# SCAP Composer

## A DITA Open Toolkit Plug-in for Packaging Security Content

Joshua Lubell
National Institute of Standards and Technology

August 1, 2019

# Disclaimer

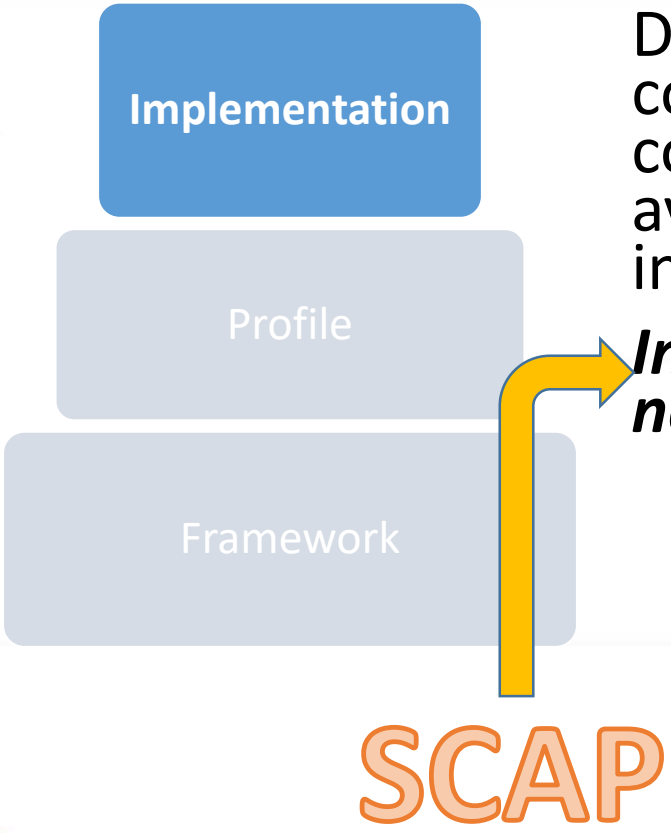
Certain commercial and third-party products are identified to help explain the research. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the products identified are necessarily the best available for the purpose.

# Outline of this talk

- SCAP[1] overview
- Today's SCAP challenges
- SCAP source data stream collections
- SCAP Composer
  - Alternative source data stream collection model
  - Implementation using DITA[2] Open Toolkit

[1]Security Content Automation Protocol
[2]Darwin Information Typing Architecture

# Cyber-risk management layer cake




Deployment of safeguards or countermeasures to protect the confidentiality, integrity, and availability of a system and its information

***Includes knowing the state of your network***

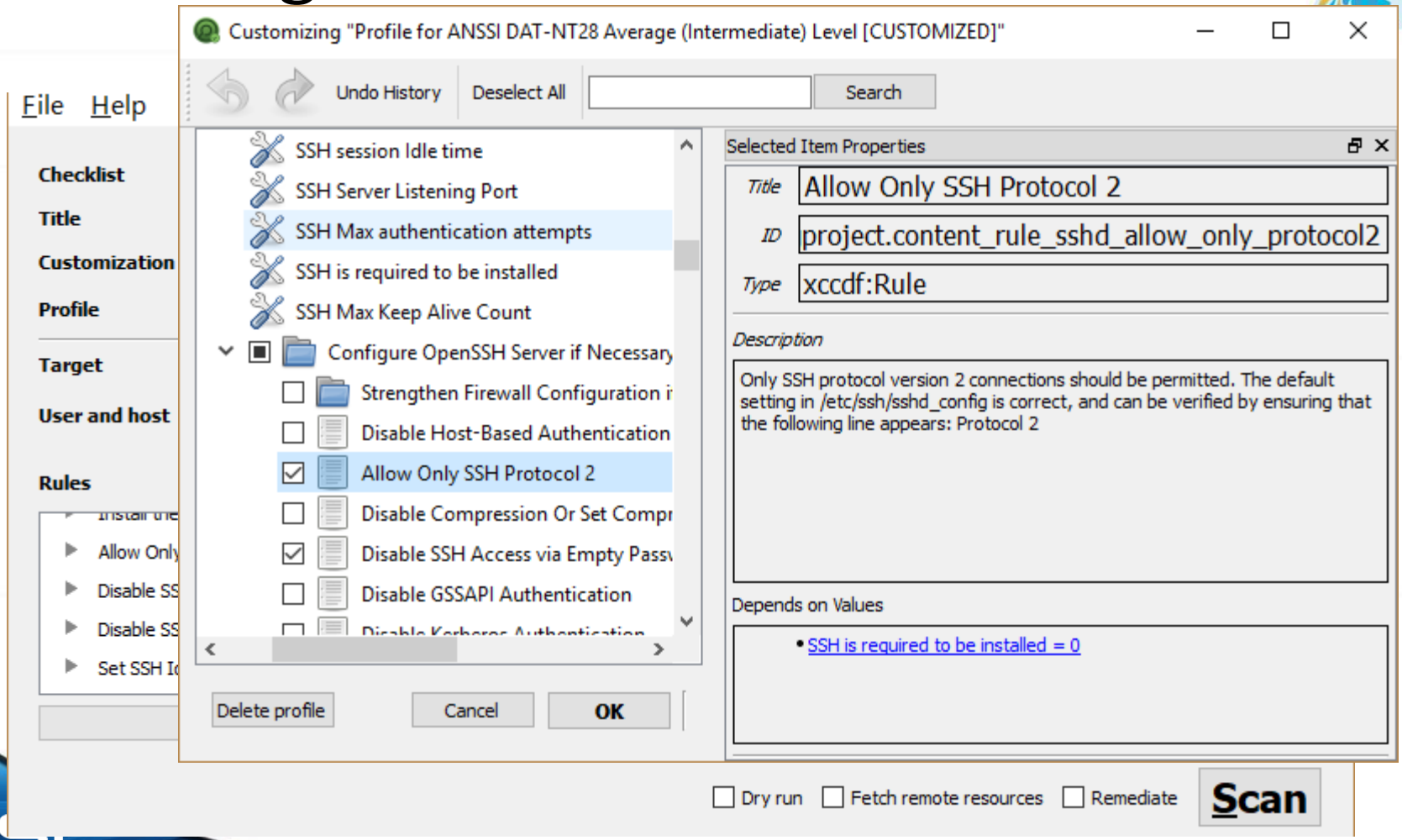
- ***What endpoints are connected to it?***
- ***Are those endpoints are authorized to be on the network***
- ***What software is installed on each endpoint?***
- ***How is that software is configured?***

# Security Content Automation Protocol (SCAP)

- Created as a NIST[1] program in 2006
- Suite of specifications for expressing, exchanging, and processing security content using standardized XML[2] formats
- Component languages
  - Extensible Configuration Checklist Description Format (XCCDF) – represents security configuration rules
  - Open Vulnerability Assessment Language (OVAL) – represents system configuration information, tests and states
  - XCCDF often uses OVAL to determine satisfaction of rule criteria
- SCAP tools use component content to determine if software vulnerabilities and misconfigurations exist on a specific managed endpoint
- By identifying endpoints with these problems, organizations can make informed remediation decisions

[1]National Institute of Standards and Technology
[2]Extensible Markup Language

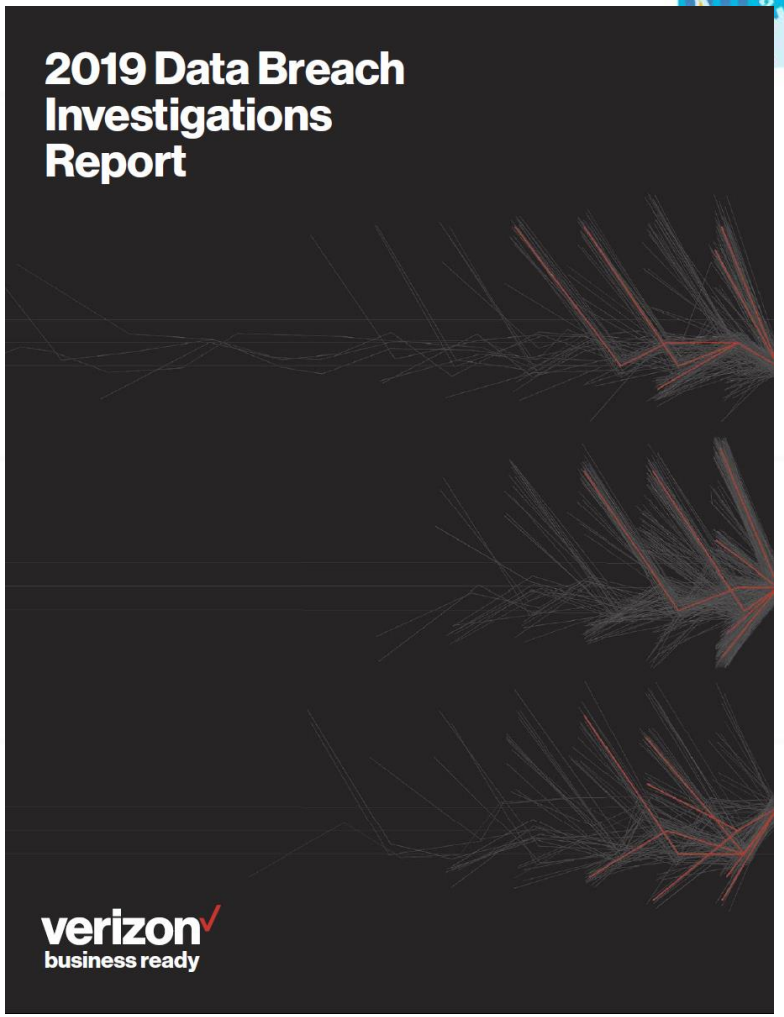
# SCAP-conforming scanning and tailoring software



# Why SCAP?

Same attacks still successful because same weaknesses still exist

- So work on low-hanging fruit first:
  - Continuous monitoring
  - Least privilege
  - Software inventory management
  - Boundary protection
  - Multi-factor authentication
  - Awareness and training
  - Physical security

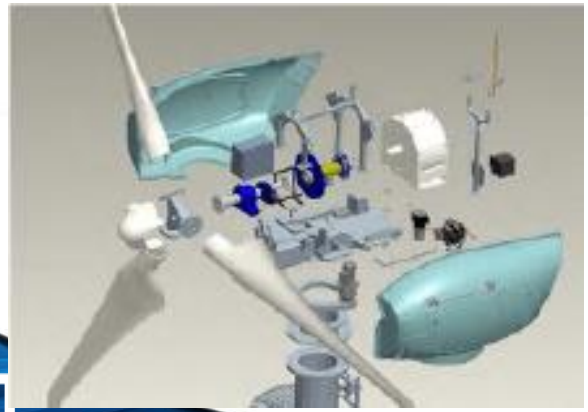


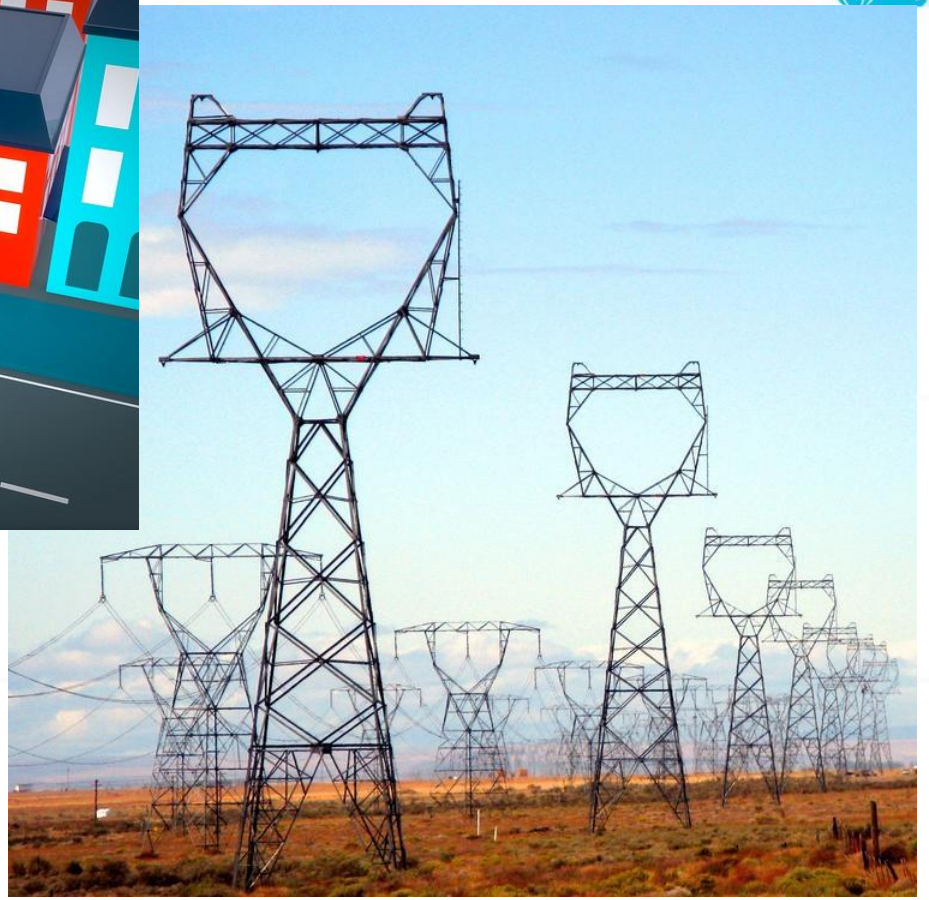
# Outline of this talk

- SCAP overview
- Today's SCAP challenges
- SCAP source data stream collections
- SCAP Composer
  - Alternative source data stream collection model
  - Implementation using DITA Open Toolkit

# The cybersecurity landscape has changed since 2006…

# Critical Infrastructure

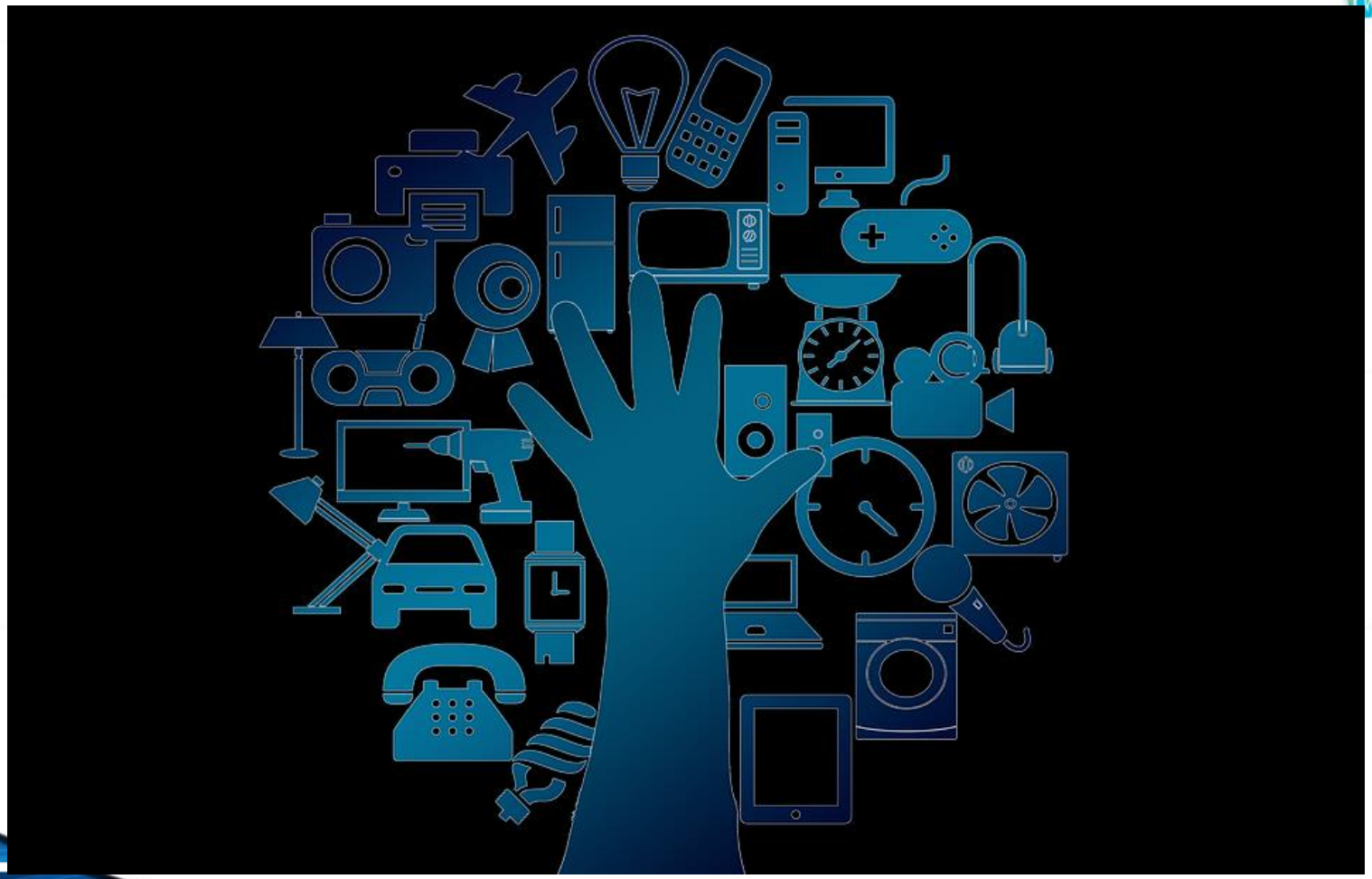




Credit: Bonneville Power Administration/Department of Energy

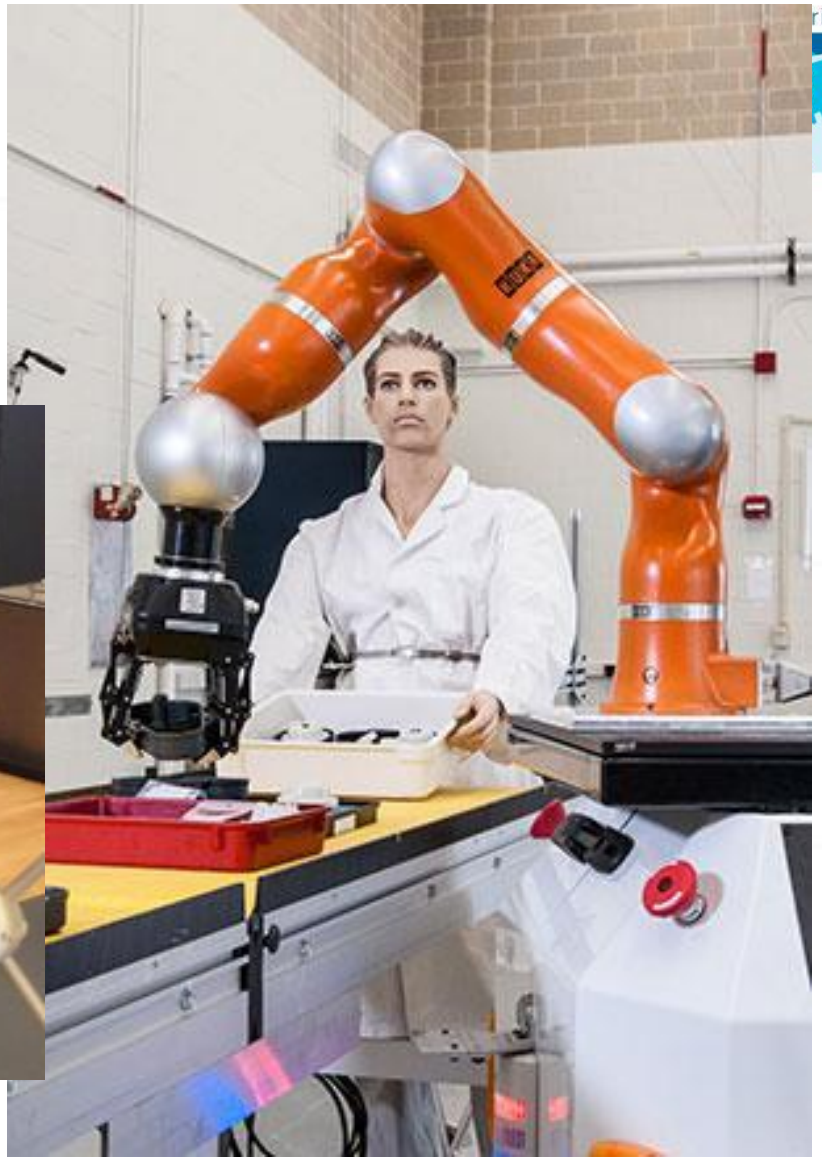
# Mobile Device Management

# Internet of Things



# Advanced Manufacturing

Credit: America Makes



# Some Implications

- More diversity of endpoint types
  - Not just servers, desktops, laptops
  - Lines blurring between hardware and software
  - Platform fragmentation
- Increased impact of successful cyberattacks
  - Need to protect system from its environment ***and*** environment from system
- More diversity of and increased need for SCAP content developers
  - No single category of stakeholders can do it alone

# …and SCAP content has struggled to keep pace

# Department of Defense Security Technical Information Guides

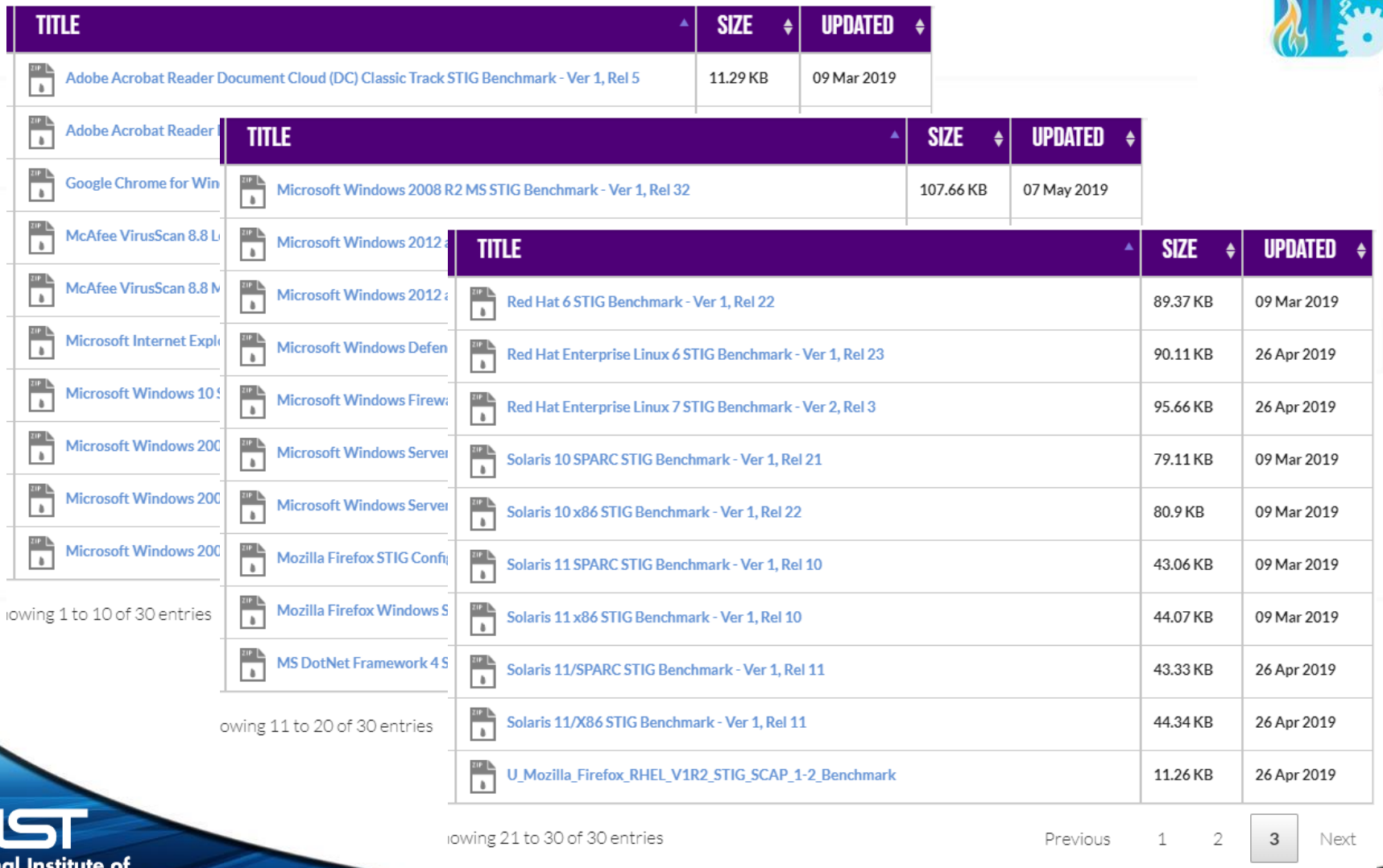
| TITLE | SIZE | UPDATED |
|---|---|---|
| 2014-09-24 DoD CIO Memo - Interim Guidance on the Use of DoD PIV Derived PKI Credentials on Unclassified Commercial Mobile | 185.15 | 30 Nov 2018 |
| 2015-05-06 DoD CIO Memo - Interim Guidance for Im Credentials on Unclass CMDs w/ Attachment and FAQs | | |
| 2016-04-21 DoD CIO Memo - Use of Wearable Device Spaces with FAQ | | |
| A10 Networks Application Delivery Controller (ADC) | | |
| A10 Networks Application Delivery Controller (ADC) | | |
| A10 Networks Application Delivery Controller (ADC) Overview, Ver 1 | 86.24 KB | 30 Nov 2018 |
| A10 Networks Application Delivery Controller (ADC) STIG Ver 1 Release Memo | 70.89 KB | 30 Nov 2018 |
| AAA SRG - Ver 1, Rel 1 | 353.63 KB | 20 May 2019 |
| Active Directory Domain STIG - Ver 2, Rel 12 | 624.63 KB | 09 Mar 2019 |
| Active Directory Domain STIG - Ver 2, Rel 13 | 625.32 | 26 Apr 2019 |

- [ ] Application Security (160) [+]
- [ ] Network/Perimeter/Wireless (145) [+]
- [ ] Sunset (117) [+]
- [ ] Operating Systems (76) [+]
- [ ] Security Content Application Protocols (SCAP) (32) [+]

## STIG TOPICS

- [ ] Application Security (160) [+]
- [ ] Network/Perimeter/Wireless (145) [+]
- [ ] Sunset (117) [+]
- [ ] Operating Systems (76) [+]
- [ ] Security Content Application Protocols (SCAP) (32) [+]
- [ ] Mobility (31) [+]
- [ ] STIG Viewing (8)
- [ ] DoD Cloud Computing Security (DCCS) (6)
- [ ] NIAP Protection Profiles (6)
- [ ] Cloud Security (4)
- [ ] Control Correlation Identifier (CCI) (4)
- [ ] Host-Based Security Systems (HBSS) (3) [+]
- [ ] Group Policy Objects (GPO) (2)
- [ ] STIG Compilations (2)
- [ ] STIG Policy (2)
- [ ] STIG Tools (2)
- [ ] Cross Domain Solutions (1)
- [ ] Vendor Process (1)

# SCAP 1.2 Content



| TITLE | SIZE | UPDATED |
|---|---|---|
| Adobe Acrobat Reader Document Cloud (DC) Classic Track STIG Benchmark - Ver 1, Rel 5 | 11.29 KB | 09 Mar 2019 |
| Adobe Acrobat Reader | | |
| Google Chrome for Win | | |
| McAfee VirusScan 8.8 L | | |
| McAfee VirusScan 8.8 M | | |
| Microsoft Internet Explo | | |
| Microsoft Windows 10 S | | |
| Microsoft Windows 200 | | |
| Microsoft Windows 200 | | |
| Microsoft Windows 200 | | |

owing 1 to 10 of 30 entries

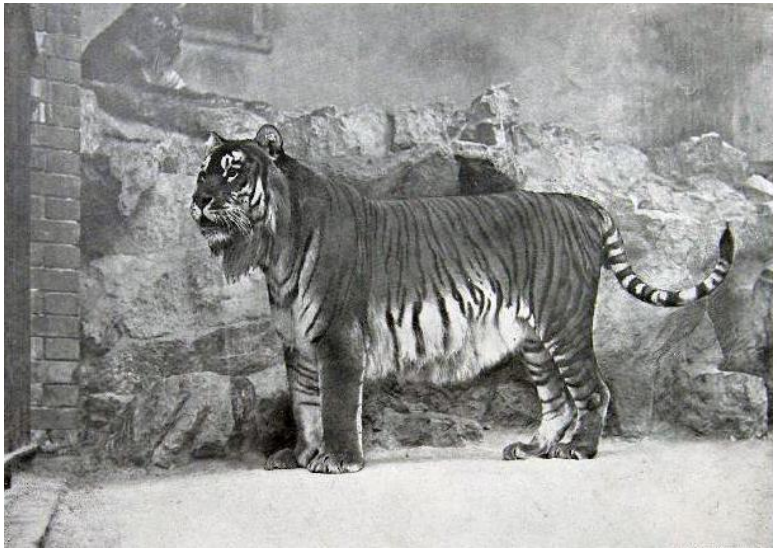
| TITLE | SIZE | UPDATED |
|---|---|---|
| Microsoft Windows 2008 R2 MS STIG Benchmark - Ver 1, Rel 32 | 107.66 KB | 07 May 2019 |
| Microsoft Windows 2012 | | |
| Microsoft Windows 2012 | | |
| Microsoft Windows Defen | | |
| Microsoft Windows Firewa | | |
| Microsoft Windows Server | | |
| Microsoft Windows Server | | |
| Mozilla Firefox STIG Confi | | |
| Mozilla Firefox Windows S | | |
| MS DotNet Framework 4 S | | |

owing 11 to 20 of 30 entries

| TITLE | SIZE | UPDATED |
|---|---|---|
| Red Hat 6 STIG Benchmark - Ver 1, Rel 22 | 89.37 KB | 09 Mar 2019 |
| Red Hat Enterprise Linux 6 STIG Benchmark - Ver 1, Rel 23 | 90.11 KB | 26 Apr 2019 |
| Red Hat Enterprise Linux 7 STIG Benchmark - Ver 2, Rel 3 | 95.66 KB | 26 Apr 2019 |
| Solaris 10 SPARC STIG Benchmark - Ver 1, Rel 21 | 79.11 KB | 09 Mar 2019 |
| Solaris 10 x86 STIG Benchmark - Ver 1, Rel 22 | 80.9 KB | 09 Mar 2019 |
| Solaris 11 SPARC STIG Benchmark - Ver 1, Rel 10 | 43.06 KB | 09 Mar 2019 |
| Solaris 11 x86 STIG Benchmark - Ver 1, Rel 10 | 44.07 KB | 09 Mar 2019 |
| Solaris 11/SPARC STIG Benchmark - Ver 1, Rel 11 | 43.33 KB | 26 Apr 2019 |
| Solaris 11/X86 STIG Benchmark - Ver 1, Rel 11 | 44.34 KB | 26 Apr 2019 |
| U_Mozilla_Firefox_RHEL_V1R2_STIG_SCAP_1-2_Benchmark | 11.26 KB | 26 Apr 2019 |

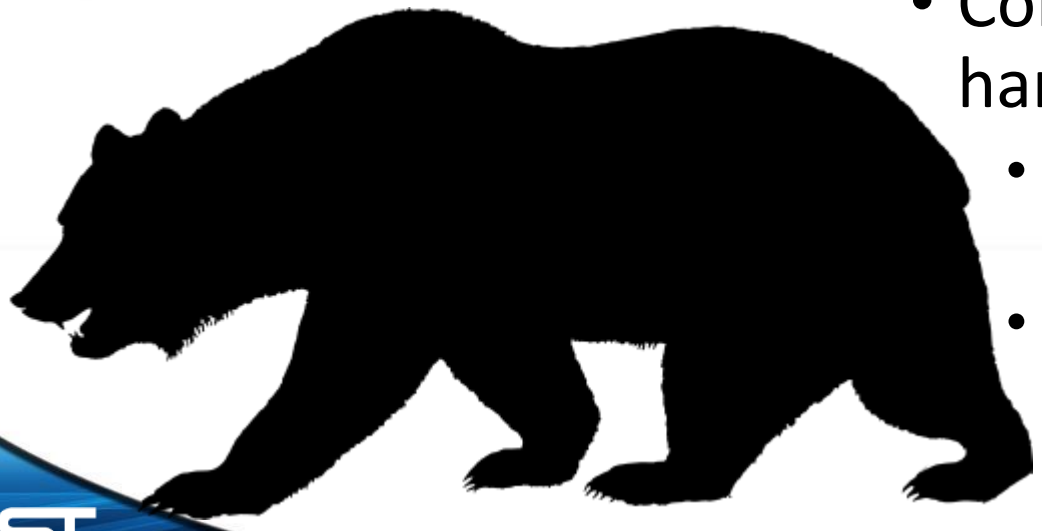
owing 21 to 30 of 30 entries  Previous 1 2 3 Next

# SCAP Content in the Wild



- Content repositories are hard to find/search
- And quality is variable
- And content is hard to reuse
  - Often needs to be adapted for a new endpoint type or usage scenario

# Why is this so?



- Limited coverage of networked devices other than servers, desktops, laptops
- Content creation is hard
  - Requires low-level system knowledge
  - SCAP XML languages are a bear for content authors

# Outline of this talk

- SCAP overview
- Today's SCAP challenges
- SCAP source data stream collections
- SCAP Composer
  - Alternative source data stream collection model
  - Implementation using DITA Open Toolkit

# Source Data Stream (SDS) Collection

***A good place to start for easing content authoring***

- Packages SCAP software input and output into *self-contained* and *reversible* bundle for easy deployment and sharing
- XML data model specified in NIST SCAP Technical Specification (Special Publication 800-126)

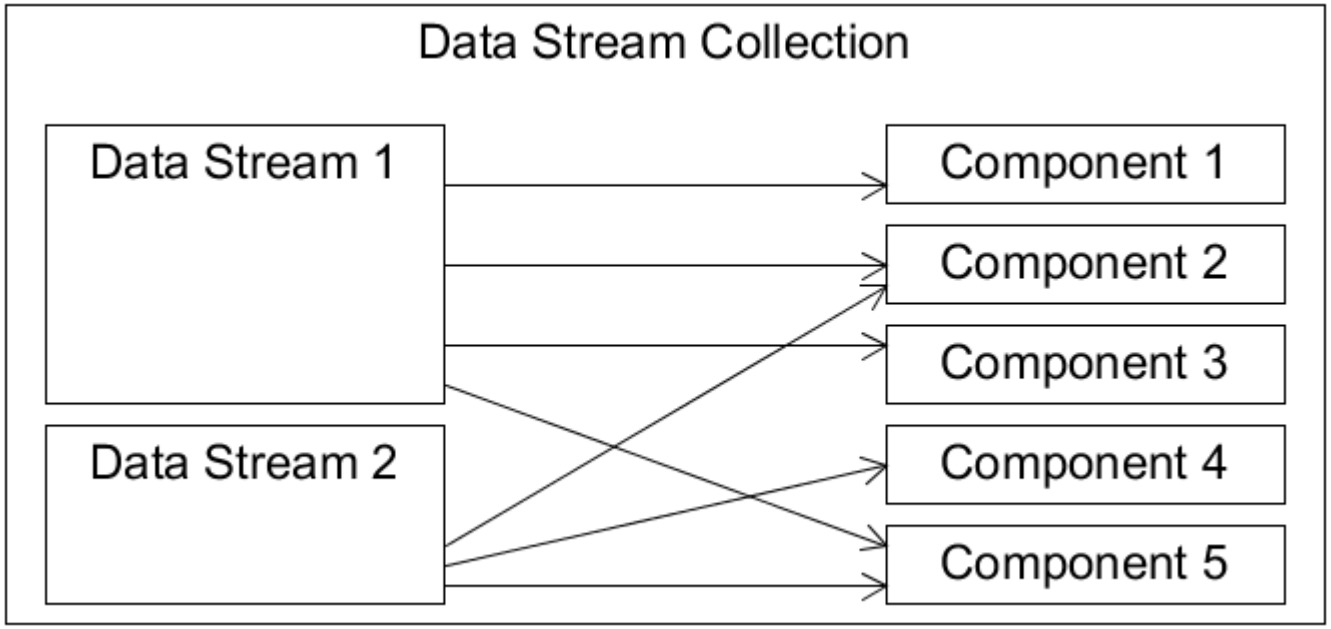
# Why self containment?


- Required when network connectivity and filesystem access restricted (often the case for industrial control systems)
- Promotes portability
- Enables digital signing

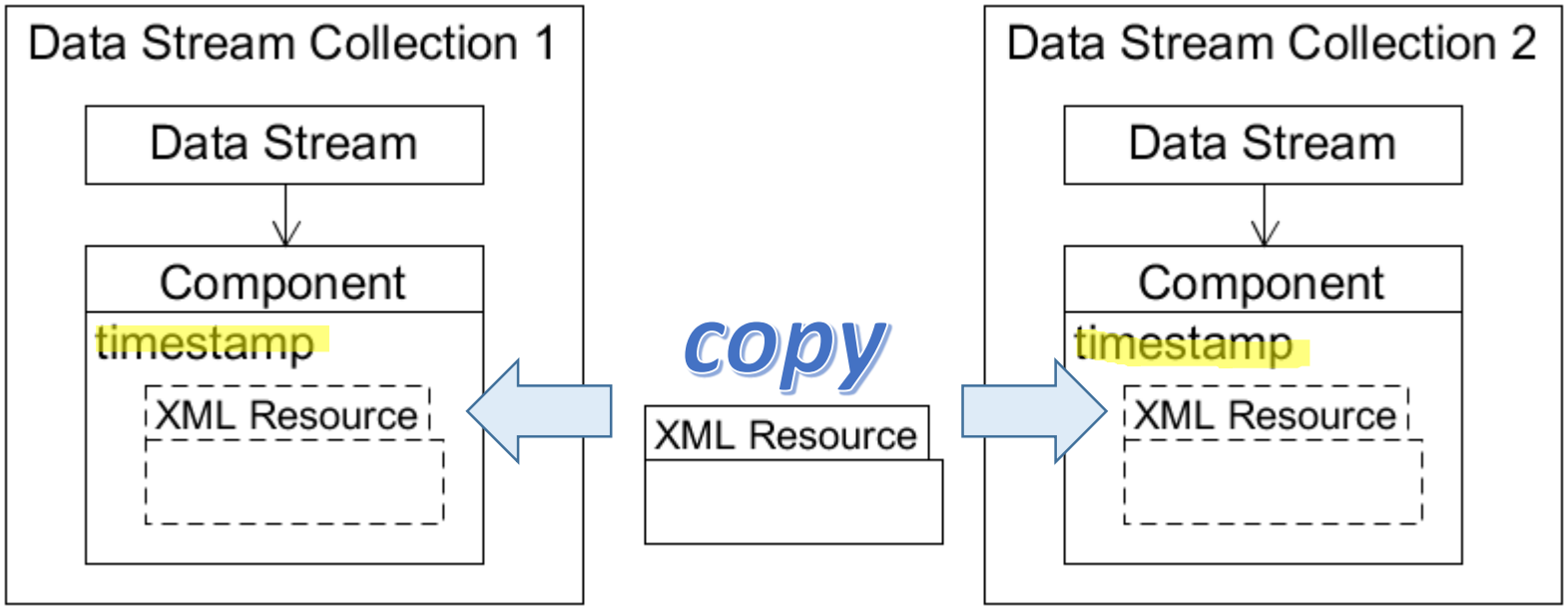
# Reversibility

- Any XML resource encapsulated in an SCAP component can be extracted and re-bundled into a new collection without modification to the XML
- Requires that XML resources the components contain are unmodified from their original states
- Promotes reuse

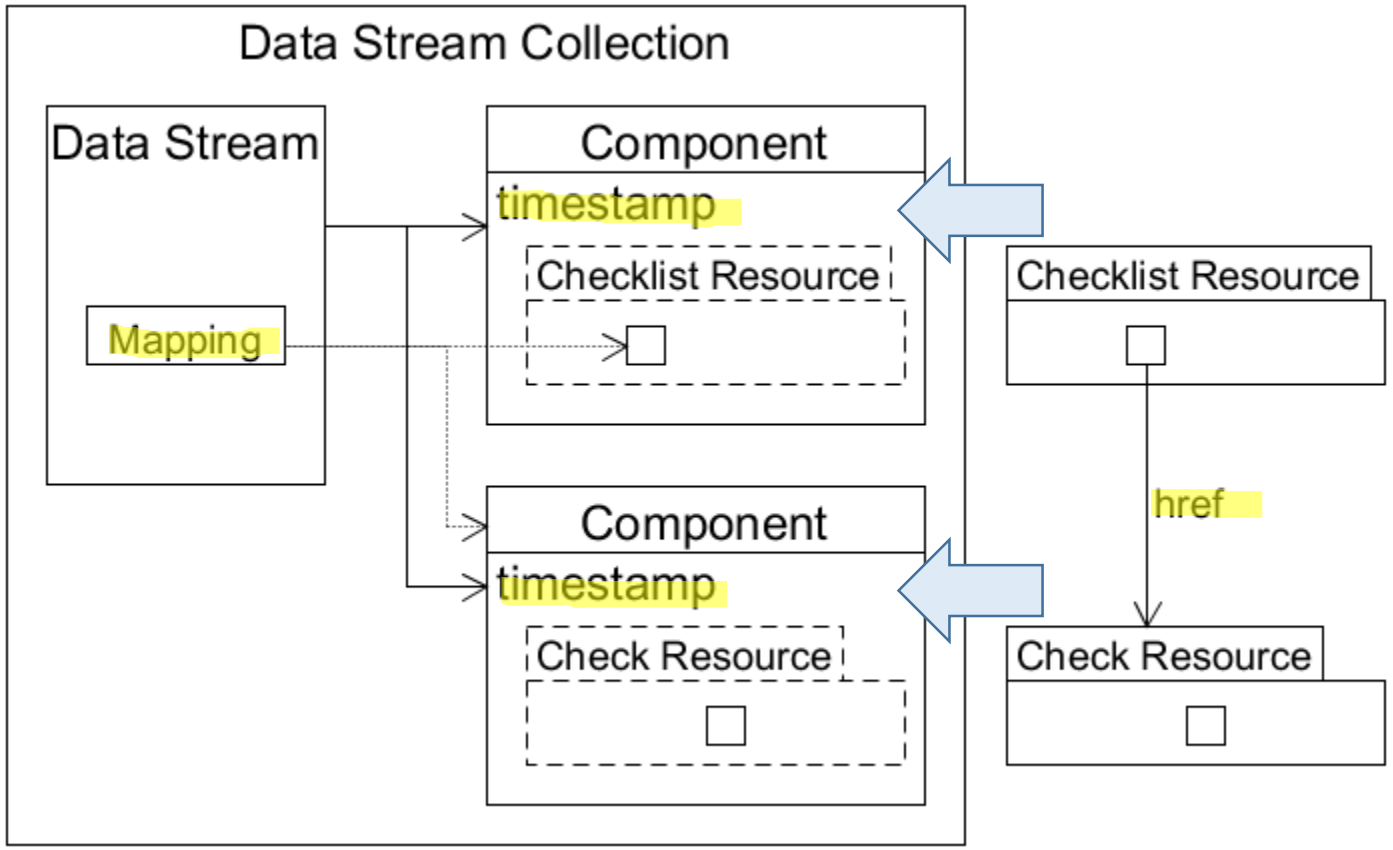
# SCAP source data stream collection



# Components reusing the same XML



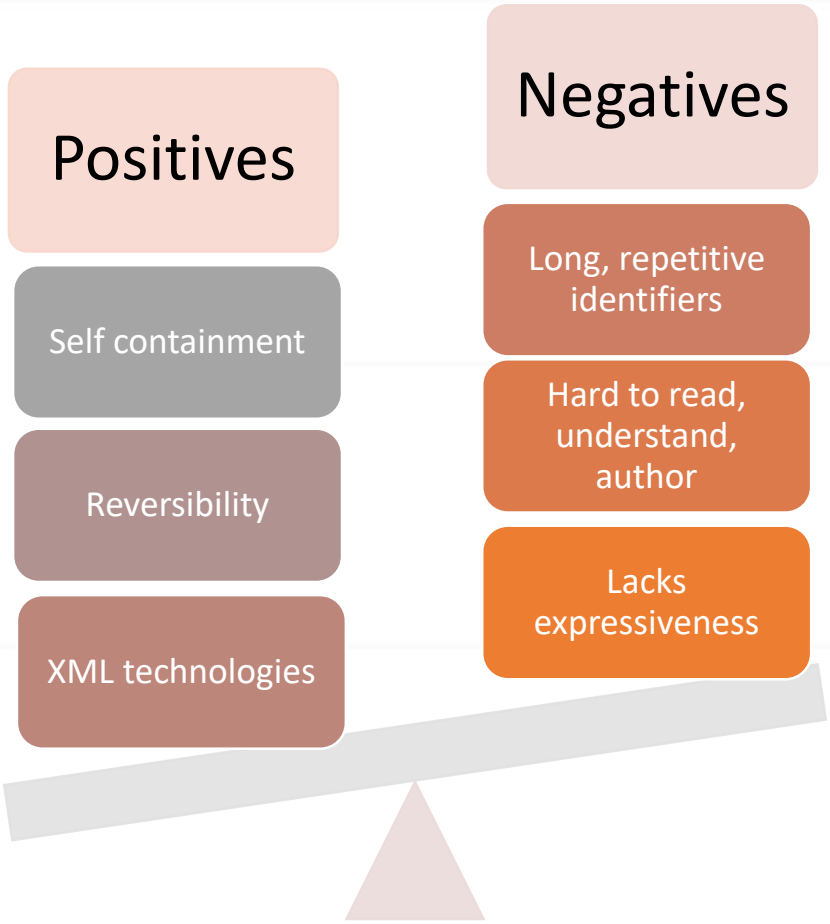
# Checklist resource with URI reference to a Check resource



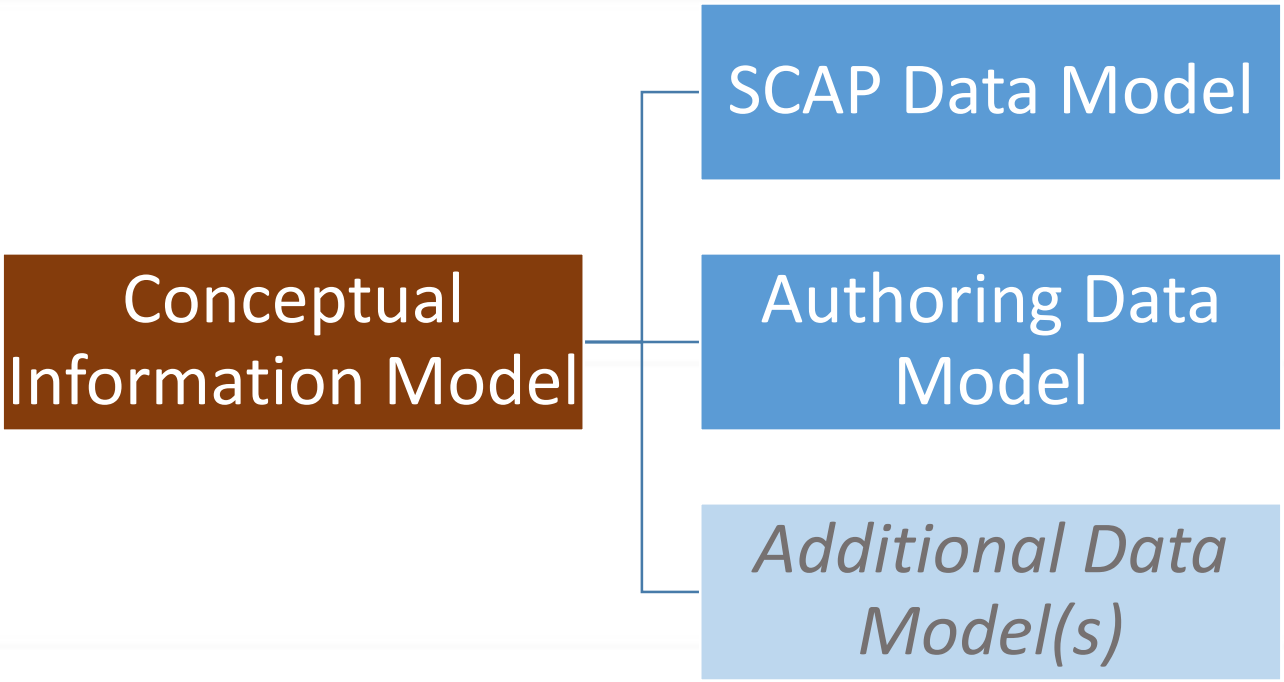
# The mapping in XML

| SCAP data model | Authoring-friendly data model |
| --- | --- |
| <sds:component-ref id="scap_com.example_cref_xccdf" xlink:href="#scap_com.example_comp_xccdf"><br><cat:catalog><br><cat:uri name="ssg-ubuntu1404-oval.xml" uri="#scap_com.example_cref_oval" /><br></cat:catalog><br></sds:component-ref> | <scapBenchmarkRef keyref="xccdf"><br><scapExternalLinks><br><scapUri keyref="oval"/><br></scapExternalLinks><br></scapBenchmarkRef> |

# SCAP data model tradeoffs



# One data model is not enough!




Lubell J (2018) A New SCAP Information and Data Model for Content Authors. Critical Infrastructure Protection XII, Springer International Publishing, Vol. 542, pp 127–146.

# Outline of this talk

- SCAP overview
- Today's SCAP challenges
- SCAP source data stream collections
- SCAP Composer
  - Alternative source data stream collection model
  - Implementation using DITA Open Toolkit

# SCAP Composer

- Uses DITA[1]
- Takes an incremental approach – limited scope
- Implemented with DITA Open Toolkit
  - Enables integration with NIST's SCAP Content Validation Tool
  - Deployable in most DITA authoring environments, or standalone with non-DITA XML editors

[1]Darwin Information Typing Architecture, Organization for the Advancement of Structured Information Systems (OASIS) standard

# What is DITA?

- Standardized XML-based architecture for authoring, managing, reusing, transforming technical content
- Not an XML schema or set of schemas
- Set of architectural **building blocks** for vocabularies called **element types**
  - **Topic** – a chunk of information
  - **Map** – a structured collection of references to topics, maps, or non-DITA resources
- Architectural rules for creating:
    - New element types that **specialize** existing element types
    - Author-friendly **document type shells** for creating content conforming to an element type

# Specialization

- Specialized element types inherit **processing behavior** of the element type from which they inherit
- DITA specialization architecture is modular
  - Specialization building blocks called **domains** can be reused in multiple element types
  - Encourages interoperability among element types
- Makes DITA unique among XML technologies

# Specialization cost/benefit

Burden falls almost entirely on DITA architects

Document shells and DITA-aware authoring tools hide the complexity from content creators

Inherited functionality means big cost savings for tool developers

- Well worth the added burden on architects

# Source Data Stream Collection Element Type

- Goals satisfied
  - Enable author-friendly composition of SCAP XML resources
  - Leverage DITA capabilities to simplify transformation logic and content management
- Development approach
  - Specializes the DITA base map element type
    - **Because collections and data streams are map-like!**
  - Uses DITA keys to create aliases to XML resource locations
    - **Because SCAP components are key-like!**

# DITA map elements specialized

| DITA map element | Description | Source Data Stream Collection Specialization |
|---|---|---|
| **map** | Root element of a DITA map. | **scapDataStreamCollection** |
| **keydef** | Maps a key name to a text fragment. | **scapComponent** |
| **topicref** | References a topic or resource, or aggregates a collection of **topicref** elements. | **scapDataStream**<br>**scapDictionaries**<br>**scapChecklists**<br>**scapChecks**<br>**scap*component*Ref**<br>**scapExternalLinks**<br>**scapUri** |

# Source data stream collection

```
<scapDataStreamCollection reverseDNS="com.example" scapName="ubuntu1404"
    schematronVersion="1.3">
    <title>Source Data Stream Collection</title>
    <scapComponent keys="oval" href="ssg-ubuntu1404-oval.xml"/>
    <scapComponent keys="cpe-oval" href="ssg-ubuntu1404-cpe-oval.xml"/>
    <scapComponent keys="xccdf" href="ssg-ubuntu1404-xccdf-1.2.xml"/>
    <scapComponent keys="dict" href="ssg-ubuntu1404-cpe-dictionary.xml"/>
    <scapComponent keys="ocil" href="ssg-ubuntu1404-ocil.xml"/>
    <scapComponent href="apparmor-xccdf.xml" keys="xccdf-apparmor"/>
    <scapComponent href="apparmor-oval.xml" keys="oval-apparmor"/>
    <scapDataStream scapName="sds" scapVersion="1.3" useCase="OTHER">
        <scapDictionaries>
            <scapCpeListRef keyref="dict"> [4 lines]
        </scapDictionaries>
        <scapChecklists>
            <scapBenchmarkRef keyref="xccdf"> [5 lines]
            <scapBenchmarkRef keyref="xccdf-apparmor"> [4 lines]
        </scapChecklists>
        <scapChecks>
            <scapOvalRef keyref="cpe-oval"/>
            <scapOvalRef keyref="oval"/>
            <scapOvalRef keyref="oval-apparmor"/>
            <scapOcilRef keyref="ocil"/>
        </scapChecks>
    </scapDataStream>
</scapDataStreamCollection>
```

# SCAP data model representation…

```
<sds:data-stream-collection xmlns:sds="http://scap.nist.gov/schema/scap/source/1.2"
                            xmlns:cat="urn:oasis:names:tc:entity:xmlns:xml:catalog"
                            xmlns:xlink="http://www.w3.org/1999/xlink"
                            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                            id="scap_com.example_collection_ubuntu1404"
                            schematron-version="1.3">
   <sds:data-stream id="scap_com.example_datastream_sds"
                    scap-version="1.3"
                    timestamp="2019-07-10T13:08:33.255-04:00"
                    use-case="OTHER">
      <sds:dictionaries>
         <sds:component-ref id="scap_com.example_cref_dict" [6 lines]
      </sds:dictionaries>
      <sds:checklists>
         <sds:component-ref id="scap_com.example_cref_xccdf" [6 lines]
         <sds:component-ref id="scap_com.example_cref_xccdf-apparmor" [5 lines]
      </sds:checklists>
      <sds:checks>
         <sds:component-ref id="scap_com.example_cref_cpe-oval"
                            xlink:href="#scap_com.example_comp_cpe-oval"/>
         <sds:component-ref id="scap_com.example_cref_oval"
                            xlink:href="#scap_com.example_comp_oval"/>
         <sds:component-ref id="scap_com.example_cref_oval-apparmor"
                            xlink:href="#scap_com.example_comp_oval-apparmor"/>
         <sds:component-ref id="scap_com.example_cref_ocil"
                            xlink:href="#scap_com.example_comp_ocil"/>
      </sds:checks>
   </sds:data-stream>
```

# …continued

```
<sds:component id="scap_com.example_comp_oval"
               timestamp="2019-07-10T13:08:33.255-04:00">
   <ns3:oval_definitions xmlns:ns3="http://oval.mitre.org/XMLSchema/oval-definitions-5" [9397 lines]
</sds:component>
<sds:component id="scap_com.example_comp_cpe-oval"
               timestamp="2019-07-10T13:08:33.255-04:00">
   <ns3:oval_definitions xmlns:ns3="http://oval.mitre.org/XMLSchema/oval-definitions-5" [1480 lines]
</sds:component>
<sds:component id="scap_com.example_comp_xccdf"
               timestamp="2019-07-10T13:08:33.255-04:00">
   <ns10:Benchmark xmlns:ns10="http://checklists.nist.gov/xccdf/1.2" [29104 lines]
</sds:component>
<sds:component id="scap_com.example_comp_dict"
               timestamp="2019-07-10T13:08:33.255-04:00">
   <ns13:cpe-list xmlns:ns13="http://cpe.mitre.org/dictionary/2.0" [53 lines]
</sds:component>
<sds:component id="scap_com.example_comp_ocil"
               timestamp="2019-07-10T13:08:33.255-04:00">
   <ns9:ocil xmlns:ns9="http://scap.nist.gov/schema/ocil/2.0"> [4134 lines]
</sds:component>
<sds:component id="scap_com.example_comp_xccdf-apparmor"
               timestamp="2019-07-10T13:08:33.255-04:00">
   <Benchmark xmlns="http://checklists.nist.gov/xccdf/1.2" [36 lines]
</sds:component>
<sds:component id="scap_com.example_comp_oval-apparmor"
               timestamp="2019-07-10T13:08:33.255-04:00">
   <oval_definitions xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5" [2000 lines]
</sds:component>
</sds:data-stream-collection>
```

# Outline of this talk

- SCAP overview
- Today's SCAP challenges
- SCAP source data stream collections
- SCAP Composer
  - Alternative source data stream collection model
  - Implementation using DITA Open Toolkit

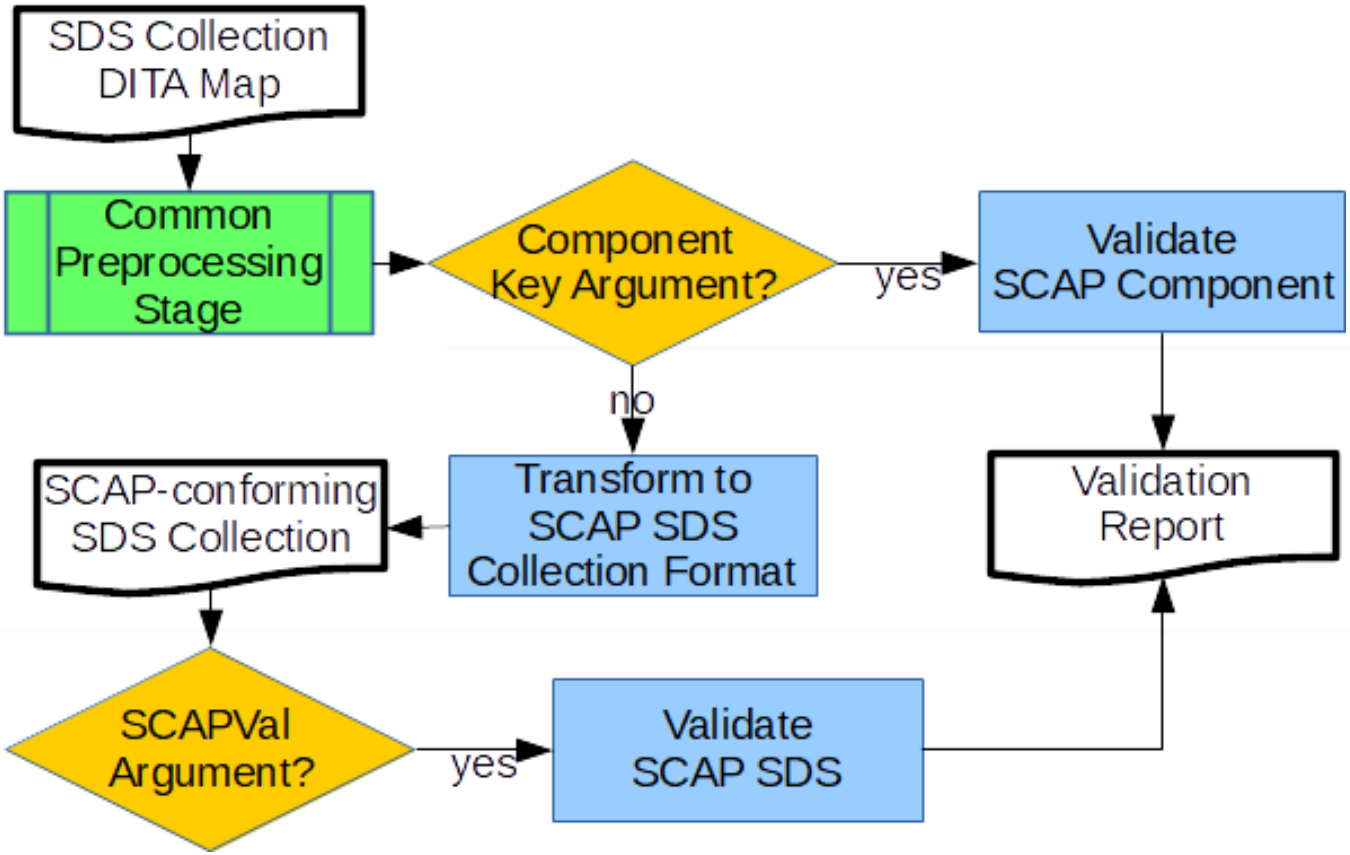
# DITA Open Toolkit (DITA-OT)


- Specialization-aware, output-producing DITA processor
- Modular plug-in architecture
- Open source
  - Reference implementation of the OASIS DITA standard
  - Included in several commercial DITA products
  - Also can be run standalone from command line
- Extensible processing workflow

# SCAP Composer DITA-OT implementation

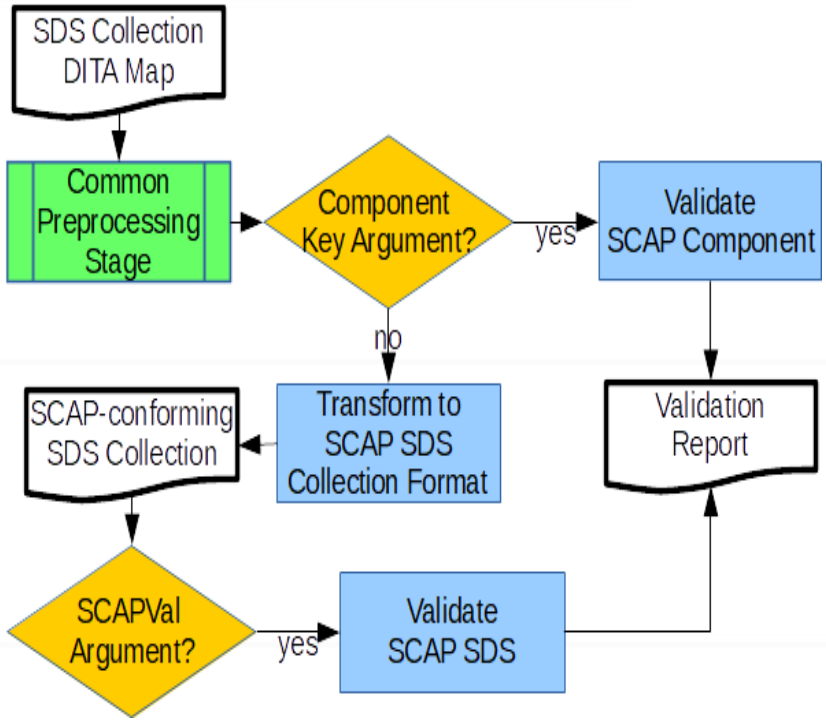
- Document type plug-in
  - Implements the source data stream collection element type
  - Defines a document type shell for authoring
- Transformation plug-in
  - Converts XML authored using the document type plug-in into an SCAP-conforming source data stream collection
  - Uses the **NIST SCAP Content Validation Tool (SCAPval)** to check conformance of individual XML resources and converted result to SCAP requirements
  - Arguments
    - Input document (DITA map valid with respect to the element type)
    - Location of SCAPVal Java Archive (JAR) file
    - Key value of a **scapComponent** element (if validating an XML resource)

# SCAP Composer processing flow



# The power of extensibility

- Makes it easier to implement processing semantics of a DITA specialization
  - "Transform" box
- Also makes it possible to integrate DITA processing with non-DITA processing
  - SCAPVal validation boxes
- Many integration possibilities
  - Jason Fox's splash screen plug-in (https://github.com/jason-fox/fox.jason.splash)
  - "Decomposer" plug-in



# Implementation details

- Document type plug-in
  - XML **schema definitions** for
    - Specialized element and attribute domains
    - Constraints
    - Document type shell  for specialized map type
- Transformation plug-in
  - **XSLT** to supplement basic DITA processing
    - XSLT (Extensible Style Language Transformations): a language for transforming XML data into other formats
  - **Ant** code to supplement default DITA-OT workflow
    - Ant: a language for declaring a sequence of build actions in XML

# Summary

- SCAP represents implementation-specific content for detecting endpoint security issues
- Author-friendly data models can help SCAP content creators meet today's cybersecurity challenges
  - Which include assuring security of operational technology deployed in manufacturing environments
- SCAP Composer uses DITA and the DITA Open Toolkit to:
  - Simplifies source data stream authoring without sacrificing self-containment or reversibility
  - Integrate authoring with SCAP validation and potentially other workflows

# For More Information

- Lubell J (2019) **SCAP Composer: A DITA Open Toolkit Plug-in for Packaging Security Content**. Proceedings of Balisage: The Markup Conference.
- Lubell J (2018) **A New SCAP Information and Data Model for Content Authors**. Critical Infrastructure Protection XII, Springer International Publishing, Vol. 542, pp 127–146. https://www.nist.gov/publications/new-scap-information-model-and-data-model-content-authors
- Security Content Tools GitHub repository: https://github.com/usnistgov/sctools
  - Contains SCAP Composer source code – software release planned later this year
- SCAP
  - Website: https://scap.nist.gov
  - White Paper: Transitioning to the Security Content Automation Protocol (SCAP) Version 2. https://doi.org/10.6028/NIST.CSWP.09102018
  - SCAP Discussion and Development Google Group: https://groups.google.com/a/list.nist.gov/forum/#!forum/scap-dev