Using XCCDF for Remediation: Best Current Practice

Author: Luis Nunez[1]

Contents

[1] The author would like to acknowledge David Mann for providing feedback and review on this paper.

1 Executive Summary

Security configuration management requires that three kinds of content be published: configuration guidance, audit check logic and remediation logic. Standards and best practices exist for the publication of guidance documents and audit check logic such as the National Institute of Standards and Technology (NIST) Security Content Automation Protocol (SCAP)[2], which includes the Extensible Configuration Checklist Description Format (XCCDF)[3] and Open Vulnerability and Assessment Language (OVAL)[4]. NIST SCAP is currently used by the Defense Information Systems Agency (DISA) and participating vendors for the creation and publication of DISA Security Technical Implementation Guide (STIG)[5]. However, no standards or published best practices exist for the publication of automated fix logic.

This paper describes the practices used by Red Hat and the SCAP Security Guide (SSG)[6] project to publish fix logic for Red Hat Enterprise Linux 6 using the existing XCCDF schema. This paper also describes recommendations to extend this approach to achieve more functionality. This is a short term strategy to meet current remediation needs.

The purpose of this paper is to document and describe these approaches to facilitate community discussion of standards and best-practices for the publication of automated fix content.

2 Configuration Content Basics

This section provides background on the relationships and dynamics of security content publication and use.

2.1 Content Ecosystem

There are three kinds of content that enable security configuration management. They are: configuration guides, audit check logic, and fix logic.

- Guides - Configuration guides are used for meeting security requirements and ensuring the secure posture of the system. Configuration guides include recommended security settings for operating systems, applications and technologies.

[2] http://scap.nist.gov/
[3] http://scap.nist.gov/specifications/xccdf/index.html
[4] http://oval.mitre.org/
[5] http://iase.disa.mil/stigs/index.html
[6] https://fedorahosted.org/scap-security-guide/

- Configuration audit check logic – Configuration audit check logic involves automated procedures necessary to determine the configuration state of the system. It is used to validate against a specified system configuration setting.
- Configuration fix logic – Configuration fix logic details the automated process or manual procedure necessary to change a configuration to achieve a specified state.

There are two primary sources for security configuration guidance. Software vendors typically publish security best practices as part of their documentation. For example Microsoft has a long tradition of providing security guides for the Microsoft Windows operating systems. Third-party content creators also publish security guidance. The range of third-party content producers includes both government and non-government organizations. NIST maintains the United States Government Configuration Baseline (USGCB)[7] as a government security configuration baseline for products what are widely deployed within the government. DISA, a component of the Department of Defense (DoD), publishes their STIGs for securing DoD information systems. The Center for Internet Security[8], a non-profit organization, publishes security configuration guides though consensus working groups for its members.

There are several types of consumers that utilize security configuration content. In many cases the security guides are used to support regulatory compliance efforts. Federal Departments and Agencies rely on NIST USGCB for FISMA compliance. The DoD uses directive (DoDD) 8500[9] which states that all Information Assurance (IA) devices and IA enable devices within the DoD must comply with DISA STIGs. In the private sector, corporations such as Duke Medical Center, Citibank and Ford often use Center for Internet Security benchmarks to demonstrate due diligence in meeting regulations such as the Federal Information Security Management Act (FISMA)[10], the Payment Card Industry Data Security Standard (PCI DSS)[11], the Sarbanes Oxley act (SOX)[12], and the Health Insurance Portability and Accountability Act (HIPAA)[13]. Lastly, commercial tool vendors consume security guidance content to produce capabilities within their products that are capable of performing automated auditing of end systems. Typically a consumer will purchase the tool from a tool vendor and perform audit checks of its systems. A variety of tool vendors such as McAfee, Symantec and Tripwire sell such capabilities and services to both commercial and government organizations.

[7] http://usgcb.nist.gov/
[8] http://benchmarks.cisecurity.org/
[9] http://www.prim.osd.mil/Documents/DoDD_8500_1_IA.pdf
[10] http://csrc.nist.gov/groups/SMA/fisma/index.html
[11] https://www.pcisecuritystandards.org/security_standards/
[12] http://www.gpo.gov/fdsys/pkg/PLAW-107publ204/pdf/PLAW-107publ204.pdf
[13] http://www.hhs.gov/ocr/privacy/

Summarizing, the configuration content ecosystem can be described in terms of content creators and consumers as shown in the graphic below.

Configuration guidance content can be augmented with configuration audit logic and fix logic. These are discussed in more detail in the following sections.

2.2 Configuration Audit Check logic

The goal of configuration audit is to determine if a given setting is compliant with stated policy. There are two types of audit check logic. Tool vendors will support one or both types. Proprietary check logic is a methodology conceived and owned by the configuration audit tool vendor for configuration audit checking and analysis. Tool vendors such as McAfee, Symantec and Tenable have proprietary capabilities for checking systems for configuration issues.

SCAP is a suite of specifications combined to form a NIST standard for the publication of automated security content. SCAP utilizes several specifications, two of which are relevant to this discussion. XCCDF is a language designed to convey configuration checklist policies along with assessment and remediation information. OVAL is a language to express automated check logic for testing of systems in a consistent way. USGCB SCAP benchmark content, maintained by NIST, is consumable by SCAP validated tools to perform configuration audit checks on government systems. In a similar manner, DISA is in the process of making automated check logic available by converting their STIGs into XCCDF and OVAL benchmark content. Various audit tool vendors have adopted the SCAP standard and have also gone through the NIST SCAP validation program.

2.3 Fix logic

There are three pieces of information necessary to specify a fix:

- Applicable systems - Describes an end system or set of end systems that the fix can be applied to. For example, DISA STIGs are available for Windows 2008, Solaris 10 and Red Hat 5.[14]

[14] http://iase.disa.mil/stigs/scap/index.html

- Configuration setting - The configuration setting that needs to be changed.
- Desired state - The desired value to applied to the configuration setting to bring it into compliance with policy.

For the purposes of this paper, we distinguish among the terms, mitigation**,** remediation and fix. A **mitigation** describes any change to the system or its operating environment that would lessen the effects of the mis-configuration. For example, if the configuration policy states that a service should not be running, blocking network traffic to the related port might be a mitigation action taken to defend against the threat of the service being attacked. A **remediation** describes a direct change of a non-compliant configuration setting to a specified or allowed value. In this example, stopping or removing the service would be the associated remediation. A **fix** is a sequence of actions applied to an end system or a part the end system's operating environment to implement either a mitigation or remediation.

There are a variety of system management tools and capabilities that can apply fix content on endpoints. Local fix capabilities are used to fix standalone or single systems. For example a shell script can be written to apply a fix to a UNIX or Linux system. Enterprise fix solutions provide capabilities that allow a fix to be applied to multiple end systems. Microsoft Group Policy Objects (GPOs) are one such example.

Currently fix content comes from two reliable sources: primary source vendors and systems management tool providers. With respect to primary source vendors, many major OS and application providers include manual fix content which can be applied locally as part of their security configuration guides. Additionally, Microsoft provides methods of applying fixes at the enterprise level: GPOs with their Security Compliance Manager (SCM) for implementation in Microsoft Active Directory. Third-party system-management tool vendors such as McAfee, IBM, and HP also provide enterprise-scale automated fix capabilities, but the logic is typically expressed in closed, proprietary formats.

2.4 State of Remediation Standardization

Currently, there is no standard for specifying fix logic for use by system-management tools. As a result, consumers who have heterogeneous environments (e.g. the US government) run into interoperability issues when deploying fix logic content at the enterprise level.

In February of 2011 NIST published Interagency Report (IR) 7670 "Proposed Open Specifications for an Enterprise Remediation Automation Framework (Draft)".[15] The draft covers notional enterprise architecture and remediation workflow. No revision or further work is in progress.

The Common Remediation Enumeration (CRE)[16] was introduced at about the same time as NIST IR 7670. Thus far, no real adoption of the specification has surfaced and no revision or further work is in progress.

In the summer of 2012 at the Security Automation Developer Days[17] event, a session on "Reinvigorating Remediation" was held. A proposal to leverage XCCDF as a remediation policy language was introduced. There was also a discussion on European Computer Manufacturers Association (ECMA)[18] scripting language vs. Extensible Markup Language (XML)[19] for Open Vulnerability Remediation Language (OVRL)[20].

3 DISA STIGs and Red Hat

Security Technical Implementation Guides (STIG) are produced by the DISA Field Security Office (FSO). They are used to harden DoD information systems.

DISA established a "vendor process"[21] to encourage software vendors to participate in the STIG creation process. This is a collaborative process between the software vendor and the DISA FSO to produce authoritative, product specific STIGs. To facilitate collaboration on the RHEL 6 STIG, Red Hat is participating[22] in the open source project by the name SCAP Security Guide project (SSG). The SSG project produces configuration content in accordance with the DISA STIG and the USGCB.

Consistent with NIST SCAP, the RHEL 6 STIG provides configuration guidance in XCCDF format, as well as check logic expressed in OVAL and Open Checklist Interactive Language (OCIL)[23] . The guidance also includes references to the NIST SP-800-53[24] IA controls. Note that DoD is in the process of adopting the

[15] http://csrc.nist.gov/publications/drafts/nistir-7670/Draft-NISTIR-7670_Feb2011.pdf
[16] http://csrc.nist.gov/publications/drafts/nistir-7831/Draft-NISTIR-7831.pdf
[17] http://measurablesecurity.mitre.org/participation/devdays.html#summer2012
[18] http://www.ecma-international.org/
[19] http://www.w3.org/TR/REC-xml/
[20] http://csrc.nist.gov/publications/drafts/nistir-7670/Draft-NISTIR-7670_Feb2011.pdf
[21] http://iase.disa.mil/stigs/vendor_process/index.html
[22] http://blog-shawndwells.rhcloud.com/dont-miss-the-point-of-the-rhel6-stig/
[23] http://scap.nist.gov/specifications/ocil/
[24] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf

NIST controls for use within DoD information system risk management. This information can be used to produce a security requirements traceability matrix.

The SSG chose to include fix content in the form of automated shell scripts as part of the RHEL 6 STIG. The SSG chose to use existing XCCDF elements and attributes to convey fix logic. The following is an example of fix script embedded into XCCDF.

```
<fix system="urn:xccdf:fix:script:sh">
       yum -y install screen
</fix>
```

The next section will provide more detail on embedding fix information into XCCDF.

4 XCCDF elements

The XCCDF language provides facilities to express security configuration policies organized into profiles, groups, rules, values. This section covers the various elements and attributes available for fix logic content expression.

Most of the recommendations described below are based on the experience of the RHEL 6 SSG Project. Others are recommendations made jointly by MITRE and the National Security Agency (NSA). They are both presented here to serve as a starting point for developing a set of best practices for conveying fix logic in manner that can be automated and shared.

The following table summarizes which XCCDF elements should be used to convey fix logic and whether the recommendation is based on the experience of the RHEL 6 SSG or based on the analysis of MITRE and the NSA. A detailed discussion of each element follows.

4.1 <xccdf:Value> and <xccdf:sub>

The <xccdf:Value> is defined as follows:

> The <xccdf:Value> is a named parameter (with a unique *@id* attribute) that can be substituted into properties of other elements within the benchmark, including the interior of structured check specifications and fix scripts.[25]

The <xccdf:sub> is defined as follows:

> The <xccdf:sub> element represents a reference to a parameter value that may be set during tailoring.[26]

The example of the value and the sub elements is used to pass parameters to a script for remediation. The Value element is used with the sub element. These elements may also be with checking logic. The following is an example of Red Hat's use of these elements to convey login banner text:

```
<Value id="login_banner_text" operator="equals" type="string">
       <title xml:lang="en-US">Login Banner Verbiage</title>
       <description xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">
Enter an appropriate login banner for your organization. Please note that new lines
must be expressed by the '\n' character and special characters like parentheses and
quotation marks must be escaped with '\'.</description>
       <value selector="usgcb_default"> USGCB Default Banner</value>
       <value selector="dod_default"> DOD Default Banner</value>
       <value selector="dod_short">I\'ve read \&amp; consent to terms in IS user
       agreem\'t.</value>
</Value>
<fix system="urn:xccdf:fix:script:sh">login_banner_text="<sub
idref="login_banner_text"/>"
       cat &lt;&lt;EOF &gt;/etc/issue
       $login_banner_text
       EOF
</fix>
```

4.2 <xccdf:fixtext>

The <xccdf:fixtext> is defined as follows:

> Data that describes how to bring a target system into compliance with this rule. Each <xccdf:fixtext> element MAY be associated with one or more <xccdf:fix> element values.[27]

[25] Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2. See: http://csrc.nist.gov/publications/nistir/ir7275-rev4/NISTIR-7275r4.pdf.
[26] Ibid.

This element is used to describe manual procedures, commands, and details to bring a system to a desired state. The following is an example taken from the DISA Red Hat RHEL6 STIG benchmark.

```
<fixtext fixref="F-RHEL-06-000071_fix">
   To enable console screen locking when in text mode, install   the  "screen"
   package:
   # yum install screen
   Instruct users to begin new terminal sessions with the following command:
   $ screen
   The console can now be locked with the following key combination:
   ctrl+a x
</fixtext>
```

4.3 fixref

The fixref attribute of the fixtext element is defined as:

> A reference to a specific <xccdf:fix> @id attribute. This allows pairing explanatory text with specific fix procedures. [28]

The following is an example from the DISA Windows 7 STIG benchmark use of the fixref attribute. The fixref attribute is used to reference the fix element id.

```
<fixtext fixref="F-104r1_fix">Configure the policy value for Computer Configuration
Windows Settings  Security Settings Local Policies Security Options  "Microsoft
Network Server: Digitally sign communications (if Client agrees)" to "Enabled".
</fixtext>

<fix id="F-104r1_fix"/>
```

4.4 <xccdf:fix>

The fix element is described as:

> A command string, script, or other system modification statement that, if executed on the target system, can bring it into full, or at least better, compliance with this rule.[29]

---

[27] Ibid.
[28] Ibid.
[29] Ibid.

The following Red Hat <xccdf:fix> example is related to the STIG id RHEL-06-000071 "The system must allow locking of the console screen in text mode." The "yum –y screen" command installs the screen utility. Installing the screen utility will satisfy the policy.

```
<fix system="urn:xccdf:fix:script:sh">
          yum -y install screen
</fix>
```

4.5 fix system

The fix system attribute of the fix element is described as:

> A URI that identifies the scheme, language, engine, or process for which the fix contents are written.[30]

The fix system is used by Red Hat to embed and to identify scripting as the remediation method. The Red Hat example below indicates the that the content is a shell script.

```
<fix system="urn:xccdf:fix:script:sh">
      if rpm -qa | grep -q xinetd; then
            yum -y remove xinetd
      fi
</fix>
```

4.6 fix system commands

This fix system "urn:xccdf:fix:commands" Uniform Resource Identifier (URI) is described as:

> A list of target system commands; executed in order, the commands should bring the target system into compliance with the rule.[31]

In this example the <xccdf: system="urn:xccdf:fix:commands"> fix is used for Cisco IOS configuration. The Cisco IOS command "ip ssh version 2" is used to satisfy the policy. The fix system commands may be used for command line interface based systems. When this fix system is used the commands must be valid on every platform allowed by the platform tag in the fix, rule, group, or benchmark.

```
<Rule id="xccdf_org.mitre_rule_ios_SSH_1" >
      <fixtext fixref="fix_ios_ssh_1" reboot="false">
      Configure the network element to use SSH version 2.</fixtext>
```

[30] Ibid.
[31] Ibid.

```
        <fix id="fix_ios_ssh_1" reboot="false" system="urn:xccdf:fix:commands"
        platform="cpe:2.3:o:cisco:ios:12.3:-:-:-:-:-:-:-">
                  ip ssh version 2</fix>
                <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
                          <check-content-ref name="oval:com.c3isecurity.ios:def:1111"
                          href="cisco-ios-oval.xml"/>
                </check>
</Rule>
```

4.7 platform

The platform attribute is described as:

> In case different fix scripts or procedures are required for different target platform types (e.g., different patches for Windows Vista and Windows 7), this attribute allows a CPE name or CPE applicability language expression to be associated with an <xccdf:fix> element. This SHOULD appear on an <xccdf:fix> when the content applies to only one platform out of several to which the rule could apply.[32]

The platform element is used to convey platform applicability information related to the fix.

```
<Rule id="xccdf_org.mitre_rule_ios_SSH_1" >
        <fixtext fixref="fix_ios_ssh_1" reboot="false">
        Configure the network element to use SSH version 2.</fixtext>
        <fix id="fix_ios_ssh_1" reboot="false" system="urn:xccdf:fix:commands"
        platform="cpe:2.3:o:cisco:ios:12.3:-:-:-:-:-:-:-">
                  ip ssh version 2</fix>
                <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
                          <check-content-ref name="oval:com.c3isecurity.ios:def:1111"
        href="cisco-ios-oval.xml"/>
                </check>
</Rule>
```

4.8 reboot

The reboot attribute exists in the <xccdf:fix> and <xccdf:fixtext> elements and is described as:

> Whether or not remediation will require a reboot or hard reset of the target. When specified, it SHALL have one of two values: true (1) or false (0) (default: 0).

Some remediations do not take effect until the system is rebooted. The reboot attribute is used to convey system reboot information related to the fix.

```
<Rule id="xccdf_org.mitre_rule_SSH_1">
```

[32] Ibid

```
        <fixtext fixref="fix_junos_ssh_1" reboot="true">Configure the network element
        to use SSH version 2.</fixtext>
        <fix id="fix_junos_ssh_1" reboot="true" system="urn:xccdf:fix:commands"
        platform="cpe:2.3:o:juniper:junos:12.1:-:-:-:-:-:-:-:-">
                 set system services ssh protocol-version 2
        </fix>
        <check system=http://oval.mitre.org/XMLSchema/oval-definitions-5
        id="junos_ssh_oval">
        <check-content-ref name="oval:com.c3isecurity.junos:def:3333" href="juniper-
        junos-oval.xml"/>
         </check>
</Rule>
```