SCAP & STIG Workshop
Red Hat Enterprise Linux 7
# Table of Contents

1 INTRODUCTION.................................................................................................................. 4

2 SCAP SECURITY GUIDE PROJECT..................................................................................... 5

  2.1 Background.................................................................................................................... 5

  2.2 Project Charter............................................................................................................. 6

  2.3 Installation.................................................................................................................... 6

3 UNDERSTANDING THE COMPONENTS............................................................................. 7

  3.1 Alphabet Soup Explained............................................................................................. 7

  3.2 Content Location.......................................................................................................... 7

  3.3 OpenSCAP.................................................................................................................... 7

  3.4 XCCDF.......................................................................................................................... 8

    3.4.1 Sample XCCDF Code............................................................................................ 9

    3.4.2 Profiles.................................................................................................................. 11

  3.5 OVAL............................................................................................................................... 12

4 OPERATION......................................................................................................................... 13

  4.1 XCCDF Validation......................................................................................................... 13

  4.2 Generate HTML Guide................................................................................................. 14

  4.3 Performing a Scan......................................................................................................... 16

  4.4 Interpreting Results....................................................................................................... 17

    4.4.1 HTML..................................................................................................................... 17

    4.4.2 XML....................................................................................................................... 20

  4.5 Remediation.................................................................................................................. 22

    4.5.1 Remediation? How Do?.......................................................................................... 22

  4.6 Scanning Automation..................................................................................................... 24

  4.7 Alternative Tools........................................................................................................... 25

    4.7.1 SPAWAR SCAP Compliance Checker (SPAWAR SCC)....................................... 25

    4.7.2 Red Hat Network Satellite...................................................................................... 25
Illustration Index

Illustration 1: HTML Results Output: Target Information................................................................. 17
Illustration 2: HTML Results Output: Rule Results Summary.......................................................... 18
Illustration 3: HTML Results Output: Rule Results Table............................................................... 18
Illustration 4: HTML Results Output: Rule Detail............................................................................ 19
Illustration 5: Sample SCAP Integration for RHN Satellite 5.5....................................................... 25

Index of Tables

Table 1: XCCDF Rule Elements........................................................................................................ 10
Table 2: Scan Results XML Output.................................................................................................. 21
## 1 INTRODUCTION

This workshop is intended to assist you in developing familiarity with, and skill in, the installation and customization of SCAP content for Red Hat Enterprise Linux 7, utilizing the DISA STIG baseline. Upon completion of this workshop, you should be able to provision a RHEL7 system directly into STIG compliance, perform continuous monitoring scans, and use SCAP Workbench to customize your security baseline.

If attending in person, each attendee will be given a virtual machine to use for this workshop. The hostnames follow the syntax of:

    student.X.lab.openscap.io

The initial login will be student.X, with a password of “redhat!2016.” Use sudo to escalate to root as needed.

This workshop has been designed to be delivered over a two hour period, with each chapter building upon the prior. Should you have any questions, please raise your hand and the instructors will be happy to assist.
2 SCAP SECURITY GUIDE PROJECT

The scap-security-guide project (SSG) delivers security guidance, baselines, and associated validation mechanisms using the Security Content Automation Protocol (SCAP). We currently provide content for Red Hat Enterprise Linux and JBoss Enterprise Application Server, as well as Fedora. Additionally, the upstream development community has expanded to CentOS and Debian.

The SSG project aspires to bridge the gap between generalized policy and specific implementation guidance, in SCAP formats to support automation whenever possible.

The project homepage is https://github.com/OpenSCAP/scap-security-guide.

2.1 Background

Security guidance documents, such as the NSA RHEL5 SNAC Guide, have historically been authored in static formats such as PDF files. This left much to be desired, particularly around automation, and Linux deployments were forced to create their own scripted hardening processes. These processes were authored in silos, by unique contractor teams, and generally not shared across programs. Additionally, from a vendor perspective, supporting hardened environments was commercially unfeasible. As each shop developed their own processes, how could the vendor know what was changed and how to support it? The system was broke.

In May 2011, the Information Assurance Directorate of the National Security Agency (NSA IAD) invited Red Hat to begin collaboration on high quality security guidance for Red Hat Enterprise Linux 6. It was decided the content would be authored in SCAP formats to support automation, and thus the SCAP Security Guide project was born.

Development of security guidance continued through May 2012, at which point Red Hat and NSA IAD contacted DISA Field Security Operations (DISA FSO) to begin development of the Red Hat Enterprise Linux 6 STIG. For the first time, content development of a STIG would be open sourced. The movement towards transparent collaboration – between government agencies, industrial base partners, and content consumers – further reflects acceptance by the NSA and DISA FSO of the benefits of open source development models. On February 12th, 2013, DISA FSO released the first public draft RHEL6 STIG - derived from the upstream SCAP Security Guide project.
This community powered innovation model has allowed the rapid creation of security baselines at dramatically reduced costs, shortened timeframes, and has produced higher-quality content that has now become official U.S. Government security baselines.

2.2 Project Charter

The SSG community has a single goal: Develop usable security baselines across all Red Hat technology areas. Patches welcome!

2.3 Installation

This workshop assumes you have access to a RHEL7 machine.

1. SCAP requires two components:
   - Content: Automation content is provided by the `scap-security-guide` package
   - Interpreter: Akin to how HTML content is rendered through a web browser, SCAP content requires an interpreter. This is provided by the `openscap scanner` package.

   ```bash
   sudo yum -y install scap-security-guide openscap-scanner
   ```

2. Red Hat provides a GUI customization tool for SCAP content, called SCAP Workbench. The `scap-workbench` package must be installed:

   ```bash
   sudo yum -y install scap-workbench
   ```

3. Lastly, Apache is used to display HTML scan reports back from the RHEL7 machine. This is required for Red Hat’s lab environment, which does not support remote desktop displays:

   ```bash
   sudo yum -y install httpd
   rm /etc/httpd/conf.d/welcome.conf
   sudo systemctl start httpd
   sudo systemctl enable httpd
   firewall-cmd --zone=public --add-port=80/tcp --permanent
   firewall-cmd --reload
   ```
3 UNDERSTANDING THE COMPONENTS

3.1 Alphabet Soup Explained

SCAP content is made of many components with a variety of acronyms attached to them.

- XCCDF: eXtensible Configuration Checklist Description
- OVAL: Open Vulnerability Assessment Language
- CPE: Common Platform Enumerator
- CCE: Common Configuration Enumerator

3.2 Content Location

The SSG project deploys content into multiple directories:

- `/usr/share/xml/scap/ssg/content/`
  Houses SCAP content utilizing the following naming conventions:

- `/usr/share/scap-security-guide/kickstart`
  Profile specific kickstart scripts, such as provisioning directly into PCI and STIG-ready baselines

- `/usr/share/doc/scap-security-guide-*/table-{profile}-*.html`
  HTML tables that map NIST 800-53 back to controls, forming the base for RTMs

3.3 OpenSCAP

The SCAP protocol suite contains multiple complex data exchange formats that are used to transmit important vulnerability, configuration, and other security data. Historically, there have been few tools that provide a way to query this data in the needed format. This lack of tooling makes the barrier to entry very high and discourages adoption of these protocols by the community. It's the goal of the OpenSCAP project to create a framework of libraries and tools to improve the accessibility of SCAP and enhance the usability of the information it presents.

SCAP is a protocol, akin to HTML, and consumers must use a protocol interpreter. OpenSCAP provides that functionality for the RHEL platform. The OpenSCAP homepage can be found at http://www.open-scap.org/.
3.4 XCCDF

As documented by NIST on their XCCDF specification page at http://scap.nist.gov/specifications/xccdf/:

XCCDF is a specification language for writing security checklists, benchmarks, and related kinds of documents. An XCCDF document represents a structured collection of security configuration rules for some set of target systems. The specification is designed to support information interchange, document generation, organizational and situational tailoring, automated compliance testing, and compliance scoring. The specification also defines a data model and format for storing results of benchmark compliance testing. The intent of XCCDF is to provide a uniform foundation for expression of security checklists, benchmarks, and other configuration guidance, and thereby foster more widespread application of good security practices.

XCCDF documents are expressed in XML, and may be validated with an XML Schema-validating parser.

Development of the XCCDF specification is being led by NSA, with contributions from other agencies and organizations. The current public draft of the specification document and related files can be downloaded below. A mailing list for XCCDF developers is available, please subscribe to participate in discussions. A publicly available archive of the XCCDF mailing list is also available.

Within the project, XCCDF content is retained in /usr/share/xml/scap/ssg/content/.
3.4.1 Sample XCCDF Code

To understand XCCDF, let's review the `disable_telnet_service` rule within `ssg-rhel7-xccdf.xml`.

```bash
$ vim /usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml
```

Once loaded, locate the `disable_telnet_service` rule:

```xml
/<Rule id="service_telnetd_disable
```

What identifies the rule title?

Where does XCCDF tie into the OVAL check system?
The code can be broken into the following parts:

<table>
<thead>
<tr>
<th>XML Tag</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Rule&gt;</td>
<td>Indicates:</td>
</tr>
<tr>
<td></td>
<td>• Rule Name</td>
</tr>
<tr>
<td></td>
<td>... id=&quot;service_telnetd_disable&quot; ....</td>
</tr>
<tr>
<td></td>
<td>• Severity</td>
</tr>
<tr>
<td></td>
<td>... severity=&quot;high&quot; ...</td>
</tr>
<tr>
<td></td>
<td>• If the rule is enforced / selected</td>
</tr>
<tr>
<td></td>
<td>.... selected=&quot;true&quot; ...</td>
</tr>
<tr>
<td>&lt;title&gt;</td>
<td>The XCCDF Rule title</td>
</tr>
<tr>
<td>&lt;description&gt;</td>
<td>Largely free form, contains implementation instructions and any policy references</td>
</tr>
<tr>
<td>&lt;rationale&gt;</td>
<td>Identifies the logic behind applying this &lt;rule&gt; to the system. Answers “why is this important?”</td>
</tr>
<tr>
<td>&lt;ident system...&gt;</td>
<td>Identifies CCE mapping</td>
</tr>
<tr>
<td>&lt;check system...&gt;</td>
<td>Identifies which version of OVAL the check was written for</td>
</tr>
<tr>
<td>&lt;check-content-ref...&gt;</td>
<td>There are two elements for this tag:</td>
</tr>
<tr>
<td></td>
<td>• href=.....</td>
</tr>
<tr>
<td></td>
<td>Path to associated OVAL file</td>
</tr>
<tr>
<td></td>
<td>• name=.....</td>
</tr>
<tr>
<td></td>
<td>Identifies the corresponding OVAL rule name, such as oval:ssg:def:302.</td>
</tr>
<tr>
<td></td>
<td>If one were to search for this rule within the href= identified file, they would be brought to the corresponding OVAL code.</td>
</tr>
</tbody>
</table>

*Table 1: XCCDF Rule Elements*
3.4.2 Profiles

XCCDF rules are logically organized into profiles. To query which profiles are contained within the SCAP Security Guide project, run the following command:

```
$ grep -n "<Profile" /usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml
72:  <Profile id="standard">
269: <Profile id="pci-dss">
520: <Profile id="rht-ccp">
773: <Profile id="common">
999: <Profile id="stig-rhel7-server-upstream">
```

If you were to view line 402 and those proceeding, you will see a sampling of which rules make up the PCI profile:

```
$ sed -n '269,275p' /usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml
  <Profile id="pci-dss">
    <description xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">This is a *draft* profile for PCI-DSS v3</description>
    <select idref="service_audittd_enabled" selected="true"/>
    <select idref="bootloader_auditt_argument" selected="true"/>
    <select idref="audittd_data_retention_num_logs" selected="true"/>
...
3.5 OVAL

OVAL is an information security community effort to standardize how to assess and report upon the machine state of computer systems. OVAL includes a language to encode system details, and an assortment of content repositories held throughout the community.

The OVAL component of SCAP can be a bit obtuse, and is arguably the most complex specification of the SCAP family. Instead of fully detailing here, we'll acknowledge its existence, and return to the topic when we begin authoring custom content.

The OVAL homepage is http://oval.mitre.org/
4 OPERATION

4.1 XCCDF Validation

Similar to validating HTML content through W3C, it’s always a good idea to validate your XCCDF content before utilizing it to perform a scan on your system. In a perfect world, by the time XCCDF content makes its way to end-users, this has already been done…. but perhaps we should make sure!

OpenSCAP contains the ability to validate a given XCCDF file for valid XML schema. Every found error is printed to the standard error. To quickly validate the XCCDF content, run the following command:

```
$ oscap xccdf validate /usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml
```

The output will show a return code, the reference table for which is:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Validation Successful</td>
</tr>
<tr>
<td>1</td>
<td>Validation Not Performed Due to Error</td>
</tr>
<tr>
<td>2</td>
<td>XCCDF Document is Invalid</td>
</tr>
</tbody>
</table>
4.2 Generate HTML Guide

In Chapter 3, you learned about logical groupings of individual XCCDF rules known as an XCCDF Profile. While users certainly can make their way through the XML files, an arguably more efficient way to review XCCDF content is to generate an HTML checklist. The OpenSCAP utility can transform your XCCDF into an HTML checklist through the “generate guide” option:

```
$ oscap xccdf generate guide --profile $profile \\n$xccdfContentPath $outputFile
```

For this workshop, generate an XCCDF Checklist for the `stig-rhel6-server-upstream` profile, placing the output in Apache's HTML directory:

```
$ oscap xccdf generate guide --profile stig-rhel6-server-upstream \\
/usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml \\
>/var/www/html/stig-rhel7-server-guide.html
```
Point your web browser to http://studentX.lab.openscap.io/stig-rhel7-server-guide.html

In the Table of Contents, click the “Rule Selection” link. The resulting HTML page will list all rules, indicating if they are “selected” or “not selected” for the stig-rhel7-server-upstream profile. Clicking on a rule title will expose full information on the rule.

<table>
<thead>
<tr>
<th>Profile Title</th>
<th>Pre-release Draft STIG for Red Hat Enterprise Linux 7 Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile ID</td>
<td>stig-rhel7-server-upstream</td>
</tr>
</tbody>
</table>

**Table of Contents**

1. **Introduction**
   1. General Principles
   2. How to Use This Guide

2. **System Settings**
   1. Installing and Maintaining Software
   2. File Permissions and Masks
   3. SELinux
   4. Account and Access Control
   5. Network Configuration and Firewalls
   6. Configure Syslog
   7. System Accounting with auditd

3. **Services**
   1. Obsolete Services
   2. Base Services
   3. Cron and At Daemons
   4. SSH Server
   5. X Window System
4.3 Performing a Scan

At this point you have a validated XCCDF file, and through the previous section, an understanding of which rules our system will be scanned against. Using the OpenSCAP utility provided through the operating system, our next step is to run a scan. There are several available options within the OpenSCAP utility. Sources of relevant documentation the following man pages:

- man oscap
- man scap-security-guide

Both documents detail sample commands, while the OpenSCAP manpage outlines all available arguments. For this workshop, we will use:

- **--profile**
  Mandatory; identifies which profile to scan against

- **--results**
  Optional; indicates location to place XML formatted results

- **--report**
  Optional, indicates location to place HTML formatted results

Putting everything together, our command looks like this:

```bash
$ sudo oscap xccdf eval --profile stig-rhel7-server-upstream \   --results /var/www/html/stig-results.xml \   --report /var/www/html/stig-report.html \   /usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml
```
4.4 Interpreting Results

4.4.1 HTML


   • Target Information

   ![Evaluation Characteristics Table]

   Evaluation Characteristics

<table>
<thead>
<tr>
<th>Target machine</th>
<th>devbox-rhel7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark URL</td>
<td>/usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml</td>
</tr>
<tr>
<td>Profile ID</td>
<td>stig-rhel7-server-upstream</td>
</tr>
<tr>
<td>Started at</td>
<td>2016-05-25T09:32:14</td>
</tr>
<tr>
<td>Finished at</td>
<td>2016-05-25T09:32:23</td>
</tr>
<tr>
<td>Performed by</td>
<td>shawnr</td>
</tr>
</tbody>
</table>

   ![CPE Platforms and Addresses]

   CPE Platforms
   - `cpe:/a/redhat:enterprise_linux:7`
   - `cpe:/a/redhat:enterprise_linux:7:client`

   Addresses
   - IPv4: 127.0.0.1
   - IPv4: 10.211.55.3
   - IPv4: 0.0.0.0/0.0.0.0
   - IPv6: fe80::2:1c:21cf:42ff:fe84:3983
   - IPv6: fe80::32:1c:21cf:42ff:fe84:3983
   - MAC: 00:00:00:00:00:00
   - MAC: 00:1c:42:84:39:83

   **Illustration 1: HTML Results Output: Target Information**

   This information will include hostname, network information, and relevant platform data. The CPE is of particular importance, as it calls out the host platform (RHEL, Fedora, etc).
• **Rule Results Summary**

**Compliance and Scoring**

The target system did not satisfy the conditions of 41 rules! Please review rule results and consider applying remediation.

**Rule results**

- 15 passed
- 41 failed
- 4 other

**Severity of failed rules**

- 35 low
- 8 medium

**Score**

<table>
<thead>
<tr>
<th>Scoring system</th>
<th>Score</th>
<th>Maximum</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>urn:xccdf:scoring:default</td>
<td>64.166672</td>
<td>100.000000</td>
<td>84.17%</td>
</tr>
</tbody>
</table>

*Illustration 2: HTML Results Output: Rule Results Summary*

• **Rule Results Table**
  
  Provides a per-rule review of pass/fail conditions, with HTML hyperlinks to specific rule results.

<table>
<thead>
<tr>
<th>Title</th>
<th>Severity</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guide to the Secure Configuration of Red Hat Enterprise Linux</td>
<td>41x fail</td>
<td></td>
</tr>
<tr>
<td>System Settings</td>
<td>4x not checked</td>
<td></td>
</tr>
<tr>
<td>Installing and Maintaining Software</td>
<td>2x fail</td>
<td></td>
</tr>
<tr>
<td>Disk Partitioning</td>
<td>2x fail</td>
<td>1x not checked</td>
</tr>
<tr>
<td>Ensure /var/log Located On Separate Partition</td>
<td>low</td>
<td>fail</td>
</tr>
<tr>
<td>Ensure /var/log/audit Located On Separate Partition</td>
<td>low</td>
<td>fail</td>
</tr>
<tr>
<td>Encrypt Partitions</td>
<td>low</td>
<td></td>
</tr>
<tr>
<td>Updating Software</td>
<td>1x not checked</td>
<td></td>
</tr>
<tr>
<td>Ensure Red Hat GPG Key Installed</td>
<td>high</td>
<td>pass</td>
</tr>
<tr>
<td>Ensure gpgcheck Enabled In Main Yum Configuration</td>
<td>high</td>
<td>pass</td>
</tr>
<tr>
<td>Ensure gpgcheck Enabled For All Yum Package Repositories</td>
<td>high</td>
<td>pass</td>
</tr>
<tr>
<td>Ensure Software Patches Installed</td>
<td>high</td>
<td>not checked</td>
</tr>
</tbody>
</table>

*Illustration 3: HTML Results Output: Rule Results Table*
• **Rule Results**
  - Provides detailed information on the rule, to include severity and CCE reference.

![Rule Results Table]

**Set Password Maximum Age**

<table>
<thead>
<tr>
<th>Rule ID</th>
<th>accounts_maximum_age_login_defs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>fail</td>
</tr>
<tr>
<td>Time</td>
<td>2016-05-25T09:32:14</td>
</tr>
<tr>
<td>Severity</td>
<td>medium</td>
</tr>
<tr>
<td>Identifiers and References</td>
<td>CCE-27051-2</td>
</tr>
</tbody>
</table>

**Description**
To specify password maximum age for new accounts, edit the file `/etc/login.defs` and add or correct the following line, replacing DAYS appropriately:

```
PASS_MAX_DAYS DAYS
```

A value of 180 days is sufficient for many environments. The DoD requirement is 60.

**Rationale**
Setting the password maximum age ensures users are required to periodically change their passwords. This could possibly decrease the utility of a stolen password. Requiring shorter password lifetimes increases the risk of users writing down the password in a convenient location subject to physical compromise.

**Remediation script:**
```
var_accounts_maximum_age_login_defs="60"
grep -q "^PASS_MAX_DAYS /etc/login.defs && 
  sed -i "s/PASS_MAX_DAYS.*/PASS_MAX_DAYS \$var_accounts_maximum_age_login_defs/g" /etc/login.defs
if ! [ $? = 0 ]; then
  echo "PASS_MAX_DAYS \$var_accounts_maximum_age_login_defs" >> /etc/login.defs
fi
```

*Illustration 4: HTML Results Output: Rule Detail*
4.4.2 XML

The OpenSCAP utility can output results in XML. Load studentX-results.xml:

$ vim /var/www/html/stig-results.xml

The results are contained within <rule-result> tags. Issue a search query in vi:

//<rule-result

You will be brought to XML stanzas similar to:

```xml
<rule-result idref="package_aide_installed" time="2014-07-28T17:04:53" severity="medium" weight="1.000000">
  <result>pass</result>
  <ident system="http://cce.mitre.org">CCE-27024-9</ident>
  <fix xmlns:xhtml="http://www.w3.org/1999/xhtml" system="urn:xccdf:fix:script:sh">
    if ! rpm -qa | grep -q aide; then
      yum -y install aide
    fi
  </fix>
  <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
    <check-content-ref name="oval:ssg:def:1331" href="ssg-rhel6-oval.xml"/>
  </check>
</rule-result>
```
The XML above can be parsed as follows:

<table>
<thead>
<tr>
<th>XML Tag</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rule-result idref....&gt;</td>
<td>Identifies which XCCDF rule the results reflect</td>
</tr>
<tr>
<td>&lt;result&gt;</td>
<td>Pass/Fail/NotApplicable</td>
</tr>
<tr>
<td>&lt;ident system....&gt;</td>
<td>Identifies corresponding CCE</td>
</tr>
<tr>
<td>&lt;fix&gt;</td>
<td>Identifies remediation actions, in bash, to perform</td>
</tr>
<tr>
<td>&lt;check system....&gt;</td>
<td>Identifies which version of OVAL the check content was written against</td>
</tr>
<tr>
<td>&lt;check-content-ref .......&gt;</td>
<td>Corresponding OVAL check name (name=....) and source OVAL file (href=....)</td>
</tr>
</tbody>
</table>

*Table 2: Scan Results XML Output*
4.5 Remediation

SCAP Security Guide contains remediation capabilities for RHEL 7. These exist in the form of bash scripts. In this section, we will explore the SSG's remediation scripts and how they are applied to aide in system hardening.

4.5.1 Remediation? How Do?

Earlier in this workshop we ran a scan, and created an XML output file at /var/www/html/stig-results.xml. Let's quickly review that file again:

$ vim /var/www/html/stig-results.xml

Once loaded, let's review the remediation for disable_prelink:

:1920

You will be brought to XML stanzas similar to:

```
<fix xmlns:xhtml="http://www.w3.org/1999/xhtml" system="urn:xccdf:fix:script:sh">#
# Disable prelinking altogether
#
if grep -q ^PRELINKING /etc/sysconfig/prelink
then
  sed -i 's/PRELINKING.*/PRELINKING=no/g' /etc/sysconfig/prelink
else
  echo -e "
  # Set PRELINKING=no per security requirements"
  /etc/sysconfig/prelink
  echo "PRELINKING=no" /
fi
#
# Undo previous prelink changes to binaries
#
/usr/sbin/prelink -ua
</fix>
```

OpenSCAP has the ability to transform the <fix> section into an executable script for any rules which failed. To generate this fix script, run the following from the command-line:

$ oscap xccdf generate fix \\--result-id xccdf_org.open-scap_testresult_stig-rhel7-server-upstream \\/var/www/html/stig-results.xml
$ oscap xccdf generate fix \
-result-id xccdf_org.open-scap_testresult_stig-rhel6-server-upstream \
/var/www/html/studentX-results.xml

#!/bin/bash
# OpenSCAP fix generator output for benchmark: DRAFT Guide
# to the Secure Configuration of Red Hat Enterprise Linux 7

# Generating fixes for all failed rules in test result
#'xccdf_org.open-scap_testresult_stig-rhel7-server-upstream'.

# XCCDF rule: accounts_minimum_age_login_defs
# CCE-27002-5
var_accounts_minimum_age_login_defs="1"
grep -q ^PASS_MIN_DAYS /etc/login.defs && \
    sed -i "s/PASS_MIN_DAYS.*/PASS_MIN_DAYS     $var_accounts_minimum_age_login_defs/g" /etc/login.defs
if ! [ $? -eq 0 ]; then
    echo "PASS_MIN_DAYS      $var_accounts_minimum_age_login_defs" >> /etc/login.defs
fi

# XCCDF rule: accounts_maximum_age_login_defs
# CCE-27051-2
var_accounts_maximum_age_login_defs="60"
grep -q ^PASS_MAX_DAYS /etc/login.defs && \
    sed -i "s/PASS_MAX_DAYS.*/PASS_MAX_DAYS     $var_accounts_maximum_age_login_defs/g" /etc/login.defs
if ! [ $? -eq 0 ]; then
    echo "PASS_MAX_DAYS      $var_accounts_maximum_age_login_defs" >> /etc/login.defs
fi

This output could be redirected to a bash script, or build into your RHEL7 provisioning process (e.g. the %post section of a kickstart).
4.6 Scanning Automation

Scanning can be automated/scheduled via cron. To schedule a daily scan of your system, place the following script into /etc/cron.daily/ssg-scan

```bash
oscap xccdf eval --profile stig-rhel7-server-upstream \
--results /var/www/html/stig-results.xml \
--report /var/www/html/stig-report.xml \
/usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml
```

*Note:* The XML results and HTML reports contain security information of your system. When creating reports on a production system, particularly those within the DoD and Intelligence Community, ensure your output is placed into a restricted location on the system. The `/root` directory is often ideal.
4.7 Alternative Tools

4.7.1 SPAWAR SCAP Compliance Checker (SPAWAR SCC)

The SCAP Compliance Checker (SCC) is a Security Content Automation Protocol (SCAP) scanner developed by Space and Naval Warfare (SPAWAR) Systems Center Atlantic.

The SCC homepage is located at:
http://www.public.navy.mil/spawar/Atlantic/ProductsServices/Pages/SCAP.aspx

4.7.2 Red Hat Network Satellite

Starting with RHN Satellite >= 5.5, users can now centralize their SCAP scanning through their Satellite administrative console (GUI) or via the Satellite API.

This capability is documented at:

A video presentation of this integration is available on YouTube:
https://www.youtube.com/watch?v=qGYWO2THPXMy - Security Compliance by OpenSCAP