35:<Group id="avahi_configuration">
base.xml:1:<Group id="base">
cron.xml:1:<Group id="cron_and_at">
cron.xml:66:<Group id="restrict_at_cron_users">
dhcp.xml:1:<Group id="dhcp">
dhcp.xml:15:<Group id="disabling_dhcp_server">
dhcp.xml:59:</Group> <!-- <Group id="disabling_dhcp_server"> -->
dhcp.xml:61:<Group id="dhcp_server_configuration">
dhcp.xml:176:</Group> <!-- <Group id="dhcp_server_configuration"> -->
dhcp.xml:178:<Group id="disabling_dhcp_client">
dhcp.xml:225:</Group> <!-- <Group id="disabling_dhcp_client"> -->
dhcp.xml:227:<Group id="dhcp_client_configuration">
dhcp.xml:283:</Group> <!-- <Group id="dhcp_client_configuration"> -->
dhcp.xml:284:</Group> <!-- <Group id="dhcp"> -->
dns.xml:1:<Group id="dns">
dns.xml:9:<Group id="disabling_dns_server">
dns.xml:50:</Group> <!--<Group id="disabling_dns_server">-->
dns.xml:52:<Group id="dns_server_isolation">
dns.xml:59:<Group id="dns_server_dedicated">
```

Contributing New Rule - Contributing XCCDF part (continued) - Steps



- 4) Determine the correct `<xccdf:Group>` to place the rule under (or create a new one if suitable group doesn't exist yet)

Based on experience, we have decided to place the code for the “`require_smb_client_signing`” rule into the “`smb_restrict_file_sharing`” XCCDF group located in file: `<product>/input/xccdf/services/smb.xml`

```
Terminal
File Edit View Search Terminal Help
[iankko@localhost services]$ pwd
/tmp/github/scap-security-guide/RHEL/6/input/xccdf/services
[iankko@localhost services]$ grep -rHn 'require_smb_client_signing' *
smb.xml:99:<Rule id="require_smb_client_signing">
smb.xml:122:<oval id="require_smb_client_signing" />
[iankko@localhost services]$
```

Contributing New Rule - Contributing XCCDF part

(continued) - Steps



- 5) Instruct the SSG build system to include the newly added rule into the benchmark for the product during the build

This means the **XML file** holding the definition of the **XCCDF group** being parent group for the newly added **XCCDF rule** is listed in `<product>/input/guide.xslt` transformation for that `<product>`:

```
Terminal
File Edit View Search Terminal Help
[iankko@localhost 6]$ pwd
/tmp/github/scap-security-guide/RHEL/6
[iankko@localhost 6]$ grep 'smb.xml' input/guide.xslt
    <xsl:apply-templates select="document('xccdf/services/smb.xml')" />
[iankko@localhost 6]$
```


Contributing New Rule - Contributing XCCDF part (continued) - Steps



- 6) Include the newly added XCCDF rule into:
- some existing profile for product
 - new profile for product

```
Terminal
File Edit View Search Terminal Help
<Profile id="ftp" extends="server" xmlns="http://checklists.nist.gov/xccdf/1.1"
>
<title override="true">ftp</title>
<description override="true">This profile is for FTP servers.</description>
<select idref="package_vsftpd_installed" selected="true"/>
<select idref="ftp_log_transactions" selected="true"/>
<select idref="ftp_present_banner" selected="true"/>
<select idref="ftp_restrict_to_anon" selected="true"/>
<select idref="ftp_disable_uploads" selected="true"/>
<select idref="ftp_home_partition" selected="true"/>
</Profile>
```

Note: Ensure this profile is configured to be built for benchmark !!

Contributing New Rule - Contributing XCCDF part (continued) - Steps



7) Rebuild the benchmark:

```
Terminal
File Edit View Search Terminal Help
[iankko@localhost 6]$ pwd
/tmp/github/scap-security-guide/RHEL/6
[iankko@localhost 6]$ make clean >& /dev/null; echo $?
0
[iankko@localhost 6]$ make content >& /dev/null; echo $?
0
[iankko@localhost 6]$
```

8) And verify the newly added rule got included:

```
Terminal
File Edit View Search Terminal Help
[iankko@localhost 6]$ pwd
/tmp/github/scap-security-guide/RHEL/6
[iankko@localhost 6]$ grep 'Rule id=\"require_smb_client_signing' output/ssg-rhel6-xccdf.xml
<Rule id="require_smb_client_signing" selected="false" severity="low">
[iankko@localhost 6]$
```

Contributing New Rule - Contributing OVAL part



Still remember the [<xccdf:Rule> example](#) ?

The OVAL check defined for the rule (e.g. “**require_smb_client_signing**”) is:

- what will be listed in the `<xccdf:check-content-ref>` element for the XCCDF rule,
- what will be used to scan the particular system property.

Contributing New Rule - Contributing OVAL part

(continued)



Relevant SSG repository paths for contributing new OVAL checks are:

`<product>/input/oval`

`shared/input/oval`

`<product>/input/oval/oval_5.11`

How do I know which of them to use for my use case?

Contributing New Rule - Contributing OVAL part

(continued) - Purpose of different locations of various OVAL checks



- `<product>/input/oval` -- Is dedicated to OVAL checks that are specific to only one product:
 - they would NOT work on different product
 - they contain just this product in their `<platform>` element

Contributing New Rule - Contributing OVAL part

(continued) - Purpose of different locations of various OVAL checks



- **shared/input/oval** -- Is dedicated to hold OVAL checks for multiple (two or more) products:
 - they have been confirmed to work on two or more products
 - they contain:
 - two or more platforms in their **<platform>** element (listed either as two platform elements or as multi_platform element)

Contributing New Rule - Contributing OVAL part

(continued) - Purpose of different locations of various OVAL checks



- `<product>/input/oval/oval_5.11` -- Is dedicated to hold OVAL checks:
 - using OVAL language constructs that have appeared starting from OVAL-5.11 version of the language (e.g. `<linux:systemdunitdependency_{test, object, state}>` or e.g. `<glob_to_regex>` function)
 - but also definitions for **dependent** (`<extend_definition>`) **OVAL 5.10 checks**

Contributing New Rule - Contributing OVAL part

(continued) - Purpose of different locations of various OVAL checks



Q: Shouldn't there be 4-th ([shared/input/oval/oval_5.11](#)) location yet?

A: Yes. In the moment there are two or more products (e.g. [RHEL/7](#) and [Fedora](#)) utilizing OVAL-5.11 language constructs already.

Contributing New Rule - Contributing OVAL part

(continued) - Purpose of different locations of various OVAL checks



Q: Why all this complexity? It's confusing!

A: The reason is compatibility. There are different openscap versions:

- RHEL-5 openscap-1.0.x (OVAL-5.10, but not 5.11),
- RHEL-6 openscap-1.0.x (OVAL-5.10, but not 5.11),
- RHEL-7, Fedora - openscap-1.2.x (OVAL-5.11),
- Debian/8 (Jessie) libopenscap8-1.0.x (OVAL-5.10, but not 5.11)
- Debian Testing (Stretch) libopenscap8-1.2.x (OVAL-5.11)

Contributing New Rule - Contributing OVAL part (continued) - Purpose of different locations of various OVAL checks



Q: Why all this complexity? It's confusing! (continued)

A: The idea is to support **all** content to be able to be built on **any** system. With OVAL-5.10 oscap versions we just throw out / don't include checks using OVAL-5.11 language constructs. With OVAL-5.11 oscap versions we include both (**OVAL-5.10 and OVAL-5.11**) checks.

Contributing New Rule - Contributing OVAL part

(continued) - Purpose of the `<platform>` element



The `<platform>` element in OVAL checks is used to represent list of products (e.g. `RHEL/6`, `RHEL/7`, `Fedora`) where this OVAL has been tested and recognized as being working properly:

- for `<product>/input/oval` checks the `<platform>` should be set just to that product (e.g. “`Red Hat Enterprise Linux 6`” for `RHEL/6` product)

Contributing New Rule - Contributing OVAL part (continued) - Purpose of the `<platform>` element



- for `<product>/input/oval/oval_5.11` checks the `<platform>` should be just to that product (e.g. “Red Hat Enterprise Linux 7” for RHEL/7 product)

Contributing New Rule - Contributing OVAL part

(continued) - Purpose of the `<platform>` element



- for `shared/input/oval` checks the `<platform>` should be set to cover **all** platforms this OVAL is shared between. It often has form of “`multi_platform_rhel`” or “`multi_platform_fedora`”

Contributing New Rule - Contributing OVAL part (continued) - Purpose of the `<platform>` element



Q: What will happen when `<platform>` element is not set properly?

A: Expect troubles ahead.

When `<platform>` element is too liberal, **make validate** for that product will report existing, but unreferenced (unused) OVAL. On the other hand `<platform>` tag being too strict will result in “notchecked” result (read like OVAL check is not available) for that product.

Contributing New Rule - Contributing OVAL part

(continued) - Steps



- 1) Identify the `<product>` (e.g. `RHEL/6`, `RHEL/7` ...) the new OVAL check is to be added to,
- 2) Based on system system property to be checked determine OVAL language constructs to use:
 - config key set => `<textfilecontent54_object>`,
 - file, dir having perms => `<file_object>`
 - (systemd) service => `<systemdunitdependency_object>`
 - (chkconfig) service => `<runlevel_object>`
 - rpm package => `<rpminfo_object>`
 - deb package => `<dpkginfo_object>`
 - `<partition_object>`, `<sysctl_object>`, `<password_object>` ...

Contributing New Rule - Contributing OVAL part

(continued) - Steps



The identical / universal parts of the various OVAL check types:

```
<def-group>
  <definition class="compliance" id="no_files_unowned_by_user" version="1">
    <metadata>
      <title>Find files unowned by a user</title>
      <affected family="unix">
        <platform>Red Hat Enterprise Linux 6</platform>
      </affected>
      <description>All files should be owned by a user</description>
      <reference source="rmercier" ref_id="20131218" ref_url="test_attestation" />
    </metadata>
    <criteria>
      <criterion comment="Check all files and make sure they are owned by a user"
        test_ref="no_files_unowned_by_user_test" />
    </criteria>
  </definition>

  ...

</def-group>
```


Contributing New Rule - Contributing OVAL part

(continued) - Steps

The different parts of the various OVAL check types:



```
<criteria>
  <criterion comment="Check all files and make sure they are owned by a user"
    test_ref="no_files_unowned_by_user_test" />
</criteria>
</definition>

<unix:file_state id="file_permissions_unowned_userid_list_match" version="1">
  <unix:user_id var_check="at least one"
    var_ref="file_permissions_unowned_userid_list"
    datatype="int" />
</unix:file_state>

<local_variable id="file_permissions_unowned_userid_list"
  comment="List of valid user ids" datatype="int" version="1">
  <object_component item_field="subexpression"
    object_ref="file_permissions_unowned_userid_list_object" />
</local_variable>

...|
```

Contributing New Rule - Contributing OVAL part (continued) - Steps



3) Create the OVAL check definition itself

Method of Creation	OVAL Knowledge	Time / Speed	Check Applicability
From templates	Almost none	Instant	Depends on template
Copy existing	Basic understanding	Slower	Depends on original
Manually	Advanced understanding	Very slow	Almost everything [*]

[*] If it's not doable using existing constructs, probably a new construct needs to be proposed to OVAL Board for consideration.

Contributing New Rule - Contributing OVAL part (continued) - Steps



- 3) Create the OVAL check definition itself (LIVE DEMO):
- From **templates**
 - Copy existing
 - Manually

Contributing New Rule - Contributing OVAL part (continued) - Steps



- 4) Once created save the OVAL definition under “**RULE_ID.xml**” and place it under (one of):
- `<product>/input/oval`
 - `<product>/input/oval/oval_5.11`
 - `<shared/input/oval`
- paths

Contributing New Rule - Contributing OVAL part (continued) - Steps



- 5) No need to instruct the SSG build system to include the newly created OVAL into final OVAL file !
- 6) Rebuild the benchmark and verify the new OVAL is working (former “notchecked” for the XCCDF rule should turn into one of “pass”, “fail”, or “error”)

OVAL - What we have just covered?



Basic Information:

- How OVAL standard fits into the picture
- Purpose of `<platform>` element
- Steps to create new OVAL check

Advanced Topics / Selected Use Cases

- Purpose of different locations of various OVAL checks
- How to contribute new rule from scratch
- How to contribute new rule from `templates`
(Using SSG `templates` to contribute new OVAL checks)

Contributing New Rule - Contributing remediation part



Still remember the [<xccdf:Rule> example](#) ?

The remediation script defined for the rule
(e.g. “**require_smb_client_signing**”) is:

- what will be listed in the `<xccdf:fix>` element for the XCCDF rule,
- what will be used to **correct** the particular system property
(after the system property in question has been scanned and recognized as not meeting the requirement)

Contributing New Rule - Contributing remediation part (continued) - Steps



Relevant SSG git repository folders for remediation scripts are:

`<product>/input/remediations/bash`
`shared/input/remediations/bash`

- 1) Identify the product (**RHEL/6**, **RHEL/7** ...) this new remediation script is intended for

Contributing New Rule - Contributing remediation part (continued) - Steps



- 2) Create the Bash code for the remediation script itself:
 - SSG provides “library of functions” intended to be re-used when creating remediation scripts:

[scap-security-guide/shared/remediations/bash/templates/remediation_functions](#)

Therefore first check if some of the routines already implemented there would not be reusable for your particular case.

Contributing New Rule - Contributing remediation part (continued) - Steps



- 2) Create the Bash code for the remediation script itself:
 - If there doesn't exist already available **remediation_function** for your use case it is necessary to create a new one:
 - Either again from the SSG **templates** (quicker, but template for concrete case might not present)
 - Or by copying an existing one
 - Or by creating own one manually (from scratch)

Contributing New Rule

- Contributing remediation part
(continued) - The purpose and syntax of “platform” element



Purpose:

Instruct the benchmark there's remediation script
(`<xccdf:fix>` element) available

Syntax:

- For the case of remediations specific to `<product>`
`# platform = Red Hat Enterprise Linux 6`
- For the case of `shared/` remediation scripts:
`# platform = multi_platform_rhel`

Contributing New Rule - Contributing remediation part (continued) - Example remediation script



```
# platform = Red Hat Enterprise Linux 6
. /usr/share/scap-security-guide/remediation_functions
populate var_umask_for_daemons

grep -q ^umask /etc/init.d/functions && \
sed -i "s/umask.*/umask $var_umask_for_daemons/g" /etc/init.d/functions
if ! [ $? -eq 0 ]; then
    echo "umask $var_umask_for_daemons" >> /etc/init.d/functions
fi
```

Contributing New Rule - Contributing remediation part (continued) - Steps



- 3) Once created save the remediation script under “**RULE_ID.sh**” and place it under (one of):
- `<product>/input/remediations/bash`
 - `<shared>/input/remediations/bash`
- paths

Contributing New Rule - Contributing remediation part (continued) - Steps



- 4) No need to instruct the SSG build system to include the newly created remediation script into final XCCDF file !
- 5) Rebuild the benchmark and verify the new remediation is working (former “fail” result for the XCCDF rule should turn into one of “fixed”, or “error” results if the `--remediate oscap` option or “Remediate” SCAP Workbench checkbox have been specified)



Selected Use Cases

Selected Use Cases

- How to port existing rules, profiles, benchmarks across different products



- Porting XCCDF
- Porting OVAL rules
- Porting remediation scripts
- Create whole new benchmarks for new products

Thanks!

Contact us:

scap-security-guide@lists.fedorahosted.org

<http://www.open-scap.org/security-policies/scap-security-guide/>

<http://www.open-scap.org/security-policies/scap-security-guide/#support>



Appendix A:



Source Code / Repository Notes

Source Code / Repository Notes - Where to obtain the source code



SSG code is hosted at GitHub:

A screenshot of the GitHub repository page for OpenSCAP / scap-security-guide. The page shows the repository name, navigation tabs for Code, Issues (79), Pull requests (3), Wiki, Pulse, Graphs, and Settings. It includes statistics for 4,739 commits, 1 branch, 23 releases, and 42 contributors. A green 'New pull request' button is visible. The latest commit is by 'Isimluk' (#960) from 'lankko/debian_make_fix_for_openscap_1.0.x', committed 3 days ago. A preview of a Chromium commit is also shown at the bottom.

OpenSCAP / scap-security-guide

Unwatch 55 Unstar 111 Fork 64

Code Issues 79 Pull requests 3 Wiki Pulse Graphs Settings

Baseline compliance content in SCAP formats <http://www.open-scap.org/security-policies/scap-security-guide> — Edit

4,739 commits 1 branch 23 releases 42 contributors

Branch: master New pull request

New file Find file SSH git@github.com:OpenSCAP/scap Download ZIP

Isimluk Merge pull request #960 from lankko/debian_make_fix_for_openscap_1.0.x Latest commit a82925e 3 days ago

Chromium Share xccdf-addremediations.xslt among the products 9 days ago

Source Code / Repository Notes - Developer workflow



- We follow the [GitHub](#) flow:
 - Fork the repo
 - Create feature branch in forked repo
 - Perform, test locally, and commit the code changes
 - Push the changes to the feature branch
 - Create a pull request (PR) with proper description, labels, milestone
 - Wait till Jenkins finishes the PR testing:
 - If PR testing by Jenkins failed, inspect the failures, and fix them (possibly commit additional changes)
 - If PR testing by Jenkins passed, your job is done

Source Code / Repository Notes - Pull request reviewer workflow



- **Reviewing existing PRs is as much as important as providing new code changes !!!**
- PR Reviewer Workflow:
 - Check out the [list of existing PRs](#)
 - Apply the proposed code changes from PR to be reviewed on top of your local branch (synchronized with upstream master)
 - Build the content
 - Test the particular code changes (verify announced problem is corrected)
 - If testing succeeded, ACK and merge the PR into upstream

Source Code / Repository Notes - Building the SSG content



- Clone upstream master (or synced master in your fork)
- Build RPM from the upstream content:
`# make clean`
`# make SSG_VERSION_IS_GIT_SNAPSHOT=no rpm`
- Or [install scap-security-guide RPM package](#) according to instructions for your distribution

Source Code / Repository Notes

Repository structure -
Currently supported SSG products



```
Terminal
File Edit View Search Terminal Help
[iankko@localhost scap-security-guide]$ tree -d -L 1 -C
.
├── Chromium
├── config
├── Debian
├── docs
├── Fedora
├── Firefox
├── JBossEAP5
├── JBossFuse6
├── JRE
├── OpenStack
├── RHEL
├── RHEVM3
├── shared
└── Webmin

14 directories
[iankko@localhost scap-security-guide]$
```

Source Code / Repository Notes

Repository structure -
Releases of Red Hat Enterprise Linux equipped with SCAP
content



```
Terminal
File Edit View Search Terminal Help
[iankko@localhost scap-security-guide]$ tree -d -L 1 -C RHEL/
RHEL/
├── 5
├── 6
└── 7

3 directories
[iankko@localhost scap-security-guide]$
```


Source Code / Repository Notes - Repository structure-

Directory structure of a concrete product ([RHEL/6](#))



```
Terminal
File Edit View Search Terminal Help
[iankko@localhost scap-security-guide]$ tree -d -L 1 -C RHEL/6
RHEL/6
├── build
├── dist
├── input
├── kickstart
├── output
├── tests
├── transforms
└── utils

8 directories
[iankko@localhost scap-security-guide]$
```

Exceptions are [JBossEAP5](#) and [JBossFuse6](#) products.

Source Code / Repository Notes - Repository structure-

Directory structure of **input/** folder per product (**RHEL/6**)



```
Terminal
File Edit View Search Terminal Help
[iankko@localhost scap-security-guide]$ tree -d -L 1 -C RHEL/6/input/
RHEL/6/input/
├── auxiliary
├── intro
├── oval
├── profiles
├── remediations
└── xccdf

6 directories
[iankko@localhost scap-security-guide]$
```

Source Code / Repository Notes

Repository structure-
Meaning and structure of special **shared/** directory



```
Terminal
File Edit View Search Terminal Help
[iankko@localhost scap-security-guide]$ tree -d -L 1 -C shared/
shared/
├── modules
├── oval
├── references
├── remediations
├── transforms
└── utils

6 directories
[iankko@localhost scap-security-guide]$
```

- Used to hold content applicable to two and more products
- Easier maintenance (avoids duplicated data)
- Process of unification (still) not finished

Appendix B:



SSG most often used OVAL objects

SSG most often used OVAL objects - RHEL/6 product



```
Count of used OVAL objects:
```

```
=====
textfilecontent54_object      167
runlevel_object               55
file_object                   41
rpminfo_object                30
partition_object              16
variable_object               15
xmlfilecontent_object         9
selinuxsecuritycontext_object 2
rpmverifyfile_object          2
password_object                1
environmentvariable58_object  1
interface_object              1
[iankko@localhost utils]$
```

Created by:

```
$ ./shared/utils/count_oval_objects.py ssg-rhel6-xccdf.xml
```

SSG most often used OVAL objects - RHEL/7 product



```
Count of used OVAL objects:
```

```
=====
textfilecontent54_object      119
systemdunitdependency_object  56
file_object                   32
rpminfo_object                32
variable_object               12
partition_object              11
sysctl_object                  2
selinuxsecuritycontext_object 2
rpmverifyfile_object          2
password_object                1
[iankko@localhost scap-security-guide]$
```