SCAP & STIG Workshop

March 2013

Table of Contents

Illustration Index

Index of Tables

Document v1.0
Shawn Wells | shawn@redhat.com

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, utilizing the DISA STIG baseline. Upon completion of this workshop, you should be able to return to your shop, obtain the SCAP components from Red Hat and DISA FSO, and be able to perform highly automated STIG scanning and reporting.

Each attendee will be given a virtual machine to use for this workshop. The hostnames follow the syntax of:

studentX.scapworkshop.com

The initial login will be studentX, with a password of "password," with the root password being "redhat".

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 instructor(s) 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 6 (RHEL6) and JBoss Enterprise Application Server 5 (JBoss EAP5). Other technologies are planned, such as KVM and Red Hat Storage, in CY2013.

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://fedorahosted.org/scap-security-guide/

2.1 Background

Security guidance documents, such as the NSA RHEL5 SNAC Guide, have historically been authored in archaic static formats such as PDF files. This left much to be desired, particularly around automation, and individual shops were forced to create their own scripted hardening processes. These processes were authored in their own silos, by unique contractor teams, generally not shared across programs, and caused gross waste of tax payer dollars. Additionally, from a vendor perspective, supporting hardened environments was commercially unfeasible. As each shop held it's 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 which was 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 (arguably) has produced higher-quality content that has now become official U.S. Government standards.

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

The SCAP Security Guide content was developed on, and for, Red Hat Enterprise Linux 6. The content likely will not work on prior releases, Fedora, or CentOS, without modification. This workshop assumes you have access to a RHEL6 machine. Please note latest download instructions will always be found at the SSG homepage, located here:

https://fedorahosted.org/scap-security-guide

1. To enable the SSG EPEL repository, you must download the yum .repo file for the project:

```
$ wget -O ~/epel-6-scap-security-guide.repo \
http://repos.fedorapeople.org/repos/scap-security-guide/epel-6-scap-
security-guide.repo

$ sudo sh -c “mv ~studentX/epel-6-scap-security-guide.repo \
/etc/yum.repos.d/”
```

2. Next, install SSG via yum:

```
$ sudo sh -c “yum install scap-security-guide”
```

Please note this may pull down other dependencies, such as the OpenSCAP tooling, if they are not part of your current baseline.

3 UNDERSTANDING THE COMPONENTS

3.1 Content Location

On RHEL6, SCAP content is stored under `/usr/share/xml/scap/`, with each content provider responsible for creating their own subdirectories. The SSG project deploys content into `/usr/share/xml/scap/ssg/` with the following sub directories:

- `content/`
  Houses SCAP content utilizing the following naming conventions:
  - CPE Dictionaries
    `ssg-{profile}-cpe-dictionary.xml`
  - CPE OVAL Content
    `ssg-{profile}-cpe-oval.xml`
  - OVAL Content
    `ssg-{profile}-oval.xml`
  - XCCDF Content
    `ssg-{profile}-xccdf.xml`
- `guides/`
  HTML versions of SSG profiles, easily consumable by humans.
- `policytables/`
  HTML tables reflecting which institutionalized policy a particular SSG rule conforms to.

3.2 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 tools 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/page/Main_Page.

3.3 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/`.

3.3.1 Sample XCCDF Code

To understand XCCDF, let's review the `disable_telnet_service` rule within `ssg-rhel6-xccdf.xml`.

```
$ vim /usr/share/xml/scap/ssg/content/ssg-rhel6-xccdf.xml
```

Once loaded, locate the disable_telnet_service rule:

```
/<Rule id=”disable_telnet_service
```

The code can be broken into the following parts:

XML Tag	Meaning
<Rule>	Indicates: • Rule Name … id=”disable_telnet_service” …. • Severity … severity=”high” … • If the rule is enforced / selected …. selected=”true” …
<title>	The XCCDF Rule title
<description>	Largely free form, contains implementation instructions and any policy references
<rationale>	Identifies the logic behind applying this <rule> to the system. Answers “why is this important?”
<ident>	Identifies CCE mapping
<check system....>	Identifies which version of OVAL the check was written for
<check-content-ref....>	There are two elements for this tag: • href=..... Path to associated OVAL file • name=..... Identifies the corresponding OVAL rule name, such as oval:ssg:def:281. If one were to search for this rule within the href= identified file, they would be brought to the corresponding OVAL code.

Table 1: XCCDF Rule Elements

3.3.2 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-rhel6-xccdf.xml
12:  <Profile id="test">
68:  <Profile id="common">
345:  <Profile id="desktop" extends="common">
381:  <Profile id="server" extends="common">
389:  <Profile id="ftp" extends="server">
403:  <Profile id="stig-rhel6-server" extends="common">
```

If you were to view line 403 and those proceeding, you will see a sampling of which rules make up the stig-rhel6-server profile:

```
$ sed -n '403,413p' /usr/share/xml/scap/ssg/content/ssg-rhel6-xccdf.xml
  <Profile id="stig-rhel6-server" extends="common">
    <title>Pre-release Draft STIG for RHEL 6 Server</title>
    <description>This profile is being developed under the DoD consensus
model to become a STIG in coordination with DISA FSO.</description>
    <select idref="encrypt_partitions" selected="true"/>
    <select idref="rpm_verify_permissions" selected="true"/>
    <select idref="rpm_verify_hashes" selected="true"/>
    <select idref="world_writeable_files" selected="true"/>
    <select idref="install_antivirus" selected="true"/>
    <select idref="install_hids" selected="true"/>
    <select idref="disable_ctrlaltdel_reboot" selected="true"/>
    <select idref="service_postfix_enable" selected="true"/>
```

3.4 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.

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. Theoretically, by the time XCCDF content makes its way to end-users, this has already been done.... however, why not 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. Return code 0 if validation succeeds, 1 if validation could not be performed due to some error, 2 if the XCCDF document is not valid. To quickly validate the XCCDF content, run the following command:

```
$ oscap xccdf validate /usr/share/xml/scap/ssg/content/ssg-rhel6-xccdf.xml
```

The command should return with no output (return code 0). If you ever do receive an error, please open a trouble ticket via the SSG homepage as something is broke!

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 \
$xccdfContentPath $outputFile
```

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

```
$ oscap xccdf generate guide --profile stig-rhel6-server \
/usr/share/xml/scap/ssg/content/ssg-rhel6-xccdf.xml \
>/var/www/html/stig-rhel6-server-guide.html
```

Point your web browser to http://studentX/stig-rhel6-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-rhel6-server profile. Clicking on a rule title will expose full information on the rule.

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

- `--cpe`
  Mandatory, identifies location of CPE dictionary

Putting everything together, our command looks like this:

```
# oscap xccdf eval --profile stig-rhel6-server \
--results /var/www/html/`hostname`-results.xml \
--report /var/www/html/`hostname`-report.html \
--cpe /usr/share/xml/scap/ssg/content/ssg-rhel6-cpe-dictionary.xml \
/usr/share/xml/scap/ssg/content/ssg-rhel6-xccdf.xml
```

4.4 Interpreting Results

4.4.1 HTML

1. Using a web browser, load http://studentX/studentX-report.html

- *Target Information*

Target info

Targets	Addresses	Platforms
• rhel6	• 127.0.0.1 • 10.211.55.8 • ::1 • fdb2:2c26:f4e4:0:21c:42ff:fea6:7216 • fe80::21c:42ff:fea6:7216	• cpe:/o:redhat:enterprise_linux:6 • cpe:/o:redhat:enterprise_linux:6::client

Illustration 1: HTML Results Output: Target Information

This information will include hostname, network information, and relevant platform data.

- *Rule Results Summary*

Rule Results Summary

pass	fixed	fail	error	not selected	not checked	not applicable	informational	unknown	total
85	0	101	1	189	33	0	0	3	412

Illustration 2: HTML Results Output: Rule Results Summary

This section provides an overview of scanning results.

- *Rule Results Table*

Title	Result
Ensure /tmp Located On Separate Partition	fail
Ensure /var Located On Separate Partition	fail
Ensure /var/log Located On Separate Partition	fail
Ensure /var/log/audit Located On Separate Partition	fail
Ensure /home Located On Separate Partition	fail
Encrypt Partitions	notchecked
Ensure Red Hat GPG Key Installed	pass
Ensure gpgcheck Enabled In Main Yum Configuration	pass

Illustration 3: HTML Results Output: Rule Results Table

Provides a per-rule review of pass/fail conditions, with HTML hyperlinks to specific rule results.

- *Rule Results*

Result for Ensure gpgcheck Enabled In Main Yum Configuration

Result: **pass**

Rule ID: **ensure_gpgcheck_globally_activated**

Time: **2013-03-23 13:43**

Severity: **high**

The `gpgcheck` option should be used to ensure checking of an RPM package's signature always occurs prior to its installation. To configure yum to check package signatures before installing them, ensure the following line appears in `/etc/yum.conf` in the `[main]` section:

```
gpgcheck=1
```

Ensuring the validity of packages' cryptographic signatures prior to installation ensures the provenance of the software and protects against malicious tampering.

Security identifiers

- CCE-26709-6

results overview

Illustration 4: HTML Results Output: Rule Detail

Provides detailed information on the rule, to include severity and CCE reference.

4.4.2 XML

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

```
$ vim /var/www/html/studentX-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:

```
<rule-result idref="install_aide" time="2013-03-23T13:43:32"
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:

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

Table 2: Scan Results XML Output

4.5 Remediation

The current version of the SCAP Security Guide (scap-security-guide-0.1-10) has limited remediation capabilities, which are planned to be extended through Spring CY2013. To show the community where we're headed, let's explore current capabilities.

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

```
$ vim /var/www/html/studentX-results.xml
```

Once loaded, search for the install_aide result:

```
/<rule-result idref="install_aide
```

You will be brought to XML stanzas similar to:

```
<rule-result idref="install_aide" time="2013-03-23T13:43:32"
severity="medium" weight="1.000000">

  <result>fail</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>
```

OpenSCAP contains 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-rhel6-server \
/var/www/html/studentX-results.xml
```

You will receive output similar to the following:

```
$ oscap xccdf generate fix \
–result-id xccdf_org.open-scap_testresult_stig-rhel6-server \
/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 6


# Generating fixes for all failed rules in test result
#'xccdf_org.open-scap_testresult_stig-rhel6-server'.

# XCCDF rule: install_aide
# CCE-27024-9

if ! rpm -qa | grep -q aide; then
yum -y install aide
fi


# XCCDF rule: uninstall_xinetd
# CCE-27005-8

if rpm -qa | grep -q xinetd; then
yum -y remove xinetd
fi


# XCCDF rule: uninstall_ypserv
# CCE-27079-3

if rpm -qa | grep -q ypserv; then
yum -y remove ypserv
fi


# generated: 2013-03-23T16:32:50-04:00
# END OF SCRIPT
```

This output could be redirected to a bash script, or build into your RHEL6 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

```
oscap xccdf eval --profile stig-rhel6-server \
--results /var/www/html/`date +%F-%R`-`hostname`-results.xml \
--report /var/www/html/`date +%F-%R`-`hostname`-report.xml \
--cpe /usr/share/xml/scap/ssg/content/ssg-rhel6-cpe-dictionary.xml \
/usr/share/xml/scap/ssg/content/ssg-rhel6-xccdf.xml
```

Note: The XML results and HTML reports contain security information of your system. When creating reports on a production system, particularly those of governments, 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. As of March 2013, SPAWAR SCC v3.1 does not the SCAP Security Guide or RHEL6 STIG content.

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 centralized their SCAP scanning through their Satellite administrative console (GUI) or via the Satellite API.

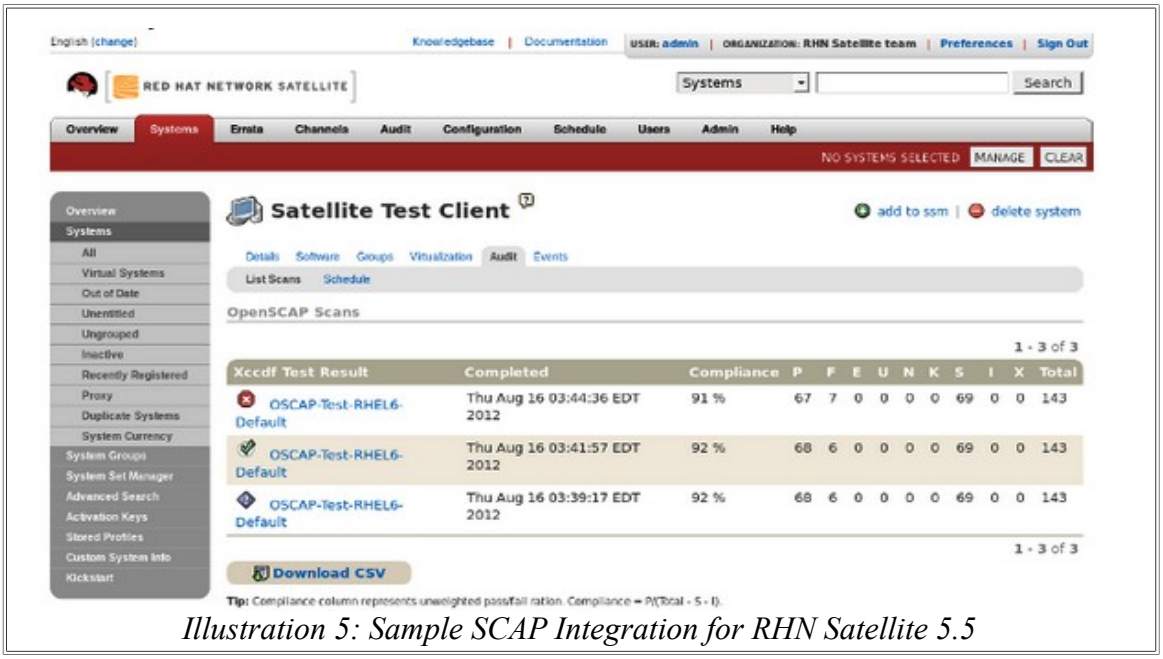


Illustration 5: Sample SCAP Integration for RHN Satellite 5.5

This capability is documented at:

https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Network_Satellite/5.5/html/User_Guide/sect-Red_Hat_Network_Satellite-User_Guide-OpenSCAP-OpenSCAP_in_RHN_Satellite.html

5 CONTENT CUSTOMIZATION

5.1 So, you wanna be a developer?

Welcome! Making changes to the project requires posting a patch to the mailing list, so that it can be vetted. Once there, another commit-level project member must issue acknowledgement ("ACK") to accept it, and then it can be pushed. Assuming another project member has not issued a NACK in protest first, that is! The following instructions assume familiarity with git and git-send-email, but project members are happy to provide tips if you encounter any roadblocks.

To properly join the project you must first establish a few required accounts:

1. Join the mailing list, it's how developers and users communicate:

   https://fedorahosted.org/mailman/listinfo/scap-security-guide

2. Create your FedoraHosted account. It formally registers you as a developer, and allows you to upload your public SSH key (required for commits).

   https://admin.fedoraproject.org/accounts/

3. Once your FedoraHosted account is created, request membership to the SCAP Security Guide project:

   https://admin.fedoraproject.org/accounts/group/view/gitscap-security-guide

4. Lastly, review a few coding standards:

   * How to Create a New Guidance Item in XCCDF

   * How to Create a New Compliance Check in OVAL

5.2 Cloning the Source

If you've been given commit-level access, you will interface with the git repository over SSH. Change directory to where you'd like the source code to be placed (e.g. `cd ~/MyProjects/`) and run the following command:

NOTE: For this workshop, use /var/www/html/

```
$ cd /var/www/html/
$ git clone ssh://git.fedorahosted.org/git/scap-security-guide.git
```

If you have not been given commit access, use the standard HTTP interface:

```
$ git clone ssh://git.fedorahosted.org/git/scap-security-guide.git
```

5.3 Understanding the Code Tree

You've now cloned the projects source code. A new directory, `scap-security-guide`, was created where you ran the clone command. Change directory into it and perform a directory listing:

```
$ cd scap-security-guide; ls -l

total 36
drwxrwxr-x. 4 shawn shawn 4096 Mar 14 20:51 JBossEAP5
-rw-rw-r--. 1 shawn shawn  409 Mar 14 20:51 LICENSE
-rw-rw-r--. 1 shawn shawn 2945 Mar 17 18:58 Makefile
drwxrwxr-x. 8 shawn shawn 4096 Mar 17 14:03 OpenStack
-rw-rw-r--. 1 shawn shawn  840 Mar 14 20:51 README
drwxrwxr-x. 8 shawn shawn 4096 Mar 23 14:34 RHEL6
drwxrwxr-x. 8 shawn shawn 4096 Mar 17 14:03 RHEVM3
drwxrwxr-x. 4 shawn shawn 4096 Mar 23 11:32 rpmbuild
-rw-rw-r--. 1 shawn shawn 3229 Mar 14 20:51 scap-security-guide.spec
```

Top level directories have been created to contain the per-technology SCAP content. Change directory into `RHEL6` and perform a directory listing:

```
$ cd RHEL6; ls -l
total 40
drwxrwxr-x. 2 shawn shawn  4096 Mar 23 17:35 dist
drwxrwxr-x. 9 shawn shawn  4096 Mar 21 18:57 input
-rw-rw-r--. 1 shawn shawn 10277 Mar 14 20:51 Makefile
drwxrwxr-x. 2 shawn shawn  4096 Mar 23 17:35 output
-rw-rw-r--. 1 shawn shawn  1616 Mar 14 20:51 README
drwxrwxr-x. 2 shawn shawn  4096 Mar 17 18:57 references
drwxrwxr-x. 2 shawn shawn  4096 Mar 17 14:03 transforms
drwxrwxr-x. 2 shawn shawn  4096 Mar 14 20:51 utils
```

The directory usages are:

Directory	Usage
dist/	The build process generates finalized content here, which then are included into SSG RPMs.
input/	Source files that generate SCAP content, such as XCCDF and OVAL. Since a single large XML file is an impractical format for multiple authors to collaborate on editing SCAP content, efforts are made to keep logically related guidance and checking content in individual files.
output/	Used as a storage area for items generated by the files in the inputs directory. It should be empty in the repository, and built on users' individual systems (and rely on its .gitignore file to keep such files out). The output directory contains transitional output (which may only exist in order to be further transformed) as well as final output.
references/	Contain documents which are specified as references from within the SCAP content, or documents that are "seeds," viz. documents whose prose will be translated into SCAP formats, as well as other examples of SCAP content.
transforms/	Resources that enable the files inside the input directory (or output directory) to be combined and reformatted into valid SCAP formats or human-readable formats.

Table 3: SSG Code Tree: RHEL6/

5.4 Creating your first XCCDF rule

For this workshop we will create a rule which mandates installation of the SCAP Security Guide RPM, and we'll identify failure (the lack of installation) as a CAT I / SEV 1 finding.

The directory structure, and multiple XML files, can be a bit overwhelming at first. Change directory to `input/system/` and perform a directory listing:

```
$ cd input/system/; ls -l

total 116
drwxrwxr-x. 3 shawn shawn  4096 Mar 20 19:06 accounts
-rw-rw-r--. 1 shawn shawn 64882 Mar 20 19:29 auditing.xml
-rw-rw-r--. 1 shawn shawn 20365 Mar 14 20:51 logging.xml
drwxrwxr-x. 2 shawn shawn  4096 Mar 14 20:51 network
drwxrwxr-x. 2 shawn shawn  4096 Mar 21 06:47 permissions
-rw-rw-r--. 1 shawn shawn 10760 Mar 14 20:51 selinux.xml
drwxrwxr-x. 2 shawn shawn  4096 Mar 23 18:06 software
-rw-rw-r--. 1 shawn shawn    62 Mar 14 20:51 system.xml
```

To aid with multi-author content creation, the SSG project creates compartmented XCCDF files. These individual files are merged together during our Make/compilation process (more on that later!).

In the directory listing above you'll notice `auditing.xml`, `logging.xml`, and a few directories. Since we're creating a rule that deals with software, change directory into `software/` and perform a directory listing:

```
$ cd software/; ls -l
total 36
-rw-rw-r--. 1 shawn shawn  7307 Mar 14 20:51 disk_partitioning.xml
-rw-rw-r--. 1 shawn shawn 12458 Mar 17 18:57 integrity.xml
-rw-rw-r--. 1 shawn shawn   274 Mar 14 20:51 software.xml
-rw-rw-r--. 1 shawn shawn  5635 Mar 14 20:51 updating.xml
```

As the SSG project relates to security, lets create the new rule within the `integrity.xml` section. Load that file in your favorite text editor, then place yourself one line above EOF:

```
$ vim integrity.xml
{shift+g}
-
```

Your screen should be similar to the following:

```
<ref nist="SC-28, SI-3" disa="1239,1668"/>
</Rule>

</Group>
</Group>
~
                                              276,0-1       Bot
```

Illustration 6: integrity.xml File

The template for SSG XCCDF rules is below. Insert the following template into integrity.xml:

```
<Rule id="" severity="">
  <title></title>
  <description>
  </description>
  <ocil clause="">
    <package-check-macro package="" />
  </ocil>
  <rational>
  </rational>
  <oval id="" />
</Rule>
```

Using the template above, create a rule which:

1. Has an XCCDF rule id of “package_scap-security-guide_installed” with a severity of “high”
2. Has a human readable title of 'Install SCAP Security Guide”
3. Outlines a method to install SSG. For example, “yum install scap-security-guide”
4. States that “if SCAP Security Guide is not installed” this is a finding
5. Includes the proper package name, scap-security-guide, in the package check macro
6. Includes rational on why the SSG project is awesome, and should be installed
7. Corresponds to a (currently non-existent) OVAL rule named “package_scap-security-guide_installed”

Your completed template will look similar to:

```
<Rule id="package_scap-security-guide_installed" severity="high">
  <title>Install SCAP Security Guide</title>
  <description>The SCAP Security Guide may be installed by first enabling the SSG
  YUM repository. This can be accomplished by running:
  wget -O /etc/yum.repos.d/epel-6-scap-security-guide.repo \
  http://repos.fedorapeople.org/repos/scap-security-guide/epel-6-scap-security-guide.repo

  Once the EPEL repository has been installed, install SSG through the following command:
  $ sudo sh -c "yum install scap-security-guide"
  </description>
  <ocil clause="SCAP Security Guide is not installed">
    <package-check-macro package="scap-security-guide" />
  </ocil>
  <rational>
  I'm only creating this because the instructor told me to, but SSG is awesome and
  everyone should use it.
  </rational>
  <oval id="package_scap-security-guide_installed" />
</Rule>
```

Illustration 7: Sample must_install_ssg XCCDF Rule

Done! Hopefully that wasn't to painful. If you're curious on where the “package-check-macro” comes from, check out `RHEL6/transforms/shorthand2xccdf.xslt` and search for lines that begin with “`<xsl:template match=""`”

The shorthand2xccdf.xslt file contains many short-hand macros that are available, which inserts template text into final XCCDF output. Unfortunately, in a two hour workshop, we don't have enough time to properly cover all embedded XSLT transformations within the SSG. Feel free to direct questions to the public mailing list!

Now that the XCCDF language is written, let's see how it looks in the HTML guide. For this we will need to run a quick SSG compilation:

```
$ cd /var/www/html/scap-security-guide/RHEL6
$ make content
```

To ensure your XCCDF is still SCAP compliant, run a quick “make validate”:

```
$ make validate
oscap xccdf validate-xml output/ssg-rhel6-xccdf.xml
oscap oval validate-xml output/ssg-rhel6-oval.xml
oscap oval validate-xml output/ssg-rhel6-cpe-oval.xml
```

As mentioned earlier, the `output/` directory contains artifacts from the build. Using a web browser, view `http://studentX/scap-security-guide/output/rhel6-guide.html`. You'll notice your XCCDF Rule Title is now listed in the table of contents:

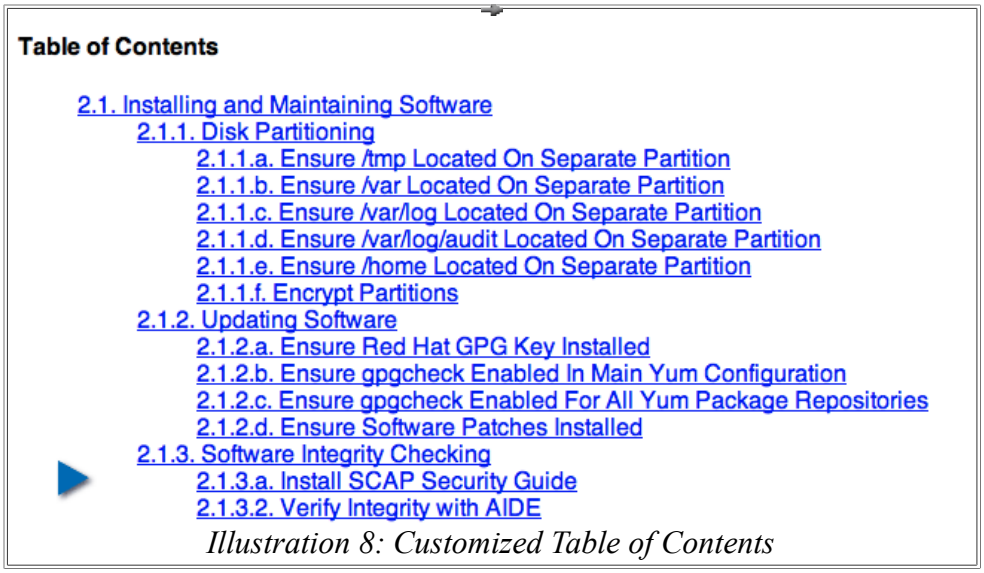
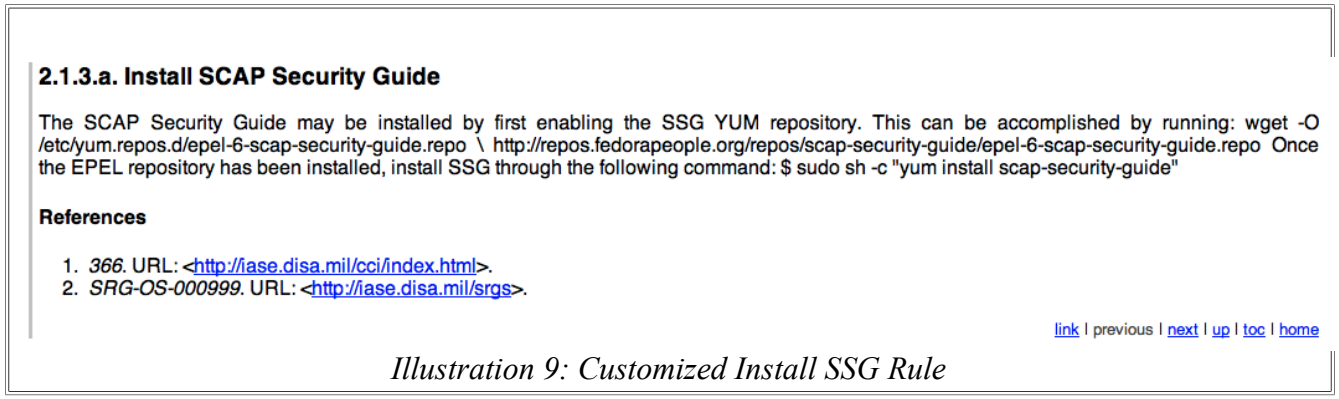


Illustration 8: Customized Table of Contents

Click on the “Install SCAP Security Guide” link, and you'll be brought to the newly created rule:



2.1.3.a. Install SCAP Security Guide

The SCAP Security Guide may be installed by first enabling the SSG YUM repository. This can be accomplished by running: wget -O /etc/yum.repos.d/epel-6-scap-security-guide.repo \ http://repos.fedorapeople.org/repos/scap-security-guide/epel-6-scap-security-guide.repo Once the EPEL repository has been installed, install SSG through the following command: $ sudo sh -c "yum install scap-security-guide"

References

1. *366*. URL: <http://iase.disa.mil/cci/index.html>.
2. *SRG-OS-000999*. URL: <http://iase.disa.mil/srgs>.

link | previous | next | up | toc | home

Illustration 9: Customized Install SSG Rule

You'll notice a few things:

1. The rendering from your <description> tag isn't very pretty, and line returns didn't keep.
2. References against DISA CCI and DISA's OS SRG were automatically added. When rules are not specifically mapped to a CCI or OS SRG, these “catch all” placeholders are inserted.

The `<description>` tag has the ability to handle XHTML arguments. Let's wrap our sample commands in `<pre>` tags, and re-run the build. Your XCCDF rule should now look like this:

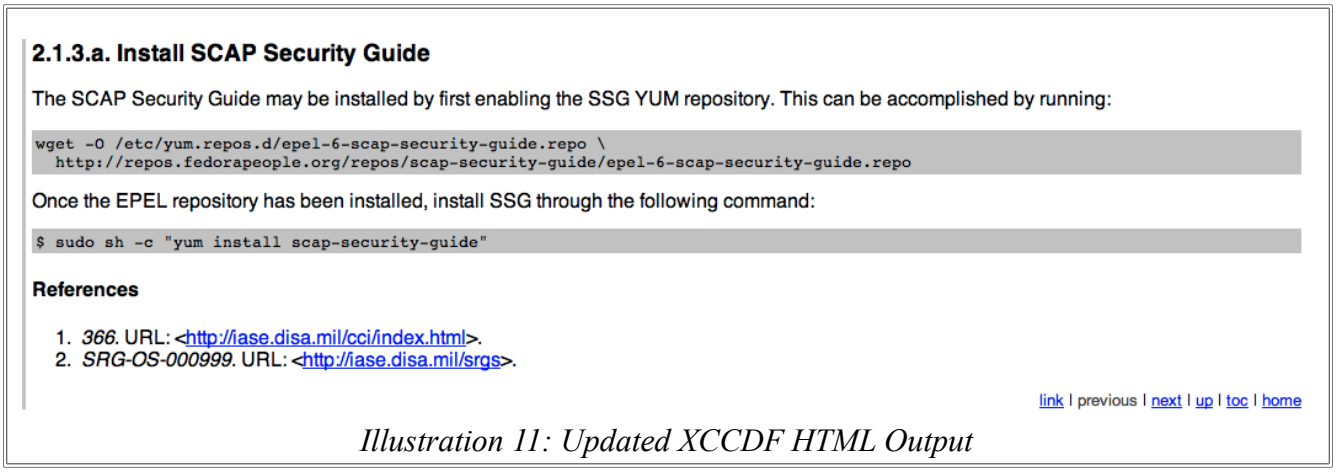
```
<Rule id="package_scap-security-guide_installed" severity="high">
  <title>Install SCAP Security Guide</title>
  <description>The SCAP Security Guide may be installed by first enabling the SSG
  YUM repository. This can be accomplished by running:
  <pre>wget -O /etc/yum.repos.d/epel-6-scap-security-guide.repo \
  http://repos.fedorapeople.org/repos/scap-security-guide/epel-6-scap-security-guide.repo</pre>

  Once the EPEL repository has been installed, install SSG through the following command:
  <pre>$ sudo sh -c "yum install scap-security-guide"</pre>
  </description>
  <ocil clause="SCAP Security Guide is not installed">
    <package-check-macro package="scap-security-guide" />
  </ocil>
  <rational>
  I'm only creating this because the instructor told me to, but SSG is awesome and
  everyone should use it.
  </rational>
  <oval id="package_scap-security-guide_installed" />
</Rule>
```

Illustration 10: XCCDF with XHTML <pre> Tags

Once updated, re-run the build:

```
$ make clean; make content; make validate
```

Upon completion, refresh your web browser to see the updated content:



2.1.3.a. Install SCAP Security Guide

The SCAP Security Guide may be installed by first enabling the SSG YUM repository. This can be accomplished by running:

```
wget -O /etc/yum.repos.d/epel-6-scap-security-guide.repo \
  http://repos.fedorapeople.org/repos/scap-security-guide/epel-6-scap-security-guide.repo
```

Once the EPEL repository has been installed, install SSG through the following command:

```
$ sudo sh -c "yum install scap-security-guide"
```

References

1. *366*. URL: <http://iase.disa.mil/cci/index.html>.
2. *SRG-OS-000999*. URL: <http://iase.disa.mil/srgs>.

link | previous | next | up | toc | home

Illustration 11: Updated XCCDF HTML Output

This looks much better. At this point we have a valid, functioning, XCCDF rule!

Now, onto OVAL content creation.

5.5 OVAL Authoring

OVAL standardizes the assessment and reporting of machine state. It's very comprehensive, with capabilities to examine boot-time and run-time configuration. MITRE has documented OVAL's built-in functions here:

http://oval.mitre.org/language/version5.10.1/ovaldefinition/documentation/linux-definitions-schema.html

The SSG project maintains all OVAL code under `RHEL6/input/checks/`, and provides a template utilities in `RHEL6/input/checks/templates/`. Change directories to `templates/` and perform a directory listing:

```
$ cd /var/www/html/scap-security-guide/RHEL6/input/checks/templates/; ls
create_kernel_modules_disabled.py  packages_removed.csv
create_package_installed.py        README
create_package_removed.py          services_disabled.csv
create_permission_checks.py        services_enabled.csv
create_services_disabled.py        sysctl_values.csv
create_services_enabled.py         template_kernel_module_disabled
create_sysctl_checks.py            template_package_installed
file_dir_permissions.csv           template_package_removed
find_untemplated.py                template_permissions
kernel_modules_disabled.csv        template_service_disabled
Makefile                           template_service_enabled
output                             template_sysctl
packages_installed.csv
```

Before continuing to the next page, take a minute to review the README file. What is the process to create a template for checking if `scap-security-guide` is installed?

As noted in the README file, several CSV files are located within the `templates/` directory. To automate the OVAL content:

1. Add scap-security-guide to the listing in packages_installed.csv:

```
$ echo “scap-security-guide” >> packages_installed.csv
```

2. Run “make templates”:

```
$ make templates
```

3. This process generated `output/package_scap-security-guide_installed.xml`. Load this file in a text editor for human-review:

```
$ vim output/package_scap-security-guide_installed.xml
```

The newly created template:

```
<def-group>
 <!-- THIS FILE IS GENERATED by create_package_installed.py.  DO NOT EDIT.  -->
  <definition class="compliance" id="package_scap-security-guide_installed"
  version="1">
    <metadata>
      <title>Package scap-security-guide Installed</title>
      <affected family="unix">
        <platform>Red Hat Enterprise Linux 6</platform>
      </affected>
      <description>The RPM package scap-security-guide should be installed.</description>
    </metadata>
    <criteria>
      <criterion comment="package scap-security-guide is installed"
      test_ref="test_package_scap-security-guide_installed" />
    </criteria>
  </definition>
  <linux:rpminfo_test check="all" check_existence="all_exist"
  id="test_package_scap-security-guide_installed" version="1"
  comment="package scap-security-guide is installed">
    <linux:object object_ref="obj_package_scap-security-guide" />
  </linux:rpminfo_test>
  <linux:rpminfo_object id="obj_package_scap-security-guide" version="1">
    <linux:name>scap-security-guide</linux:name>
  </linux:rpminfo_object>
</def-group>
```

Illustration 12: package_scap-security-guide_installed.xml

OVAL contains many pre-defined functions. In this case, we make use of `<linux:rpminfo_test>` to check for the installation of `scap-security-guide`.

4. Run “make copy” to place `package_scap-security-guide_installed.xml` into the project:

```
$ make copy
```

5. Done! You've now added an OVAL rule to check for the existence of `scap-security-guide`!

5.6 Profiles

With our XCCDF rule and OVAL content created, we must now add the rule to an XCCDF profile. Let's add this as a STIG requirement, placing it into the `stig-rhel6-server` profile.

XCCDF profiles are retained within `RHEL6/input/profiles/`. Change directory and perform a directory listing to see available profiles:

```
$ cd /var/www/html/scap-security-guide/RHEL6/input/profiles/; ls -l
total 96
-rw-rw-r--. 1 shawn shawn 16798 Mar 14 20:51 common.xml
-rw-rw-r--. 1 shawn shawn  1957 Mar 14 20:51 desktop.xml
-rw-rw-r--. 1 shawn shawn   800 Mar 14 20:51 ftp.xml
-rw-rw-r--. 1 shawn shawn  2527 Mar 14 20:51 manual_audits.xml
-rw-rw-r--. 1 shawn shawn  1902 Mar 14 20:51 manual_remediation.xml
-rw-rw-r--. 1 shawn shawn 21629 Mar 14 20:51 nist-CL-IL-AL.xml
-rw-rw-r--. 1 shawn shawn   448 Mar 14 20:51 server.xml
-rw-rw-r--. 1 shawn shawn  4166 Mar 20 18:59 stig-rhel6-server.xml
-rw-rw-r--. 1 shawn shawn  3108 Mar 14 20:51 test.xml
-rw-rw-r--. 1 shawn shawn 17127 Mar 14 20:51 usgcb-rhel6-server.xml
```

Since we're adding this rule to the STIG profile, load `stig-rhel6-server.xml`:

```
$ vim stig-rhel6-server.xml
```

Upon loading the file, you will be presented with the XCCDF code behind the STIG profile:

```
<Profile id="stig-rhel6-server" extends="common" xmlns="http://checklists.nist.gov/xccdf/1.1" >
<title>Pre-release Draft STIG for RHEL 6 Server</title>
<description>This profile is being developed under the DoD consensus model to become a STIG in co
ordination with DISA FSO.</description>

<select idref="encrypt_partitions" selected="true"/>

<select idref="rpm_verify_permissions" selected="true"/>
<select idref="rpm_verify_hashes" selected="true"/>
<select idref="world_writeable_files" selected="true"/>

<select idref="install_antivirus" selected="true"/>
<select idref="install_hids" selected="true"/>
```

Illustration 13: stig-rhel6-server.xml

XML Tag/Element	Description
<Profile id="..."	XCCDF Profile Name
<title>	XCCDF Rule Title
<description>	Brief description of the XCCDF profile
<Profile... extends="..."	Name of XCCDF profile to inherit
<select idref="..."	XCCDF rule to include or modify
<select … selected="..."	Should this rule be enabled in the profile? True | False Particularly important when extending other XCCDF profiles, allowing you to enable/disable rules from the parent source.

Table 4: XCCDF Profile Tags

Using the existing code as a reference:

1. Add the `package_scap-security-guide_installed rule` into the STIG profile
2. Ensure the rule is selected

If added correctly, you will have inserted a line that matches the following:

```
<select idref="package_scap-security-guide_installed" selected="true" />
```

Illustration 14: package_scap-security-guide_installed XCCDF Profile Rule

5.7 Patch Creation & Submission

Through this workshop we've made several modifications to the SSG source code. Specifically:

1. Creation of a new XCCDF rule, `package_scap-security-guide_installed`, which was placed into `RHEL6/input/system/software/integrity.xml`.
2. Creation of a new OVAL rule, `package_scap-security-guide_installed.xml`, which also involved updating the OVAL template file `RHEL6/input/checks/templates/packages_installed.csv`.
3. Modification of the STIG profile, located at `RHEL6/input/profiles/stig-rhel6-server.xml`.

We must now prepare our changes for submission back to the community, in the form of a patch. Change directories to `/var/www/html/scap-security-guide/` and run "`git commit`":

```
$ cd /var/www/html/scap-security-guide/; git commit
# On branch master
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working
directory)
#
#    modified:   RHEL6/input/checks/templates/packages_installed.csv
#    modified:   RHEL6/input/profiles/stig-rhel6-server.xml
#    modified:   RHEL6/input/system/software/integrity.xml
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#    RHEL6/input/checks/package_scap-security-guide_installed.xml
no changes added to commit (use "git add" and/or "git commit -a")
```

From the output above, our patch must reflect changes to the "modified" files and include the net-new "untracked" file. To do so, run the following commands:

```
$ git add RHEL6/input/checks/package_scap-security-guide_installed.xml
$ git commit RHEL6/input/checks/templates/packages_installed.csv \
RHEL6/input/profiles/stig-rhel6-server.xml \
RHEL6/input/system/software/integrity.xml \
RHEL6/input/checks/package_scap-security-guide_installed.xml
```

The "`git commit`" command will bring you into a vi editor, prompting you to enter details of your patch. The first line, which is the default location of your cursor at this point, is where to create the patch title. At the EOF you place details of the patch.

Edit your patch content to reflect:

- Patch title of "Added package_scap-security-guide_installed.xml to stig-rhel6-server profile"
- Patch description of "Added package_scap-security-guide_installed.xml into STIG profile, which will now mandate the installation of the SSG"

When done, your window will resemble the following:

```
Added package_scap-security-guide_installed.xml to stig-rhel6-server
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# Explicit paths specified without -i nor -o; assuming --only paths...
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   RHEL6/input/checks/package_scap-security-guide_installed.xml
#       modified:   RHEL6/input/checks/templates/packages_installed.csv
#       modified:   RHEL6/input/profiles/stig-rhel6-server.xml
#       modified:   RHEL6/input/system/software/integrity.xml
#
Added package_scap-security-guide_installed.xml into the STIG profile,
which will now mandate the installation of SSG.
```

Illustration 15: package_scap-security-guide_installed Patch Submission

Once complete, save and exit (:wq).

Your local source tree has now identified and grouped your changes into a consolidated patch. Using the git utility, we must “export” these changes in the format of a patch file. To do so, run the following command:

```
$ git format-patch origin

0001-Added-package_scap-security-guide_installed.xml-to-s.patch
```

A newly created file, `0001-Added-package_scap-security-guide_installed.xml-to-s.patch`, will be placed into your working directory.

The final step is to EMail this patch to the SSG project mailing list. Upon acknowledgement/signoff, you will be able to “git push” your changes into the project.