engineering laboratory

Using DITA to Create Security Configuration Checklists

A Case Study

Joshua Lubell

Cybersecurity for Smart Manufacturing Systems Project

Balisage Markup Conference
August 2017

National Institute of Standards and Technology
U.S. Department of Commerce

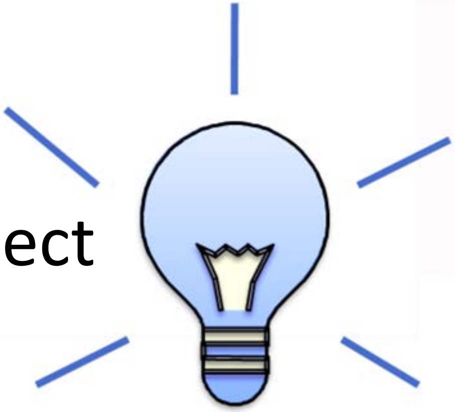
Disclaimer

About Me

- Member of NIST's Cybersecurity for Smart Manufacturing Systems project
  - Objective: Deliver a manufacturing-tailored cybersecurity risk management framework with supporting guidelines, methods, metrics and ***tools*** that addresses performance, reliability, and safety requirements
- Experienced user of markup technologies for manufacturing applications
  - Previously contributed to ISO 10303 standard (also known as STEP), used in most computer-aided design systems
  - Now I develop tools to aid producers of security control baselines, profiles, and other security content for the Industrial Internet of Things

Outline

- Security Configuration checklists
- Extensible Configuration Checklist Description Format (XCCDF)
- Show how the SCAP Security Guide project handles XCCDF authoring challenges
- Darwin Information Typing Architecture (DITA) as an alternative
- Conclusions



Security Configuration Checklists

- Also known as Benchmarks, Hardening Guides
- Provide concrete, actionable instructions for configuring an IT product to an operational environment
- Traditionally in form of documents to be read by IT security practitioners
- Automatable if represented as semantically rich XML
- Configuration settings map to security controls

Some general security controls

Boundary Protection	Restricting communications between subsystems and with external systems
Least Functionality	Disabling of unnecessary ports and services
Authentication	Verifying the identity of a user, process, or device
Least Privilege	Authorizing only the minimum access needed for users or processes to accomplish assigned tasks

Implementing these protects against a wide variety of cyberattacks.

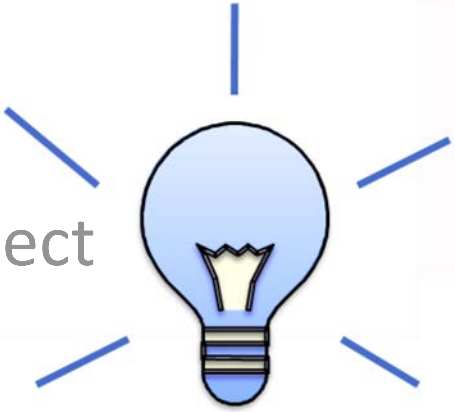
Configuration checklists improve a system's security posture …

… by ensuring proper application of security controls

National Institute of Standards and Technology
U.S. Department of Commerce

Outline

- Security Configuration checklists
- **Extensible Configuration Checklist Description Format (XCCDF)**
- Show how the SCAP Security Guide project handles XCCDF authoring challenges
- Darwin Information Typing Architecture (DITA) as an alternative
- Conclusions

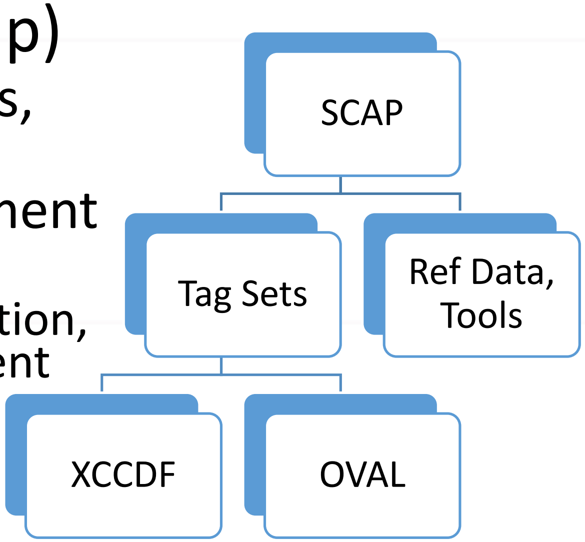


NIST
National Institute of Standards and Technology
U.S. Department of Commerce

XCCDF

security
benchmark
automation

- Standard for representing structured collections of security configuration rules for target systems
- Part of the **S**ecurity **C**ontent **A**utomation **P**rotocol (SCAP — pronounced ess-cap)
  - Ecosystem of interoperable XML tag sets, reference data, and software tools
  - Includes the Open Vulnerability Assessment Language (OVAL)
    - Represents system configuration information, assesses machine state, reports assessment results
  - Required for government agencies for ensuring enterprise IT configuration compliance

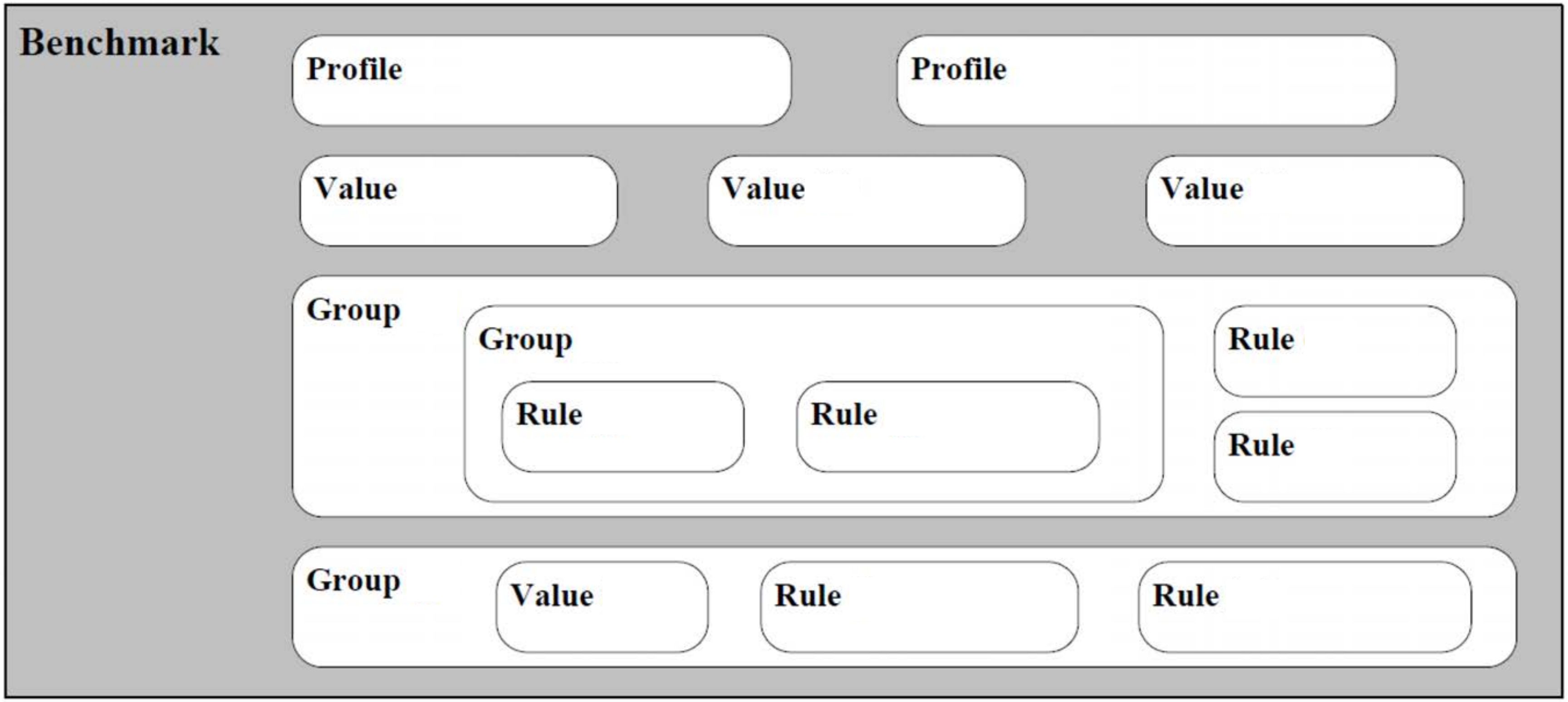


Main XCCDF elements

Element	Purpose
`<Benchmark>`	Checklist root element
`<Rule>`	Specifies single item to check, and how checking should be done
`<Value>`	Parameter to be used within rules
`<Group>`	Aggregation of rules, values, and/or other groups
`<Profile>`	Collection of references to group, rule, value elements. Allow different combinations of groups and rules to be enabled for inclusion in a series of tests.

`<check>`
`<complex-check>`
	Specifies a target system check needed to test compliance with a rule

Benchmark structure example



From XCCDF Version 1.2 Specification
(NISTIR 7275)

Simple checklist scenario

- Red Hat Enterprise Linux version 7 (REHL7) controlling collaborative robotic manufacturing process
- Focus on security principles of Boundary Protection and Least Privilege
- No `<Group>` or `<Value>` elements
- Two profiles
  - Firewall
  - Firewall with Mandatory Access Control (MAC)
    - SELinux kernel module provides MAC
    - MAC useful for implementing Least Privilege in robot controller s/w

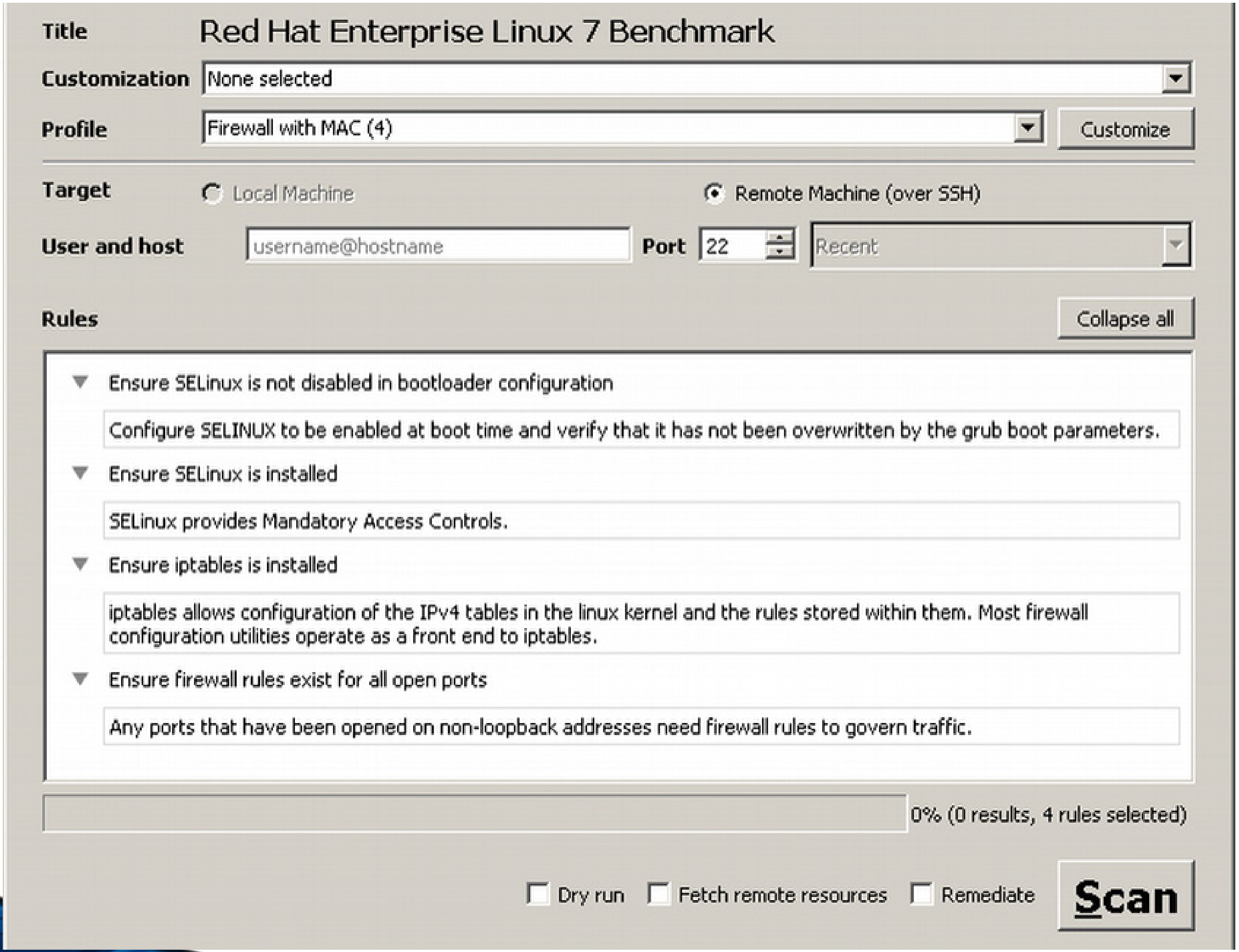
XCCDF checklist for REHL7 example

```
<Benchmark xmlns="http://checklists.nist.gov/xccdf/1.2"
id="xccdf_gov.nist_benchmark_Red_Hat_Enterprise_Linux_7_Benchmark"
style="SCAP_1.2">
   <status date="2016-06-02">interim</status>
   <title>Red Hat Enterprise Linux 7 Benchmark</title>
   <description>…</description>
   <version>2.1.0</version>
   <metadata>…</metadata>
   <Profile id="xccdf_gov.nist_profile_Firewall">…</Profile>
   <Profile
id="xccdf_gov.nist_profile_Firewall_with_MAC">…</Profile>
   <Rule id="xccdf_gov.nist_rule_Ensure_SELinux_not_disabled_in_
bootloader_configuration" selected="false"…>…</Rule>
   <Rule id="xccdf_gov.nist_rule_Ensure_SELinux_is_installed"
selected="false"…>…</Rule>
   <Rule id="xccdf_gov.nist_rule_Ensure_iptables_is_installed"
selected="false"…>…</Rule>
   …
</Benchmark>
```

Firewall_with_MAC profile

```
<Profile xmlns="http://checklists.nist.gov/xccdf/1.2"
id="xccdf_gov.nist_profile_Firewall_with_MAC">
  <title>Firewall with MAC</title>
  <description>This profile extends the "Firewall" profile to
check configuration of Mandatory Access Control(MAC).
  </description>
  <select idref=
"xccdf_gov.nist_rule_SELinux_not_disabled_in_bootloader_configu
ration“ selected="true"/>
  <select idref="xccdf_gov.nist_rule_SELinux_is_installed"
selected="true"/>
  <select idref="xccdf_gov.nist_rule_iptables_is_installed"
selected="true"/>
  <select idref=
"xccdf_gov.nist_rule_firewall_rules_exist_for_all_open_ports"
selected="true"/>
</Profile>
```

Profile in SCAP Workbench



Title Red Hat Enterprise Linux 7 Benchmark

Customization None selected

Profile Firewall with MAC (4)

Target ○ Local Machine ● Remote Machine (over SSH)

User and host username@hostname **Port** 22

Rules

- Ensure SELinux is not disabled in bootloader configuration
  - Configure SELINUX to be enabled at boot time and verify that it has not been overwritten by the grub boot parameters.
- Ensure SELinux is installed
  - SELinux provides Mandatory Access Controls.
- Ensure iptables is installed
  - iptables allows configuration of the IPv4 tables in the linux kernel and the rules stored within them. Most firewall configuration utilities operate as a front end to iptables.
- Ensure firewall rules exist for all open ports
  - Any ports that have been opened on non-loopback addresses need firewall rules to govern traffic.

0% (0 results, 4 rules selected)

☐ Dry run ☐ Fetch remote resources ☐ Remediate

SELinux rule prose information

```
<Rule xmlns=http://checklists.nist.gov/xccdf/1.2 id=
"xccdf_gov.nist_rule_SELinux_not_disabled_in_bootloader_configur
ation" selected="false">
  <title>Ensure SELinux is not disabled in bootloader
configuration</title>
  <description>Configure SELINUX to be enabled at boot time and
verify that it has not been overwritten by the grub boot
parameters.</description>
  <rationale>SELinux must be enabled at boot time in your grub
configuration to ensure that the controls it provides are not
overridden.</rationale>
  <complex-check …/>
</Rule>
```

Embedded HTML allowed

SELinux rule “check” information

```
<complex-check operator="OR">
  <complex-check operator="AND">
    <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
      <check-content-ref
href="CIS_Red_Hat_Enterprise_Linux_7_Benchmark_v2.1.0-oval.xml"
name="oval:gov.nist.redhat_redhat_enterprise_linux_7:def:1058"/>
    </check>
    <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
      <check-content-ref
href="CIS_Red_Hat_Enterprise_Linux_7_Benchmark_v2.1.0-oval.xml"
name="oval:gov.nist.redhat_redhat_enterprise_linux_7:def:1059"/>
    </check>
  </complex-check>
  <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
    <check-content-ref
href="CIS_Red_Hat_Enterprise_Linux_7_Benchmark_v2.1.0-oval.xml"
name="oval:gov.nist.redhat_redhat_enterprise_linux_7:def:1060"/>
</check>
</complex-check>
```

XCCDF authoring issues

- Long and repetitive identifiers
- Lots of namespace URIs
- Verbose and complicated check expressions

Observation

XCCDF versatility/expressiveness results in author-friendliness

Idea

Sacrifice some versatility/expressiveness to make authoring easier

Platform Fragmentation

- Occurs when multiple entities customize the same system component in differing ways
- Component can be hardware, operating system (OS), or software app
- Result is a "Wild West" for checklist developers
- Big IoT problem
- Also applies to Linux

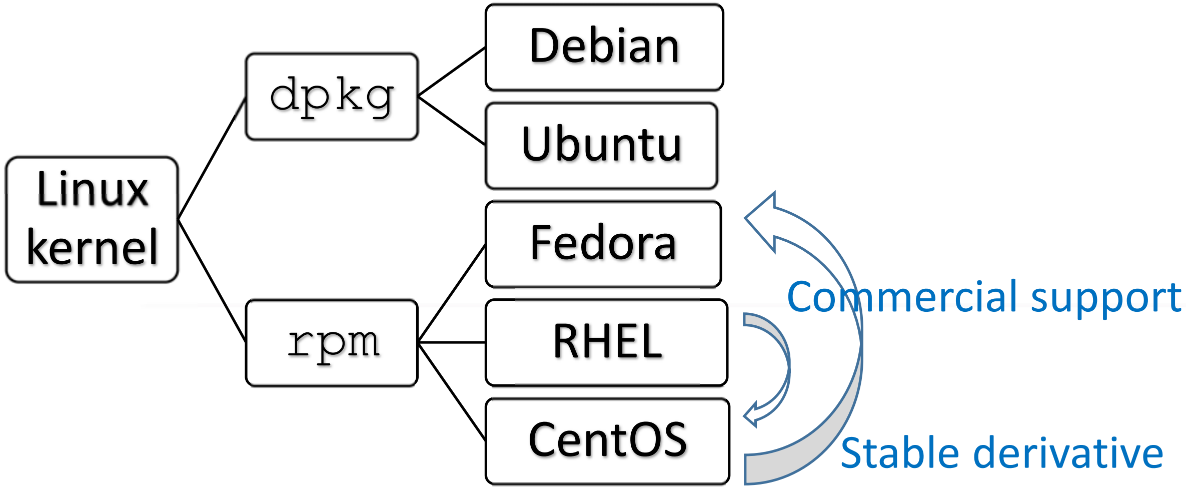
By Martin SoulStealer (Flickr: Mexican Standoff), via Wikimedia Commons

National Institute of Standards and Technology
U.S. Department of Commerce

Example: Firewall rule

- “Ensure iptables is installed”
- Iptables bundled with Linux kernel – common to most Linux systems
- But package manager needed to check compliance
- Not all Linux distributions use the same package manager
- Is this RHEL7 rule reusable for other Linux varieties?
- XCCDF of limited help
    - Support for associations between rules
    - But not platform relationships

Linux shared components



XCCDF checklists don’t convey these relationships!

Outline

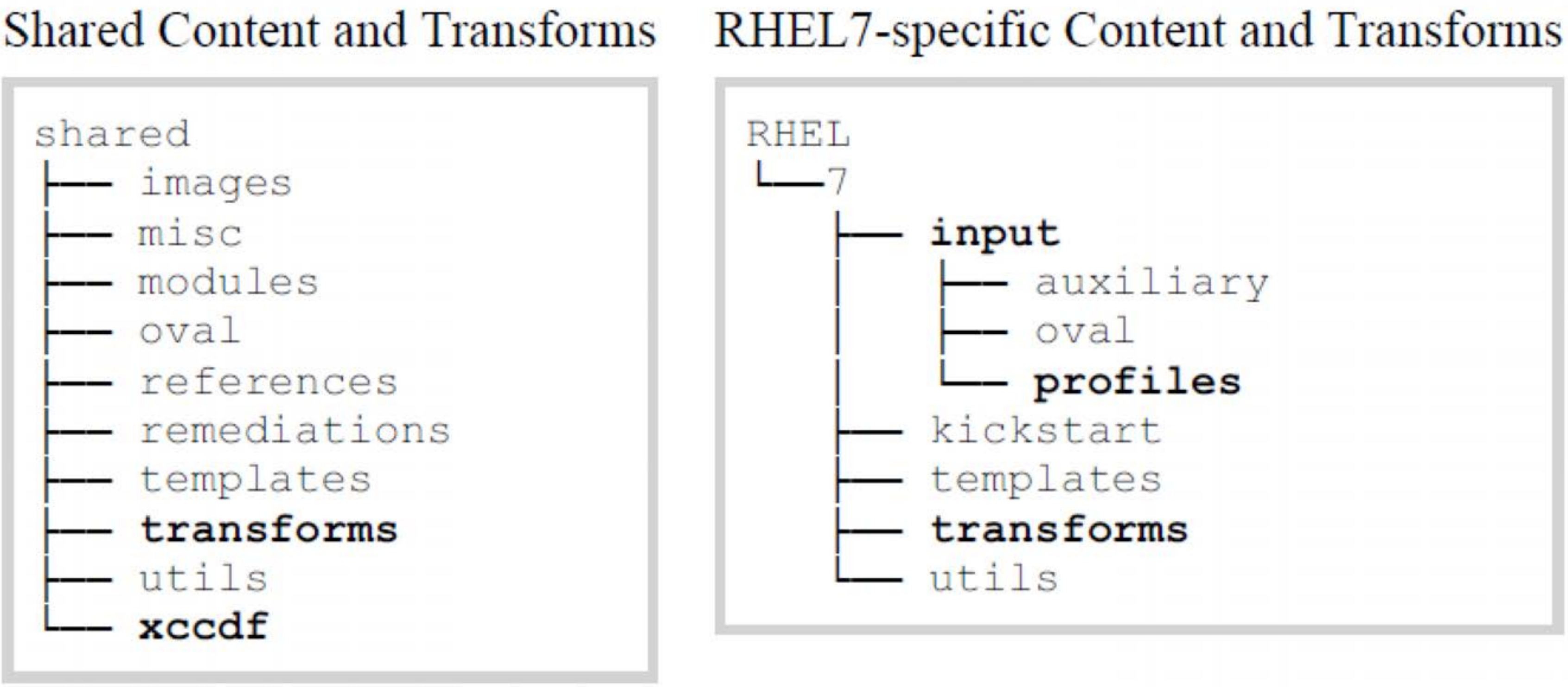
- Security Configuration checklists
- Extensible Configuration Checklist Description Format (XCCDF)
- **Show how the SCAP Security Guide project handles XCCDF authoring challenges**
- Darwin Information Typing Architecture (DITA) as an alternative
- Conclusions

National Institute of Standards and Technology
U.S. Department of Commerce

SCAP Security Guide (SSG)

- GitHub project led by Red Hat
- Output: SCAP content (including XCCDF and OVAL) for Linux OS varieties, software
- Widely used in industry and government
- Source code
  - XML files authored using a shorthand tag set
    - Relies on simplifying assumptions for XCCDF
  - Scripts and XSLT for generating SCAP content

SSG source code structure



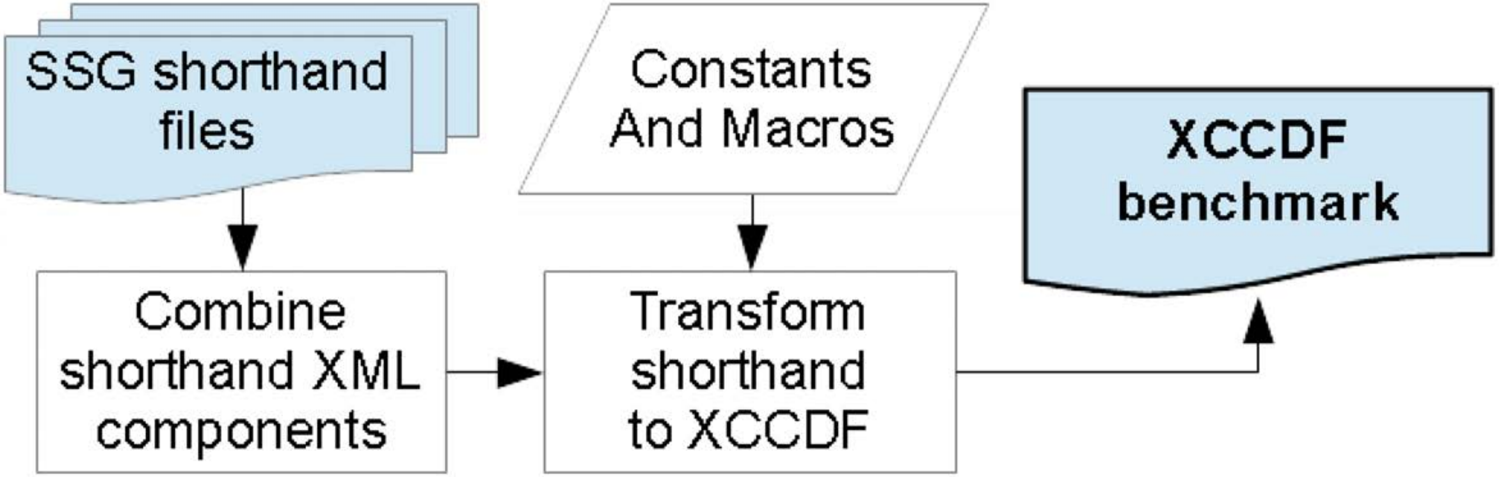
Shared Content and Transforms

```
shared
├── images
├── misc
├── modules
├── oval
├── references
├── remediations
├── templates
├── transforms
├── utils
└── xccdf
```

RHEL7-specific Content and Transforms

```
RHEL
└─7
    ├── input
    │   ├── auxiliary
    │   ├── oval
    │   └── profiles
    ├── kickstart
    ├── templates
    ├── transforms
    └── utils
```

Shorthand to XCCDF



SSG shorthand files

Constants And Macros

XCCDF benchmark

Combine shorthand XML components

Transform shorthand to XCCDF

SSG implementation approach

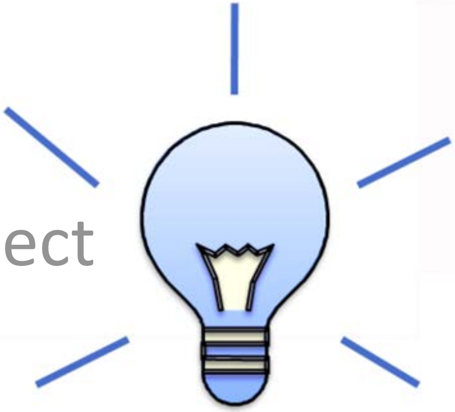
Issue	Solution
XCCDF author-unfriendliness	Shorthand XML, XSLT
Inline content fragment reuse	Shorthand tagset elements, e.g., `<product-name-macro/>`
Structural content reuse	Source code modularization, XSLT

Disadvantages

- One-off solutions unique to SSG, not interoperable with other security content authoring projects or XML content management tools
- Complicated SSG build process a barrier to potential contributors
- No formal schema for shorthand XML
- Reuse strategy lacks flexibility

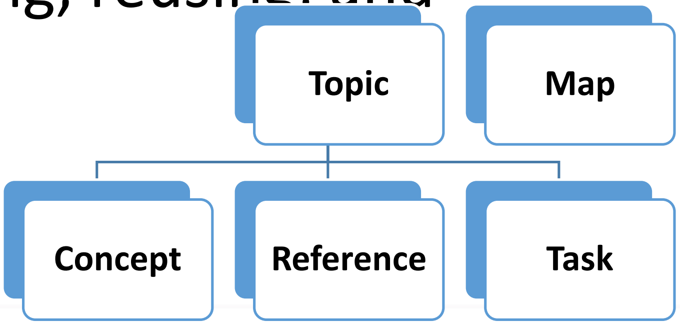
Outline

- Security Configuration checklists
- Extensible Configuration Checklist Description Format (XCCDF)
- Show how the SCAP Security Guide project handles XCCDF authoring challenges
- **Darwin Information Typing Architecture (DITA) as an alternative**
- Conclusions



DITA

- *Architecture* for creating, managing, reusing, and delivering technical content
  - Component content management
- Primary building blocks
  - Topic – reusable information chunk
  - Map – collection of topics and/or maps
- Supports inline element and fragment reuse
- Reference implementation – DITA Open Toolkit
  - Used by many third-party XML authoring and content delivery tools
  - Extensible plug-in architecture


Topic
Map
Concept
Reference
Task

Defining a new information type

- Configuration
  - Create document shell (XML schema) to support content authoring and validation
  - Must follow guidelines in DITA standard and reuse existing schema modules
- Specialization
  - Define constraints on an existing topic or map type
  - Specializations inherit all functionality of base type
    - Lowers implementation costs
    - No other XML framework has this benefit

Implementing DITA specialization

1. Design tag set for new topic or map type
2. Define specialization hierarchies
   - This is what drives DITA processing
   - Map elements from specialized type to equivalents in base type
   - Specified as default values in `@class` attribute
   - Invisible to content authors
3. Create document shell (configuration)
4. Implement new processing logic
   - Typically XSLT to transform DITA XML to delivery format (XCCDF in our case)
   - Or CSS for rendering in authoring tool or browser
   - Processing logic for base type automatically inherited

Now let’s create a DITA “rule” element type

By Strobridge Litho. Co., Cincinnati & New York.
Restoration by trialsanderrors, via Wikimedia Commons.

- Specialization of built-in “concept” topic type
- Transformable to XCCDF `<Rule>` element
- Simplifying assumptions:
  - `@selected=“false”`
  - OVAL used for checking
  - No automated remediation

SELinux rule

```
<rule id="SELinux_not_disabled_in_bootloader_configuration">
  <title>Ensure SELinux is not disabled in bootloader
configuration</title>
  <rulebody>
    <description>Configure SELINUX to be enabled at boot time and verify
that it has not been overwritten by the grub boot parameters.
    </description>
    <rationale>SELinux must be enabled at boot time in your grub
configuration to ensure that the controls it provides are not
overridden.</rationale>
    <check>
      <OR>
        <AND>
          <oval href="oval/1058.dita"/>
          <oval href="oval/1059.dita"/>
        </AND>
        <oval href="oval/1060.dita"/>
      </OR>
    </check>
  </rulebody>
</rule>
```

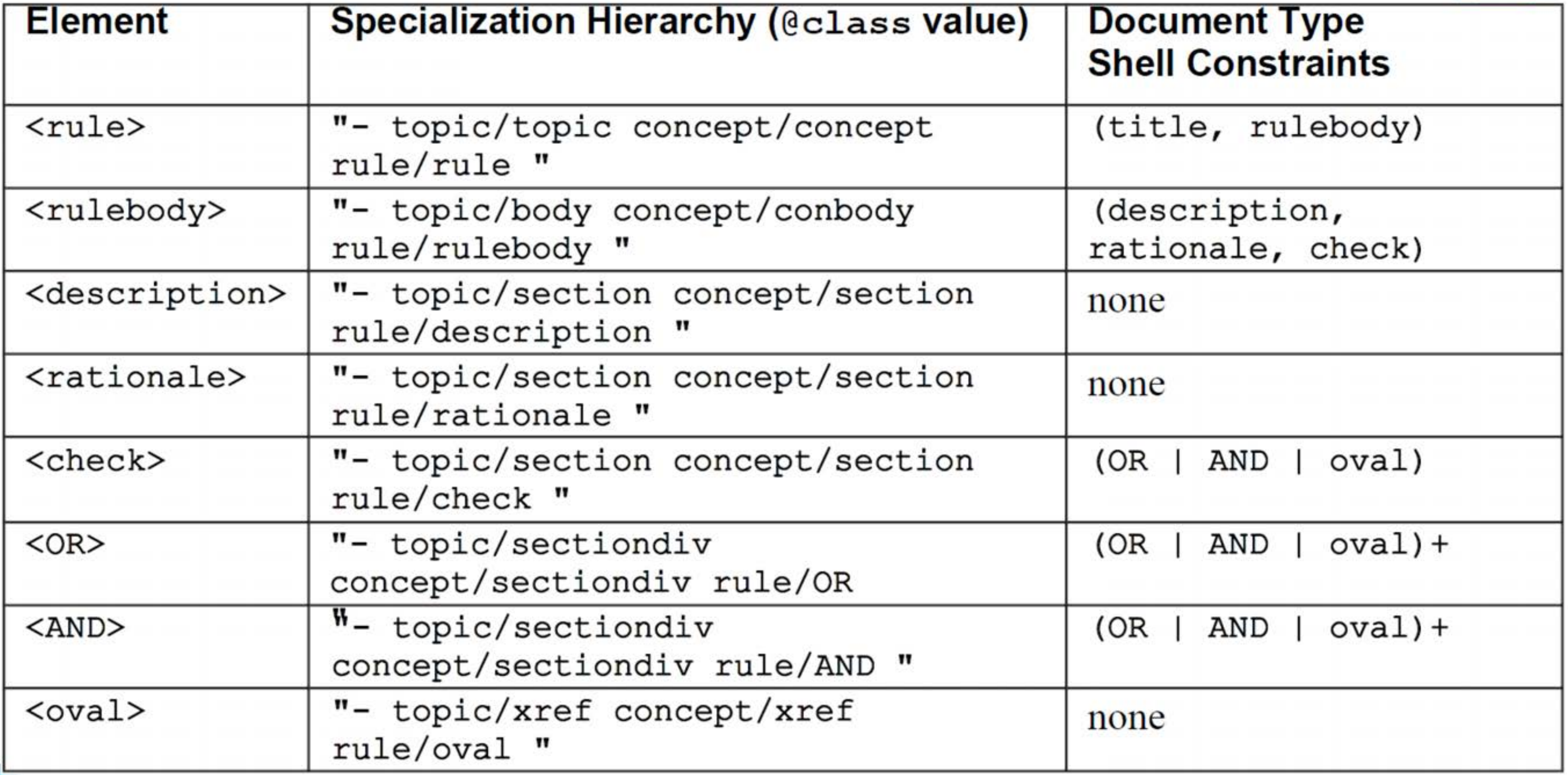
Generalized as “concept” topic

```
<concept id="SELinux_not_disabled_in_bootloader_configuration">
  <title>Ensure SELinux is not disabled in bootloader
configuration</title>
  <conbody>
    <section>Configure SELINUX to be enabled at boot time and verify
that it has not been overwritten by the grub boot parameters.
    </section>
    <section>SELinux must be enabled at boot time in your grub
configuration to ensure that the controls it provides are not
overridden.</section>
    <section>
      <sectiondiv>
        <sectiondiv>
          <xref href="oval/1058.dita"/>
          <xref href="oval/1059.dita"/>
        </sectiondiv>
        <xref href="oval/1060.dita"/>
      </sectiondiv>
    </section>
  </conbody>
</concept>
```

Specialization hierarchy for top-level `<rule>` element

`@class` value	DITA processor interpretation
`"- topic/topic concept/concept rule/rule "`	The `<rule>` element in the “rule” element type specializes `<concept>` from the “concept” element type, which in turn specializes `<topic>` from the “topic” element type.

Implementation

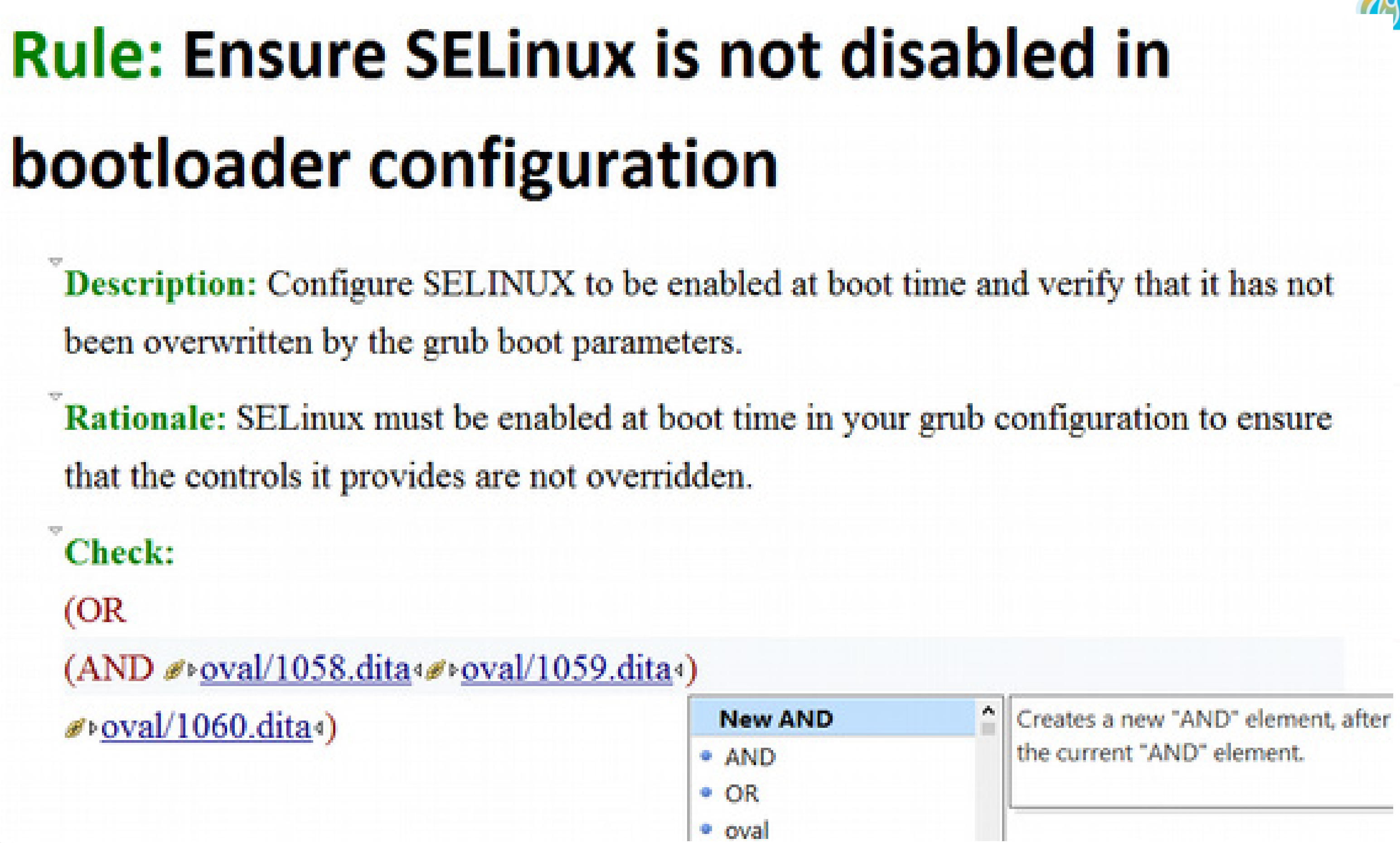


Element	Specialization Hierarchy (@class value)	Document Type Shell Constraints
<rule>	"- topic/topic concept/concept rule/rule "	(title, rulebody)
<rulebody>	"- topic/body concept/conbody rule/rulebody "	(description, rationale, check)
<description>	"- topic/section concept/section rule/description "	none
<rationale>	"- topic/section concept/section rule/rationale "	none
<check>	"- topic/section concept/section rule/check "	(OR | AND | oval)
<OR>	"- topic/sectiondiv concept/sectiondiv rule/OR	(OR | AND | oval)+
<AND>	"- topic/sectiondiv concept/sectiondiv rule/AND "	(OR | AND | oval)+
<oval>	"- topic/xref concept/xref rule/oval "	none

What the DITA processor cares about

```
<topic …
class="- topic/topic concept/concept rule/rule ">
  <title class="- topic/title ">Ensure SELinux is not disabled
in bootloader configuration</title>
  <body class="- topic/body  concept/conbody rule/rulebody ">
    <section class="- topic/section concept/section 
rule/description ">Configure SELINUX to be enabled at boot time
and verify that it has not been overwritten by the grub boot
parameters.</section>
    <section class="- topic/section concept/section 
rule/rationale ">SELinux must be enabled at boot time in your
grub configuration to ensure that the controls it provides are
not overridden.</section>
    <section class="- topic/section concept/section rule/check 
">
      …
    </section>
  </body>
</topic>
```

WYSIWYG in Oxygen XML Editor



Rule: Ensure SELinux is not disabled in bootloader configuration

Description: Configure SELINUX to be enabled at boot time and verify that it has not been overwritten by the grub boot parameters.

Rationale: SELinux must be enabled at boot time in your grub configuration to ensure that the controls it provides are not overridden.

Check:

(OR

(AND oval/1058.dita oval/1059.dita)

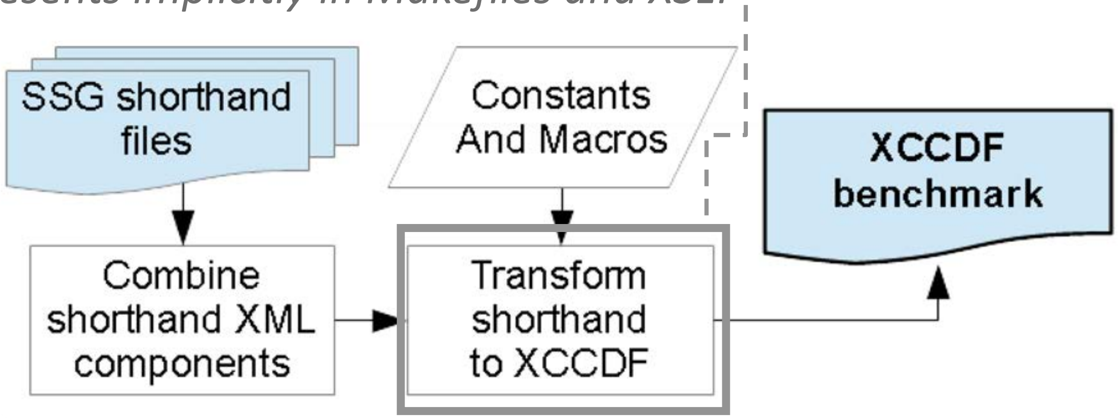
oval/1060.dita)

Creates a new "AND" element, after the current "AND" element.

Other specializations

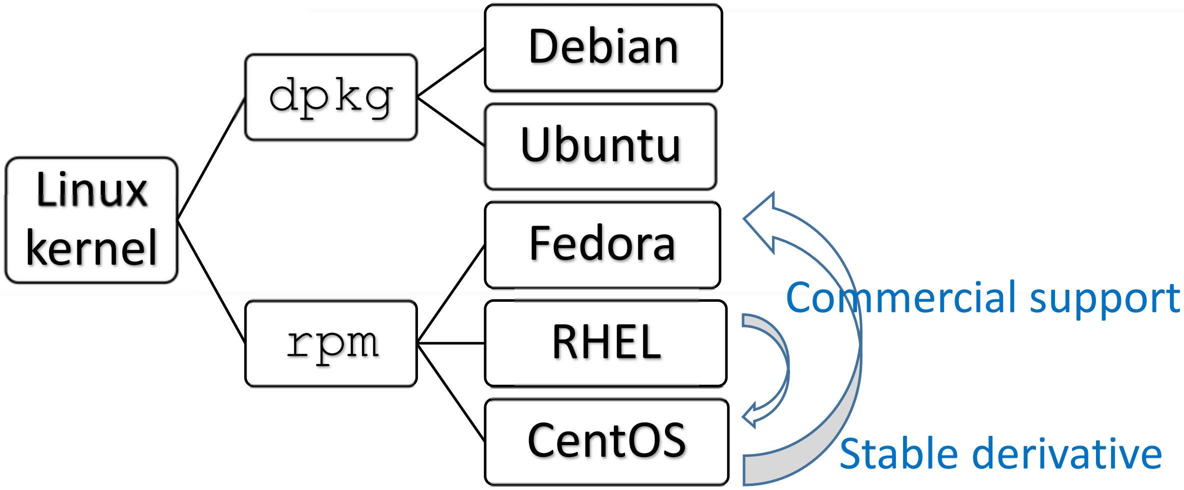
XCCDF element	DITA base type
`<Profile>`	Map
`<Group>`	Map
`<Value>`	Topic
`<Benchmark>`	Map

Explicitly represent the overall checklist structure, which the SSG build system represents implicitly in Makefiles and XSLT


SSG shorthand files
Constants And Macros
XCCDF benchmark
Combine shorthand XML components
Transform shorthand to XCCDF

Structural content reuse

DITA maps can slice and dice a repository of XCCDF resources for creating benchmarks covering a wide variety of platforms, more flexibly than the SSG directory-based approach



Linux kernel

`dpkg`

`rpm`

Debian

Ubuntu

Fedora

RHEL

CentOS

Commercial support

Stable derivative

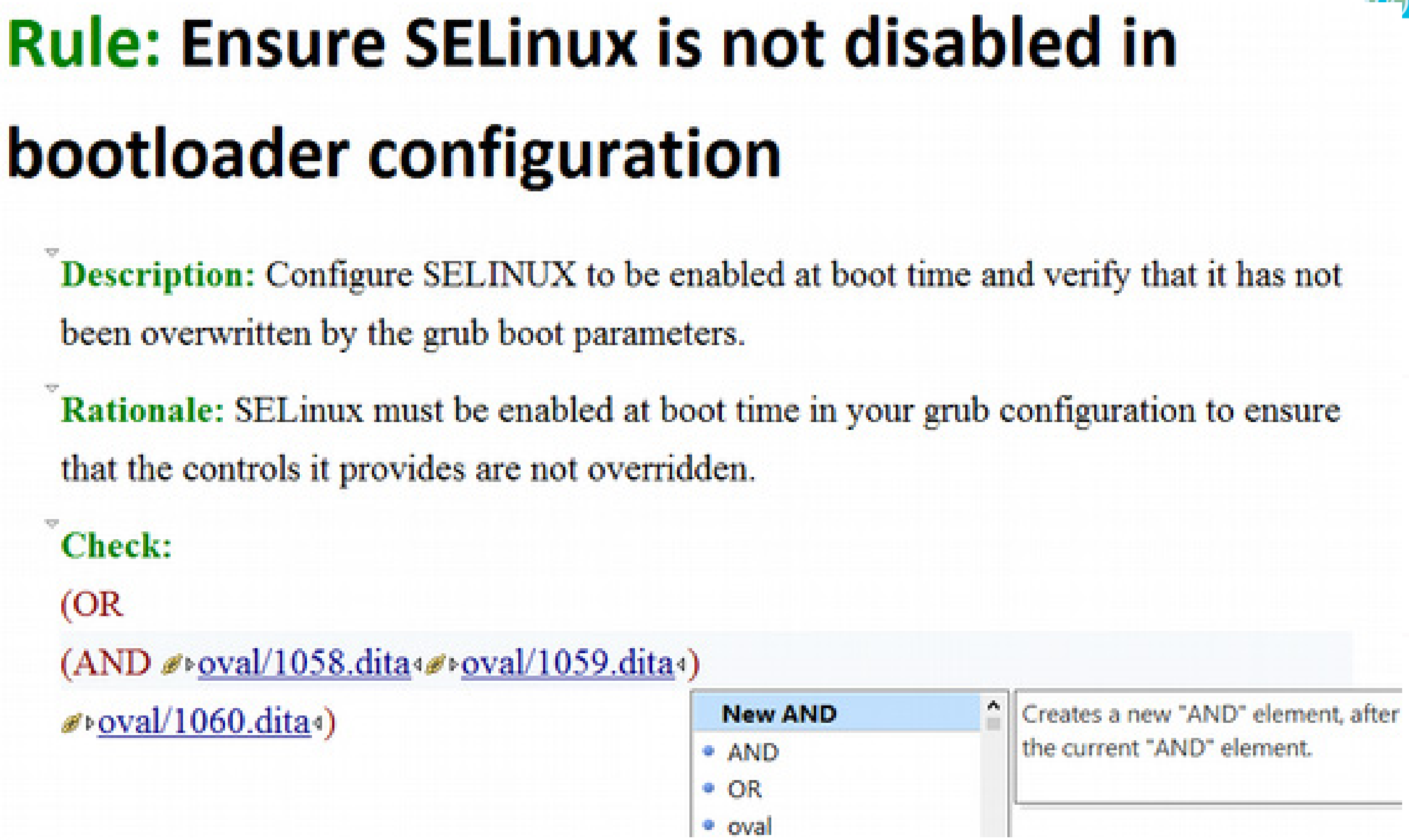
Inline content reuse

```
<benchmark class="- map/map benchmark/benchmark ">
  <title>Benchmark for <ph keyref="product_name"/>
  </title>
  <keydef keys="product_name">
    <topicmeta><keywords><keyword>Red Hat Enterprise 
Linux 7</keyword></keywords></topicmeta></keydef>
  <intro href="introduction.dita" class=
"- map/topicref benchmark/intro "/>
…
</benchmark>
```

introduction.dita

```
…
<p>This document provides prescriptive guidance for
establishing a secure configuration for
<ph keyref="product_name"/> systems.</p>
…
```

Code reuse



Rule: Ensure SELinux is not disabled in bootloader configuration

Description: Configure SELINUX to be enabled at boot time and verify that it has not been overwritten by the grub boot parameters.

Rationale: SELinux must be enabled at boot time in your grub configuration to ensure that the controls it provides are not overridden.

Check:

```
(OR
(AND oval/1058.dita oval/1059.dita)
oval/1060.dita)
```

Creates a new "AND" element, after the current "AND" element.

Cascading Style Sheet (CSS)

```
rule>title:before {
  content: 'Rule: ';
  color: green;}
description:before {
  content:
'Description: ';
  font-weight: bold;
  color: green;}
rationale:before {
  content: 'Rationale:
';
  font-weight: bold;
  color: green;}
check:before {
  content: 'Check: ';
  font-weight: bold;
  color: green;  }
```

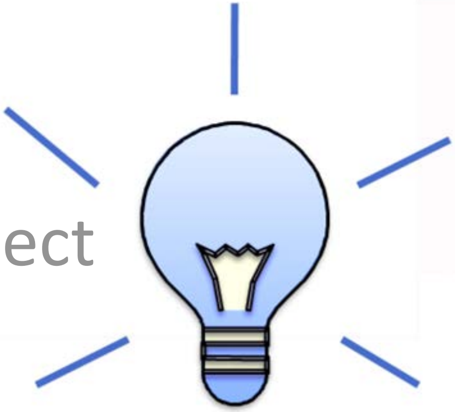
```
AND:before {
  content: '(AND ';
  color: maroon;}
AND:after {
  content: ')';
  color: maroon;}
OR:before {
  content: '(OR ';
  color: maroon;}
OR:after {
  content: ')';
  color: maroon;}
oval {
  content:
attr(href);}
```

DITA Open Toolkit

- Automatically merges topics, resolves content references
- Specialization-aware DITA processor
- An XCCDF plug-in would
  - Use built-in transformation code by default
  - Implement any new functionality based on matches against `@class` attribute
  - Require less XSLT code than the SCAP Security Guide build process

Outline

- Security Configuration checklists
- Extensible Configuration Checklist Description Format (XCCDF)
- Show how the SCAP Security Guide project handles XCCDF authoring challenges
- Darwin Information Typing Architecture (DITA) as an alternative
- **Conclusions**



Some Limitations

- Focus exclusively on XCCDF authoring
  - Ignored other options for solving platform fragmentation problem
    - J. Lubell and T. Zimmerman. “The Challenge of Automating Security Configuration Checklists in Manufacturing Environments.” In *Critical Infrastructure Protection XI*. M. Rice and S. Shenoi, Eds. Springer Berlin Heidelberg (2017).
- Case study limited to extremely simple checklist
- Simplifying assumptions might be unreasonable for many checklist developers

Summary

- Presented Linux configuration checklist scenario
- Discussed XCCDF authoring challenges and platform fragmentation problem
- Described SCAP Security Guide approach
- Proof of concept implementation of DITA “rule” specialized type
- Examples of other specialized types, DITA capabilities

Conclusion

- Platform fragmentation and author unfriendliness are barriers to XCCDF use
- DITA specialization is feasible and offers tangible benefits beyond SCAP Security Guide approach
- DITA's default processing of maps and key references should simplify implementation of the “Combine shorthand XML components” and “Transform shorthand to XCCDF” steps.
- Next step: implement an XCCDF DITA Open Toolkit plug-in
  - If successful, propose as contribution to SSG project