Using OpenSCAP

MARTIN PREISLER

Martin Preisler works as a Software Engineer at Red Hat, Inc. He works in the Identity Management and Platform Security team, focusing on security compliance using Security Content Automation Protocol. He is the principal author of SCAP Workbench, a frequent contributor to OpenSCAP and SCAP Security Guide, and a contributor to the SCAP standard specifications. Outside of Red Hat, he likes to work on open source projects related to real-time 3D rendering and game development. mpreisle@redhat.com

Security best practices dictate that we do not run any software with known and exploitable vulnerabilities, but achieving this is difficult. While vulnerability databases do exist, they are not in formats useful for scanning file systems, much less for examining VM images and containers. I work on OpenSCAP, a tool that uses information extracted from the National Vulnerability Database [1] and security policies, and checks for vulnerabilities. oscap can also remediate, or suggest remediations, for configurations that don't meet established policies. In this article, I explain how OpenSCAP works, how to use both its GUI and command-line versions, and how you can use oscap to improve your site's security.

Ensuring proper configuration and no vulnerabilities in your production environment has become an essential part of proactive security. In the past it used to be possible to manually go over a single golden image and then deploy it en masse, but that has changed radically. Typical business deployments are now much larger than they used to be and are no longer run just using physical machines. Modern deployments are using virtual machines and containers and tend to deploy many different images. This brings new challenges to both vulnerability assessment and configuration management.

Finding Vulnerabilities

Let us say we have a deployed installation. How do we figure out whether it has any vulnerabilities? We could look at the National Vulnerability Database [1] and go over the vulnerabilities one by one, comparing affected versions with the versions of software we have installed. Unfortunately, there are two major issues with this approach. First of all, we would go crazy very soon, as this approach does not scale to even one machine let alone an infrastructure. Secondly, we would not get accurate results on most enterprise Linux operating systems such as Red Hat Enterprise Linux or SUSE Enterprise Linux. The vendors of those operating systems backport fixes for vulnerabilities into older versions of the software. This way they minimize the differences between consecutive versions of the operating systems, which is something their users really appreciate. On the other hand, this makes checking version ranges for CVEs more complex because the versions no longer match the upstream original versions.

So how do we deal with this? We need to get a vulnerability database with these backports recorded. Fortunately, vendors of enterprise operating systems are increasingly supplying a so-called "CVE feed" with corrected affected versions for each vulnerability. Still, going over them manually is a lost battle; we need an automated approach.

OpenSCAP can load the CVE feed and go over all vulnerabilities for you, detecting which vulnerabilities are in your systems.

Scanning a Physical Machine for Vulnerabilities

So how does OpenSCAP perform the scanning? First, OpenSCAP loads the given CVE database which has information about security advisories from the vendor. Then it goes over every CVE item in that database, checking the package and affected version ranges to see whether we have a version in that range and thus are affected. This works very well for official, signed packages from the vendors. The vulnerability check itself does not check checksums of the RPMs; instead most security policies check checksums of every package installed from an official source. The reasoning is that we need to check all the checksums anyway because attackers might have injected into any package with any vulnerability. To save time this is done in one go as part of the "verify RPM signatures" rule in security policies. The rule does something very similar to "rpm -Va"—it verifies that properties of installed files match package metadata. We will touch on security policies later. The criteria for determining whether we are vulnerable usually look like this:

```
<criteria operator="AND">
  <criterion comment="openssl is earlier than 0:1.0.1e-30
.el6_6.2" test_ref="oval:com.redhat.rhsa:tst:20141652019"/>
  <criterion comment="openssl is signed with Red Hat
redhatrelease2 key" test_ref="oval:com.redhat.rhsa:tst
:20140679006"/>
</criteria>
```

In the example above we are checking whether `openssl` is installed, and if it is, whether it is the Red Hat signed version and also whether it's an earlier version than the one that contains the fix for the specific vulnerability. In some cases the criteria get more complex because sometimes a vulnerability gets introduced in some version and then gets fixed in another. In this case we are checking that a package is installed, is signed by Red Hat, and is either greater than some version or earlier than another version.

Let us first show how to do a vulnerability scan on a physical machine. We will assume Red Hat Enterprise Linux 6 in our example. Each vendor publishes their CVE feed at a different location; in the case of Red Hat it is https://www.redhat.com/security/data/oval/. For Red Hat Enterprise Linux 6 specifically, we need to choose Red_Hat_Enterprise_Linux_6.xml in that directory.

```
# yum install openscap-utils
# wget
https://www.redhat.com/security/data/oval/Red_Hat
        _Enterprise_Linux_6.xml
# oscap oval eval --results results.xml --report report.html
         Red_Hat_Enterprise_Linux_6.xml
```

As oscap is executed we will see lines of each of the vulnerabilities being scanned. If the line says "false", that means we are not vulnerable. The output will look like the following:

```
Definition oval:com.redhat.rhsa:def:20160286: false
Definition oval:com.redhat.rhsa:def:20160258: false
…
```

After the scan finishes we can either look at `results.xml`, the machine readable results, or `report.html`, the human-readable HTML report. Covering the machine-readable results is outside the scope of this article. Let us instead discuss the HTML report. The report will contain several rows, one for each Red Hat Security Advisory that is being checked. The green rows are the rows we do not need to be concerned about; we are not vulnerable to the CVEs in them. The rows that are highlighted orange are the ones that our infrastructure is vulnerable to. It is important to realize that each RHSA can fix one or more CVEs, that there is no direct 1:1 mapping.

Suppose we have a vulnerability in the kernel in our infrastructure. What can we do about that? To fix the situation, we should get all the latest updates installed with `yum update`. Then we need to remove the vulnerable kernels to prevent them from being booted by accident. As long as there is at least one vulnerable kernel installed, OpenSCAP will report the vulnerability being in the infrastructure.

Scanning a Container for Vulnerabilities with oscap-docker

We could scan a container by installing the tools and security policies inside it and then running oscap. But that is impractical and goes against best practices of container deployment. Instead we want to scan containers from the host without affecting them.

There are two ways of scanning a container with OpenSCAP. Let us start with oscap-docker, which is a command-line tool wrapping the functionality of oscap. It has the same command-line arguments as oscap with the exception of the first two arguments—the mode of operation and the container or image ID. Before we can use it we need to install it; on Red Hat Enterprise Linux 7.2 it is part of the openscap-utils package.

After it is installed we can use it if we have root privileges. There are two subcommands: `container-cve` scans a running container, while `image-cve` scans a container image.

```
# oscap-docker container-cve $TARGET_ID
# oscap-docker image-cve $TARGET_ID
```

To start, we can scan a single container image: for example, the `rhel7` base image.

```
# docker pull rhel7
# oscap-docker image-cve rhel7
2016-02-25 12:07:58
URL:http://www.redhat.com/security/data/oval/com.redhat.rhsa
-all.xml.bz2 [1863765/1863765] -> "docker.6xTkgY/cve-oval.xml
.bz2" [1]
Definition oval:com.redhat.rhsa:def:20160286: false
Definition oval:com.redhat.rhsa:def:20160258: false
…
```

The lines ending with "false" are telling us that we are not vulnerable to CVEs listed in the respective Red Hat Security Advisories. If any of the lines end with "true" we are in trouble and need to update our image.

Scanning a Container for Vulnerabilities with Atomic

In case you are using Atomic for container management, you can use the atomic scan functionality instead. The advantage is that it is easier to use since it automatically manages the CVE feeds for the user. The tool makes sure you are using the right CVE feed and that it is up-to-date. If you are running Red Hat Enterprise Linux 7 and do not have the atomic command-line tool installed you need to run:

```
# yum install atomic
```

If you are on Atomic Host, the atomic command should already be available. After the atomic command is installed, you need the OpenSCAP-daemon to perform the scans. You can install it directly on the host and run it or you can download a super-privileged container (SPC) image that provides it. We will go with the SPC image in this section because it is a little bit simpler to set up. For the SPC image, we will be using the Fedora 23 OpenSCAP-daemon container image. There may be other images available in the future.

```
# atomic install openscap/openscap-daemon-f23
# atomic run openscap/openscap-daemon-f23
```

When the SPC is in place and running we can issue atomic scan commands.

Let us now look at an example of atomic scan. This time we will use a custom Red Hat Enterprise Linux 7.2 image that I created that actually has vulnerabilities. It will help us demonstrate features of the atomic scan.

```
# atomic scan 6c3a84d798dc
Container/Image     Cri      Imp      Med      Low
-------------------  ---      ---      ---      ---
6c3a84d798dc         0        0        2        0
```

As we can see, container image 6c3a84d798dc has two medium-severity vulnerabilities. How do we list them? We need to use the `--detail` argument.

```
# atomic scan --detail 6c3a84d798dc
6c3a84d798dc
 OS       : Red Hat Enterprise Linux Server release 7.2 (Maipo)
 Moderate : 2
   CVE      : RHSA-2016:0008: openssl security update (Moderate)
   CVE URL  : https://access.redhat.com/security/cve/CVE-2015-7575
   RHSA ID  : RHSA-2016:0008-00
   RHSA URL : https://rhn.redhat.com/errata/RHSA-2016-0008.html

   CVE      : RHSA-2916:0007: nss security update (Moderate)
   CVE URL  : https://access.redhat.com/security/cve/CVE-2015-7575
   RHSA ID  : RHSA-2016:0007-00
   RHSA URL : https://rhn.redhat.com/errata/RHSA-2016-0007.html
```

If we need to scan a container instead of an image, we just need to replace the ID with an ID of the container. Atomic scan also allows scanning all images, all containers, or both with a single command—`--images`, `--containers`, and `--all`, respectively.

Vulnerability Scanning for Virtual Machines

Scanning for vulnerabilities on virtual machines is technically very similar to scanning containers, but the commands are different. Instead of using oscap-docker, we can use oscap-vm to scan virtual machines. Keep in mind that the oscap-vm command is fairly new. It is available on Fedora but still not available on Red Hat Enterprise Linux 7 at the time of this writing. It is part of the openscap-utils package we have installed previously.

oscap-vm allows us to scan running or shut down virtual machines, or raw storage images. Let us look at an example:

```
# wget https://www.redhat.com/security/data/oval/Red_Hat
      _Enterprise_Linux_6.xml
# oscap-vm domain rhel6vm oval eval --results results.xml
      --report report.html Red_Hat_Enterprise_Linux_6.xml
```

Here, we are testing a Red Hat Enterprise Linux 6 virtual machine called "rhel6vm" running on the host.

Checking Configuration with OpenSCAP

So far we have only talked about vulnerabilities. We also need to make sure our infrastructure is set up in a secure way, that we have hardening in place. To do that we first need to choose a set of rules—a so-called security policy. A security policy is usually a list of rules in PDF or even printed out. Each rule usually has a description, rationale, identifiers, and some steps to check and fix the machines. The workflow with these security policies is that the auditors carry them in big binders and manually check the machines for compliance. This may be fine for small infrastructures, but it does not scale and is not cost effective.

Let us explore how to use OpenSCAP for fully automated security compliance. OpenSCAP is what has searched vulnerabilities in the first section of this article, but it was hidden under a few layers of abstraction. Now we need to interact with it more

directly so it makes sense to introduce it. OpenSCAP is an open-source implementation of SCAP 1.2, the standard for automated security compliance. You can read more about OpenSCAP at https://www.open-scap.org/. We will start by choosing a suitable security policy and profile. For the purposes of this article let us use PCI-DSS profile from SCAP Security Guide for Red Hat Enterprise Linux 7. SCAP Security Guide, or SSG, is another open-source project we will be using. SSG provides SCAP security policies for various products like Red Hat Enterprise Linux 6, 7, Fedora, CentOS, Firefox, and others. We will again start by scanning a physical machine before moving to containers. Let us log into the machine and install the necessary packages—scap-workbench and scap-security-guide. You might think that using such tools as SCAP Workbench, a graphical user interface, feels out of place in system administration. Keep in mind that SCAP Workbench lets you prepare the customized policies for a fully automated deployment in the future.

After installation has finished, we can start SCAP Workbench by clicking its icon in Applications → System Tools.

SCAP Workbench will start and ask us which content to select. Since we installed SCAP Security Guide in the previous step, we have the option to select `ssg-rhel7-ds.xml` in `/usr/share/xml/scap/ssg/content`. We will discuss how to scan using the command line only later in the article.

Once the content is loaded we will be presented with the main window of SCAP Workbench, which lists the rules that will be applied to the system. Let us select the PCI-DSS profile from the profile combo box.

After selecting the PCI-DSS profile we are all set to perform the initial scan. The only thing we need to do is click Scan, elevate privileges by typing the password, and wait a few minutes. On a default Red Hat Enterprise Linux 7.2 installation at the time of writing, the results were 31 passes and 43 fails. If we click Show Report we can see more details about our system.

At this point we can make customizations to the security policy by clicking "Customize." For example, we may want our infrastructure to be set up more strictly than PCI-DSS requires. In that case we can select additional rules to check and even increase minimum password length or other values. After we are done with the customization, we can choose File → Save as RPM, which gives us a package with our customized security policy ready-made to be deployed using Satellite 6.

What we have achieved above can be done using the command line only, with the exception of the customization.

```
# oscap xccdf eval --profile xccdf_org.ssgproject.content
      profile_pci-dss --results /tmp/results.xml --report
      /tmp/report.html /usr/share/xml/scap/ssg/content/
      ssg-rhel7-ds.xml
```

The snippet above will scan the local machine for compliance with PCI-DSS and will store results and report in `/tmp/results.xml` and `/tmp/report.html`, respectively.

Changing the Configuration to Be Compliant with OpenSCAP

If we want to change configuration of the machine to make more rules pass, we need to check the Remediate checkbox in SCAP Workbench and click Scan again. If remediation is enabled, SCAP Workbench will go over the rules figuring out which are passing and which are failing.

Then for each failing rule it will run a so-called remediation—code that automatically fixes the configuration—and then check the rule again. In case the rule is now passing, SCAP Workbench will declare the rule as *fixed*. If everything worked smoothly, our Red Hat Enterprise Linux 7 installation should report no failed rules.

If we cannot use the GUI, we can do the above using the command line only:

```
# oscap xccdf eval --profile xccdf_org.ssgproject.content
      _profile_pci-dss --remediate --results /tmp/results.xml
      --report /tmp/report.html /usr/share/xml/scap/ssg/content
      /ssg-rhel7-ds.xml
```

The important difference from scanning is the `--remediate` option. This instructs oscap to run remediation scripts on every failed check.

Keep in mind that automated remediations can be dangerous and cannot be undone! They can break some of the functionality of deployed infrastructure! We recommend testing remediations on nonproduction machines before deployment.

Very likely you are running a configuration management system, such as Puppet, Chef, or Ansible. In this case the remediations will still work but the configuration management systems may override them, putting your systems out of compliance! Instead of running the remediations, it may be more valuable to see their code and adapt the settings of the configuration systems accordingly. To generate a list of fixes instead of running them, run the following:

```
$ oscap xccdf generate fix --result-id xccdf_org.open-scap
      _testresult_xccdf_org.ssgproject.content_profile_pci
      _dss /tmp/results.xml
```

The `result-id` will be correct if you ran the PCI-DSS evaluation we have just discussed. In case you used a different profile, look into the `/tmp/results.xml` file, find the `<TestResult>` element, and use its id attribute.

The above will output a shell script into stdout with all the changes OpenSCAP would make to your system if you ran the remediation.

Container Security Compliance

Now we can scan the configuration of local and remote machines, but how do we deal with containers? We could scan them as a remote machine but that would require installing SSH and openscap-scanner inside, which is impractical. Instead, let us look at how to scan containers from the host. We will need the oscap-docker tool, which is part of the openscap-utils package.

We recommend running `oscap-docker --help` to explore its capabilities. It can operate in four different modes. We have already seen how it can scan containers for vulnerabilities, so we will skip over the image-cve and container-cve modes in this section. Instead we will scan a container image for security compliance.

```
# oscap-docker image $IMAGE_ID xccdf eval --profile
      xccdf_org.ssgproject.content_profile_common
      --results /tmp/results.xml --report /tmp/report.html
      /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

These command-line arguments should look familiar. Apart from the first two, we are using the same command-line arguments as with the oscap tool. Instead of using the PCI-DSS profile in this section, we will use a new profile called the common profile. It is less focused on the financial industry and contains rules checking common security practices instead.

With the command-line snippet above we are doing roughly the same thing as with SCAP Workbench and oscap earlier in this article; we have scanned a container image with the common profile with content coming from the SCAP Security Guide project for Red Hat Enterprise Linux 7. The results are stored in `/tmp/results.xml`, and the HTML report is stored in `/tmp/report.html`.

Virtual Machine Security Compliance

As is the case with vulnerability assessment, scanning for security compliance is very similar between containers and virtual machines. Instead of using the `oscap-docker` command, we need to use `oscap-vm`.

```
# oscap-vm domain rhel7vm xccdf eval --profile
      xccdf_org.ssgproject.content_profile_common --results
      /tmp/results.xml --report /tmp/report.html
      /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

The semantics are the same between `oscap-vm` and `oscap-docker`. The key difference is under the hood. `oscap-vm` uses guestmount to inspect virtual machines instead of the atomic mount mechanism inside `oscap-docker`.

Offline Evaluation Advantages and Limitations

What we have used in the previous two sections is called offline SCAP evaluation. When we are scanning local or remote machines, we are using the normal online evaluation. When scanning containers and virtual machines from the host, we are using the offline evaluation. The difference between offline and online evaluation is that the latter has access to all running processes and can do runtime checks. That means that it can, for example, ask systemd about information about a unit. Offline scanning does not have access to the running system; it is exploring the file system mounted somewhere in read-only mode.

This has advantages and disadvantages. One advantage of offline scanning is that we can scan a running container or virtual machine without any risk of affecting them. On the other hand, we cannot perform some of the runtime checks like checking which processes are running. We also cannot fix systems in the offline evaluation mode since the file systems are mounted read-only. That is not a limitation of the offline mode but rather its implementation in OpenSCAP. There is an outstanding feature request to fix this [2].

Conclusion

Using OpenSCAP helps businesses prevent vulnerabilities and insecure configuration settings in their infrastructure. In this article we have explored how to use OpenSCAP for physical machines as well as for virtual machines and containers. Mixing automated SCAP remediations with configuration management systems proved difficult but can be handled by going through the remediation steps and adapting configuration systems accordingly. Using SCAP for containers and virtual machines requires a new approach called offline evaluation. While limited and fairly new, it is proving useful for practical container and virtual machine evaluation.

Acknowledgments

I would like to thank Jan Černý, Rik Farrow, Yoana Ruseva, and anonymous reviewers for helping me with this article.

References

[1] National Vulnerability Database: https://nvd.nist.gov/.

[2] Feature request to add offline mode repair: https://fedorahosted.org/openscap/ticket/467.