Avoiding Inconsistencies in the Security Content Automation Protocol

William M. Fitzgerald
Computer Science Department
University College Cork
Ireland
Email: wfitzgerald@4c.ucc.ie

Simon N. Foley
Computer Science Department
University College Cork
Ireland
Email: s.foley@cs.ucc.ie

***Abstract*—The Security Content Automation Protocol (SCAP) provides a standardized approach to specifying system configuration, vulnerability, patch and compliance management. SCAP comprises a family of existing standards, such as the Open Source Vulnerability Language (OVAL) and the Common Platform Enumeration (CPE). Defining new or extending existing SCAP content is non-trivial and potentially error-prone. For example, specifying a vulnerability in OVAL may appear straightforward, however, the challenge is to specify the vulnerability in such as way that it is consistent with respect to, not just other OVAl data, but also data described under any other standards in SCAP.**

This paper identifies a number of consistency problems that can occur in SCAP specifications and these are illustrated using examples from existing OVAL, CPE, CVE and CCE repositories. It is argued that an ontology-based approach can be used as a means of providing a uniform vocabulary for specifying SCAP data and its relationships. A SCAP ontology is developed based on Semantic Threat Graphs and it is argued that its use can help to ensure consistency across large-scale SCAP repositories.

I. Introduction

The Security Content Automation Protocol (SCAP) [1] is a NIST standard for automating vulnerability management, asset inventory and policy compliance of security configurations. SCAP encompasses a family of standards. For example,

```
<cpe-item name="cpe:/o:redhat:enterprise_linux:3:ga:ws">
```

is a name describing a Redhat enterprise Linux 3 workstation using the CPE [2] standard for naming assets and is structured according to attributes *part, vendor, product, version, update edition* and *language*.

A challenge is the construction of a repository of consistent SCAP definitions. This is non-trivial and potentially error-prone. For example, in addition to the CPE name above, a CPE repository may include the name definition

```
<cpe-item name="cpe:/o:redhat:redhat_ent_linux:3:-:ws">
```

where the only distinguishing difference is that of the *product* and *update* attribute values. The *product* attribute values `enterprise_linux` and `redhat_ent_linux` are implicitly equivalent. Similarly, both `ga` (Linux general realease) and `-` signify that the Redhat operating system is an initial release. While the CPE specification provides a *syntactic* structure for name definitions, it does not explicitly support a *semantics* for name definitions. In the above example, both CPE definitions are semantically equivalent. As a consequence, the CPE repository is composed of some name definitions that are seemingly distinct but are implicitly equivalent.

Thet paper identifies a number of inconsistencies that can arise in SCAP repositories. These inconsistencies arise due to the syntactic nature of the SCAP data and the paper considers how more consistent SCAP repositories may be constructed by using an ontology. An ontology is used to describe a conceptual model of a domain of interest and provides a semantics for the data. We have developed ontologies for SCAP, including CPE, CVE, CCE and OVAL.

This paper is outlined as follows. Section II identifies and discusses some potential inconsistencies in SCAP data. Section III proposes an ontology-based model for SCAP that is an extension of Semantic Threat Graphs [3]. Section IV argues that taking an ontology-based approach helps to avoid inconsistencies in the SCAP data. SectionV considers related research and Section VI concludes the paper.

II. SCAP Challenges

SCAP is intended to provide a standard way to describe configuration artifacts representing assets, vulnerabilities threats and countermeasures. While SCAP represents an evolution of a family of existing standards it is constrained in that it must remain faithful to these standards. This can give rise to challenges in ensuring that SCAP artifacts are fully described and that their use is consistent across the standards.

As part of our study a number of SCAP repositories were inspected, including OVAL, CPE, CVE and CCE, for evidence of inconsistencies in specification. The Open Source Vulnerability Language (OVAL) [4] is a standard for describing asset inventory, vulnerabilities, misconfiguration and patch state, in terms of system characteristics and configuration information. The Common Platform Enumeration (CPE) [5] is a standard for identifing and classifying hardware, operating systems and applications for enterprise asset inventory. The Common Vulnerabilities & Exposures (CVE) standard [6] is used to identify and describe known vulnerabilities. The Common Configuration Enumeration (CCE) standard [7] is used to identify and describe best practice recommendations for security configuration.

```
<definition id="oval:org.mitre.oval:def:7123"
version="3" class="vulnerability">
<title>Cisco 10000, uBR10012, uBR7200 Series
Devices IPC Vulnerability</title>
<affected family="ios">
<platform>Cisco IOS</platform>
<reference source="CVE" ref_id="CVE-2008-3806"
ref_url="http://cve.mitre.org/cgi-bin/
cvename.cgi?name=CVE-2008-3806"/>
<description>Cisco IOS 12.0 through 12.4
on Cisco 10000, uBR10012 and uBR7200 series
devices handles external UDP packets that are
sent to 127.0.0.0/8 addresses intended for
IPC communication within the device, which
allows remote attackers to cause a denial of
service (device or linecard reload) via crafted
UDP packets, a different vulnerability than
CVE-2008-3805.</description>
<criterion comment="IOS vulnerable versions"
test_ref="oval:org.mitre.oval:tst:9269"/>
```

Fig. 1. OVAL fragment for DOS vulnerability in Cisco routers

A. Implicit Rationale

The content in a SCAP repository tends to be operational in nature and does not necessarily provide the full rationale for the definition of a SCAP artifact. This implicit rationale may lead to inconsistencies in the artifact definition. For example, the fragment of the OVAL artifact `oval:org.mitre.oval:def:7123` in Figure 1[1] describes a Denial of Service related vulnerability in a Cisco router. This OVAL definition *explicitly* describes how a particular vulnerability may be executed. However, the rationale for the vulnerability is implicit. The definition does not explicitly define why the vulnerability exists: it is *implicit* that these routers are unable to un-authenticate the origin (local or external) of UDP 127.0.0.0/8 IP packets. Furthermore, it is implicit, in the reference to CVE-2008-3806, that the Denial of Service is exploitable only when spoofed 127.0.0.0/8 IP are sent to the UDP port (1975) .

The extent to which this kind of implicit information is included in a SCAP definition, when it being written, is arbitrary. Definitions that provide explicit rational and those that rely on implicit rationale are equally possible in SCAP. As a consequence, implicit rationale may lead to misinterpretation by those who seek to use the SCAP repository. For example, a SCAP compliant vulnerability scanner may overlook attributes that are implicit in OVAL diagnosis tests.

B. Implicit Intra-Definition Relationships

The artifacts described in a SCAP repository do not necessarily exist independent of one another; in considering one artifact one needs to be aware of how it relates to the other artifacts in the repository. We argue that these relationships should be explicitly described in the SCAP artifact, however, we have found SCAP definitions whereby the relationships are implicit. These implicit relationships can occur within (intra-) the same kind of SCAP artifact or between (inter) different kinds of SCAP artifacts.

[1]Note that for ease of exposition many of the XML tags are removed

```
<cce cce_id='CCE-14264-6' platform='rhel5'
modified='2011-10-07'> <description>The default
policy for iptables INPUT table should be set
as appropriate.</description>
<parameter>ACCEPT / DROP / QUEUE
/RETURN</parameter>
<technical_mechanism>via /etc/sysconfig/iptables
<reference resource_id='NSA "Guide to the
Secure Configuration of Red Hat Enterprise
Linux 5" - Revision 4, September 14,
2010'>Section: 2.5.5.3.1 - Change the Default
Policies</reference>
```

Fig. 2. Excerpt of SCAP CCEv5 `CCE-14264-6` recommendation

For example, the OVAL vulnerability fragment in Figure 1 does not explicitly identify/consider relationships with the relevant threats and countermeasures. In this example, there are two related threats, namely *spoofing* of UDP/IP packets with a source address range of 127.0.0.0/8 with a UDP port of 1975 and *Denial of Service* as a consequence of these spoofed IPC packets. In the definition it (the relationship) is implicit that these threaten the identified Cisco *assets* by exploiting the IPC vulnerability that is a weakness of those Cisco systems. As a consequence, by viewing everything as a single vulnerability, other implicit assets, threats, countermeasures and relationships may be overlooked.

The CCE [7] repository provides a basis for explicit countermeasure recommendations. The consequences of not implementing such recommendations are not always clear. Consider, for example `CCE-14264-6` (Figure 2) which states that one should apply a default firewall policy. However, there is no explicit intuition regarding the kinds of implicit threats or vulnerabilities associated with that recommendation. Notwithstanding compliance requirements [8]–[10], we argue that in order to provide an effective configuration, an administrator needs to properly understand security countermeasure recommendations. Furthermore, the more information that can be made explicit then the more effective is the operation of an automated tool that uses that information.

C. Implicit Inter-Definition Relationships

The different kinds of SCAP artifacts tend to be siloed with respect to each other. For example, rather than explicitly utilize CPE asset identifiers, OVAL, CVE and CCE definitions implicitly encode the semantics of asset dependencies within the text of their titles and definitions. For example, asset references `Cisco 10000`, `uBR10012` and `uBR7200` in Figure 1 implicitly refer to the CPE's outlined in Figure 3. There is no CPE definition for `Cisco 10000`. Implicitly it is not the specific Cisco hardware that is vulnerable but rather specific Cisco IOS versions that the hardware supports. Again, the semantics of the affected Cisco IOS CPE definitions (Figure 4) is implicit in the OVAL definition description. While there is an explicit CVE reference defined within the OVAL definition, it is also implicitly referred to within its description.

```
<cpe-item name="cpe:/h:cisco:ubr10012:-">
<cpe-item name="cpe:/h:cisco:ubr7200">
```

Fig. 3. Excerpt of SCAP CPE Cisco Hardware Definitions

```
<cpe-item name="cpe:/o:cisco:ios:12.4">
```

Fig. 4. Excerpt of SCAP CPE Cisco OS Definitions

Consider the CCE recommendation in Figure 2 concerning the Linux iptables firewall application. Based on the platform defined in `CCE-14264-6`, one knows that the recommendation is one for a RedHat platform. Curiously there is no CPE identifier for RedHat iptables, however, there is one for Suse Linux iptables. As a consequence, is not immediately clear whether the RedHat iptables recommendation is for iptables firewalls in general, or is specifically related to RedHat. One must refer to the implicit relationship with the NSA's best practice document referenced within the `CCE-14264-6` and in other external firewall best practice such as NIST-800-41 [11] to conclude that this recommendation is not RedHat OS specific.

D. Unclear Relationship Reuse

The reuse of relationships across different SCAP definitions may be unclear. For example, the definition of `oval:org.mitre.oval:def:11167` definition in Figure 5 is supported by `CVE-2009-4272`. This defines a denial of service threat as a consequence of crafted packets that force collisions in the IPv4 routing hash table (vulnerability) that is Redhat-centric. However, it is not particularly clear that this same Redhat-centric vulnerability is also a weakness of CentOS and Oracle Linux platforms (assets) as described by the OVAL definition in Figure 5.

It is not necessarily obvious how SCAP data in one definition may be reused in another definition. For example, the vulnerability described by OVAL definition `oval:org.mitre.oval:def:11167` is reused by `oval:org.mitre.oval:def:7026` (Figure 5). However, in this instance the affected platform is the VMWare ESX Server. It is not explicit how the vulnerability described by `oval:org.mitre.oval:def:11167` that affects various Linux platforms also affects VMWare ESX Server described by `oval:org.mitre.oval:def:7026`. The title and description of `oval:org.mitre.oval:def:7026` explicitly refers to RedHat. Through external knowledge one will discover that VMWare ESX Server is built upon a Linux kernel.

Note that with these reuse examples if the vulnerability was explicitly defined (discussed in Section II-B) in the first place then one could conclude that the vulnerability described by both `oval:org.mitre.oval:def:11167` and `oval:org.mitre.oval:def:7026` is Linux vendor neutral. That is, it is a Linux kernel 2.6.31 and earlier IPv4 routing hash table collision vulnerability.

```
<definition id="oval:org.mitre.oval:def:11167"
version="5"
class="vulnerability">
<affected family="unix">
<platform>Red Hat Enterprise Linux 5</platform>
<platform>CentOS Linux 5</platform>
<platform>Oracle Linux 5</platform>
<reference source="CVE" ref_id="CVE-2009-4272"
ref_url="http://cve.mitre.org/cgi-bin/
cvename.cgi?name=CVE-2009-4272"/>
<description>A certain Red Hat patch for
net/ipv4/route.c in the Linux kernel 2.6.18
on Red Hat Enterprise Linux (RHEL) 5 allows
remote attackers to cause a denial of service
(deadlock) via crafted packets that force
collisions in the IPv4 routing hash table, and
trigger a routing "emergency" in which a hash
chain is too long. NOTE: this is related to
an issue in the Linux kernel before 2.6.31,
when the kernel routing cache is disabled,
involving an uninitialized pointer and a
panic.</description>

<definition id="oval:org.mitre.oval:def:7026"
version="3"
class="vulnerability">
<title>Red Hat Linux Kernel Routing
Implementation Multiple Remote Denial of
Service Vulnerabilities</title>
<affected family="unix">
<platform>VMWare ESX Server 4</platform>
<reference source="CVE" ref_id="CVE-2009-4272"
ref_url="http://cve.mitre.org/cgi-bin/
cvename.cgi?name=CVE-2009-4272"/>
<description>A certain Red Hat patch for
net/ipv4/route.c in the Linux kernel 2.6.18
on Red Hat Enterprise Linux (RHEL) 5 allows
remote attackers to cause a denial of service
(deadlock) via crafted packets that force
collisions in the IPv4 routing hash table, and
trigger a routing "emergency" in which a hash
chain is too long. NOTE: this is related to
an issue in the Linux kernel before 2.6.31,
when the kernel routing cache is disabled,
involving an uninitialized pointer and a
panic.</description>
```

Fig. 5. Excerpt of SCAP OVAL Vulnerability Definition Reuse

E. Unclear Definition Hierarchy

SCAP definitions are typically developed in isolation, that is, they focus on a single asset, threat, vulnerability or countermeasure. As a consequence, it can become difficult to inter-relate implicit information across multiple SCAP definitions. Consider OVAL vulnerability definitions `oval:org.mitre.oval:def:7123` (Figure 1) and `oval:org.mitre.oval:def:5910` (Figure 6), which are effectively equivalent. The only distinguishing difference is that `oval:org.mitre.oval:def:5910` prescribes an additional SIP test in the test tag reference (`test_ref="oval:org.mitre.oval:tst:23990"/>`). This test is carried out in order to verify whether an asset is vulnerable to un-authenticated UDP 127.0.0.0/8 IP packets. The OVAL definitions state that: *"a different vulnerability*

```
<definition id="oval:org.mitre.oval:def:5910"
version="3" class="vulnerability">
<title>Cisco 10000, uBR10012, uBR7200 Series
Devices IPC Vulnerability</title>
<affected family="ios">
<platform>Cisco IOS</platform>       <reference
source="CVE" ref_id="CVE-2008-3805"
ref_url="http://cve.mitre.org/cgi-bin/
cvename.cgi?name=CVE-2008-3805"/>
<description>Cisco IOS 12.0 through 12.4
on Cisco 10000, uBR10012 and uBR7200 series
devices handles external UDP packets that are
sent to 127.0.0.0/8 addresses intended for
IPC communication within the device, which
allows remote attackers to cause a denial of
service (device or linecard reload) via crafted
UDP packets, a different vulnerability than
CVE-2008-3806.</description>
<criterion comment="IOS vulnerable versions"
test_ref="oval:org.mitre.oval:tst:9269"/>
<criterion comment="config contains SIP"
test_ref="oval:org.mitre.oval:tst:23990"/>
```

Fig. 6. Excerpt of SCAP OVAL `oval:org.mitre.oval:def:5910`

```
<cpe-item name="cpe:/h:cisco:ubr7200">
<cpe-item name="cpe:/h:cisco:Cisco_ubr7200">
<cpe-item name="cpe:/h:cisco:Cable_Router_ubr7200">
```

Fig. 7. SCAP CPE Definition Name Ambiguity

than CVE-2008-3805" and *"a different vulnerability than CVE-2008-3806"*, respectively, and, therefore one concludes there is no accidental duplication of OVAL definitions nor reuse of vulnerability information. However, one knows that these two OVAL definitions should be related but it is unclear whether one subsumes the other or whether they are disjoint siblings. If the latter, then their relationship is unclear. A similar issue arises when comparing CVE definitions `CVE-2008-3805` and `CVE-2008-3806`.

F. Definition Name Ambiguity

Ambiguities can arise in the definition of SCAP names/identifiers. In the CPE repository there is a CPE definition for a Cisco `ubr7200` cable router (the 1st cpe-item XML tag in Figure 7). However, there is nothing to prevent additional CPE names being added at a later date (for example, the 2nd and 3rd cpe-item XML tags in Figure 7). The CPE naming specification does not allow for defining explicit inter-relationships between (equivalent) CPE definitions. As a consequence, it is unclear if these CPE definitions represent the same Cisco ubr7200 cable router asset or a set of distinct Cisco ubr7200 based cable router assets.

G. The Potential for SCAP Inconsistencies

The extent to which implicit information is included in a SCAP definition, when it being written, is arbitrary. The above examples demonstrate the potential for inconsistencies in SCAP definitions as a consequence of a reliance on implicit information. In principle, if SCAP definitions/artifacts were sufficiently explicit then such inconsistencies should not emerge. In the next section a formal model is developed for

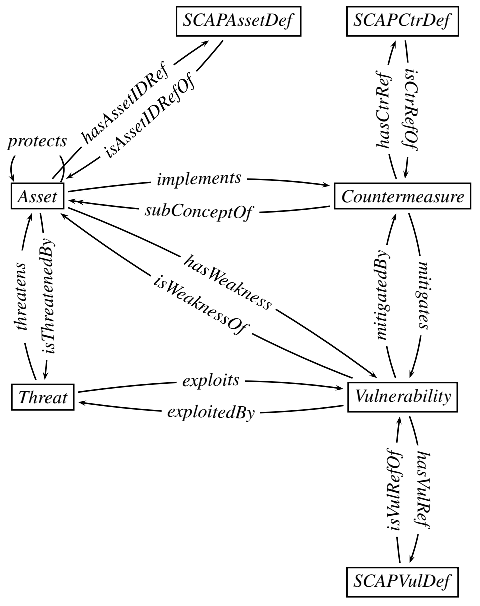

Fig. 8. Abstract Semantic Threat Graph Model.

SCAP which seeks to avoid these inconsistencies by requiring explicit declaration of information used in SCAP definitions.

III. A Semantic Threat Graphs Model for SCAP

Rather than developing a SCAP ontology from scratch, we found it more effective to describe SCAP concepts as an extension of a *Semantic Threat Graph (STG)* ontology [3]. This is not unreasonable given that both SCAP and STGs are concerned with describing assets, vulnerabilities, threats and countermeasures and their relationships. Semantic Threat Graphs are defined using Description Logic (*DL*), a decidable portion of first-order logic [12]. Concepts represent sets of individuals and properties represent binary relations applied to individuals. Note that in presenting the model components, for reasons of space, we do not provide complete specifications in particular, definitions do not include disjoint axioms, sub-properties, data type properties or closure axioms.

Figure 8 provides an abstract model of a semantic threat graph that illustrates the SCAP concepts involved and their relationships. Figure 9 provides an example instantiation of this model for `oval:org.mitre.oval:def:7123`, the OVAL definition illustrated in Figure 1. Note, the dashed property lines and the hollow individual nodes are not explicitly part of the encoding of this OVAL definition

A. Asset

Concept $Asset$ represents any entity of interest within the enterprise that may be the subject of a threat. While assets can include people and physical infrastructure, this paper only considers computer-system based entities such as Cisco routers, firewalls and so forth.

Assets may have zero or more vulnerabilities ($\forall$ restriction) along property $hasWeakness$. As a result, those assets may be exposed to various individuals of the $Threat$

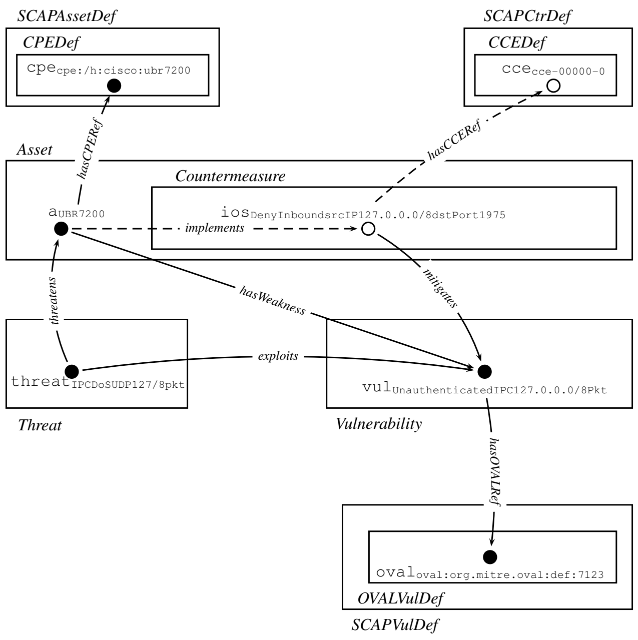

Fig. 9. Example Fragment of a Semantic Threat Graph Instance

concept. An asset may have the capability to implement a $Countermeasure$ that $protects$ itself or other assets. Assets have zero or more SCAP compliant asset identification definitions (individuals of concept *SCAPAssetDef*) along property $hasAssetIDRef$.

$$\begin{aligned}Asset \sqsubseteq\ & \forall_{\geq 0} hasWeakness.Vulnerability \sqcap \\ & \forall_{\geq 0} isThreatenedBy.Threat \sqcap \\ & \forall_{\geq 1} protects.Asset \sqcap \\ & \forall_{\geq 0} implements.Countermeasure \sqcap \\ & \forall_{\geq 0} hasAssetIDRef.\mathit{SCAPAssetDef}\end{aligned}$$

Concept $Asset$ is further specialised to have more specific kinds of asset concepts, for example, $Application, OperatingSystem, Hardware \sqsubseteq Asset$ representing the set of $Application$, $OperatingSystem$ and $Hardware$ individuals that an enterprise may have. The $Asset$ concept property relation $hasAssetIDRef$ is also further specialised to more specific kinds of range relationships. For example, the $hasCPERef \sqsubseteq hasAssetIDRef$ defines a sub-property relation used to relate individuals of the $Asset$ concept to specific range concept, namely $CPEDef$ (CPE definitions). The $hasCPERef$ property relation asserts the relationship between assets and their associate CPE definition names. Note, this sub-property has the following restriction: $Asset \sqsubseteq \exists_{=1} hasCPERef.CPEDef$ stating that an asset must have at most one reference to a CPE definition.

Consider the `oval:org.mitre.oval:def:7123` definition outlined in Figure 1 as a running example. A Cisco cable router individual $\mathtt{a}_{\mathtt{UBR7200}}$, an instance of the $Hardware$ concept (inferred as an individual of concept $Asset$) is vulnerable to receiving unauthenticated 127.0.0.0/8 UDP packets (individual $\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}$). As a consequence, $\mathtt{a}_{\mathtt{UBR7200}}$ $isThreatenedBy$ an IPC Denial of Service attack (individual $\mathtt{threat}_{\mathtt{IPCDoSUDP127/8pkt}}$). The following fragment of the ontology asserts these facts.

$$\begin{aligned}& Asset(\mathtt{a}_{\mathtt{UBR7200}}) \\ \leftarrow\ & hasWeakeness(\mathtt{a}_{\mathtt{UBR7200}}, \mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}) \sqcap \\ & \quad isThreatenedBy(\mathtt{a}_{\mathtt{UBR7200}}, \mathtt{threat}_{\mathtt{IPCDoSUDP127/8pkt}})\end{aligned}$$

Note, in order to avoid having to explain the low-level details of each ontology individual, individuals are given human readable names as a syntactic sugar that contain a semantics of what each individual represents.

Asset individual $\mathtt{a}_{\mathtt{UBR7200}}$ is asserted to have a relationship with the following $\mathtt{cpe}_{\mathtt{cpe:/h:cisco:ubr7200}}$ individual along property $hasCPERef$. Note, for further reference, an ontology model for CPE definitions is presented in Appendix A.

$$\begin{aligned}& Asset(\mathtt{a}_{\mathtt{UBR7200}}) \\ \leftarrow\ & hasCPERef(\mathtt{a}_{\mathtt{UBR7200}}, \mathtt{cpe}_{\mathtt{cpe:/h:cisco:ubr7200}})\end{aligned}$$

Cisco's UBR2700 router (individual $\mathtt{a}_{\mathtt{UBR7200}}$) has a role that *protects* internal assets inclusive of itself. The following fragment of the ontology asserts that the $\mathtt{a}_{\mathtt{UBR7200}}$ asset *protects* itself from the $\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}$ by implementing an IOS firewall countermeasure that denies inbound external IP packets with source addresses within the 127.0.0.0/8 range destined to port 1975 ($\mathtt{ios}_{\mathtt{DenyInSrcIP127/8DstPort1975}}$). This recommendation is considered a Cisco best practice [13].

$$\begin{aligned}& Asset(\mathtt{a}_{\mathtt{UBR7200}}) \\ \leftarrow\ & protects(\mathtt{a}_{\mathtt{UBR7200}}, \mathtt{a}_{\mathtt{UBR7200}}) \sqcap \\ & \quad implements(\mathtt{a}_{\mathtt{UBR7200}}, \mathtt{ios}_{\mathtt{DenyInSrcIP127/8DstPort1975}})\end{aligned}$$

B. Threat

A threat is a potential for violation of security [14]. An individual of the $Threat$ concept is considered to exploit one or more vulnerabilities ($\exists_{\geq 1}$ restriction).

$$\begin{aligned}Threat \sqsubseteq\ & \exists_{\geq 1} exploits.Vulnerability \sqcap \\ & \exists_{\geq 1} threatens.Asset\end{aligned}$$

Continuing to use the `oval:org.mitre.oval:def:7123` definition as a running example, an individual of concept $Threat$, $\mathtt{threat}_{\mathtt{IPCDoSUDP127/8pkt}}$, $threatens$ the Cisco cable router $\mathtt{a}_{\mathtt{UBR7200}}$ asset.

$$\begin{aligned}& Threat(\mathtt{threat}_{\mathtt{IPCDoSUDP127/8pkt}}) \\ \leftarrow\ & exploits(\mathtt{threat}_{\mathtt{IPCDoSUDP127/8pkt}}, \\ & \qquad\qquad \mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}) \sqcap \\ & \quad threatens(\mathtt{threat}_{\mathtt{IPCDoSUDP127/8pkt}}, \mathtt{a}_{\mathtt{UBR7200}})\end{aligned}$$

C. Vulnerability

A vulnerability is a flaw or security weakness in an asset that has the potential to be exploited by a threat. Concept $Vulnerability$ is the set of concrete vulnerabilities. Each vulnerability individual may have zero or more SCAP vulnerability definition (*SCAPVulDef* concept) reference relationships ($hasVulRef$).

$$\begin{aligned} Vulnerability \sqsubseteq\ & \exists_{\geq 1} isExploitedBy.Threat \sqcap \\ & \exists_{\geq 1} \mathit{isWeaknessOf}.Asset \sqcap \\ & \forall_{\geq 0} hasVulRef.SCAPVulDef \end{aligned}$$

Note, property $hasVulRef$ is further specialised as $hasCVERef$ and $hasOVALVulRef$. In addition, concept $SCAPVulDef$ is further specialised as follows: $CVEDef, OVALVulDef \sqsubseteq SCAPVulDef$ and is representative of CVE and OVAL vulnerability definitions respectively.

The following fragment in the ontology states that asset $\mathtt{a}_{\mathtt{UBR7200}}$ is susceptible to a $\mathtt{threat}_{\mathtt{IPCDoSUDP127/8pkt}}$ attack via the $\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}$ weakness. Note, $\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}$ is representative of a weakness in the TCP stack where it is not possible to authenticate the origin of localhost IP packets. This explicit vulnerability has a reference to the $\mathtt{oval}_{\mathtt{oval:org.mitre.oval:def:7123}}$ definition individual along the *hasOVALVulRef* property relation.

$$\begin{aligned} & Vulnerability(\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}) \\ \leftarrow\ & isExploitedBy(\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}, \\ & \qquad \mathtt{threat}_{\mathtt{IPCDoSUDP127/8pkt}}) \sqcap \\ & isWeaknessOf(\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}, \mathtt{a}_{\mathtt{UBR7200}}) \sqcap \\ & hasOVALVulRef(\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}, \\ & \qquad \mathtt{oval}_{\mathtt{oval:org.mitre.oval:def:7123}}) \end{aligned}$$

D. Countermeasure

A countermeasure is an action or process that mitigates vulnerabilities and prevents and/or reduces threats. A countermeasure is an asset and may have zero or more SCAP countermeasure references along property *hasCtrRef*.

$$\begin{aligned} Countermeasure \sqsubseteq\ & Asset \sqcap \\ & \forall_{\geq 0} \mathit{hasCtrRef.SCAPCtrDef} \end{aligned}$$

Note, property $hasCtrRef$ is further specialised as $hasCCERef$ and $hasOVALComplianceRef$.

Concept $IOSRule$ is representative of the Cisco IOS firewall rules that mitigate one or more vulnerabilities, provided they are implemented by $Firewall$ individuals where $Firewall \sqsubseteq Asset$.

$$\begin{aligned} IOSRule \sqsubseteq\ & Countermeasure \sqcap \\ & \exists_{\geq 1} mitigates.Vulnerability \sqcap \\ & \forall_{\geq 0} implementedBy.Firewall \end{aligned}$$

The following fragment in the ontology states that the Cisco IOS firewall rule individual $\mathtt{ios}_{\mathtt{DenyInSrcIP127/8DstPort1975}}$, mitigates the vulnerability $\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}$ on the Cisco cable router ($\mathtt{a}_{\mathtt{UBR7200}}$).

$$\begin{aligned} & IOSRule(\mathtt{ios}_{\mathtt{DenyInSrcIP127/8DstPort1975}}) \\ & mitigates(\mathtt{ios}_{\mathtt{DenyInSrcIP127/8DstPort1975}}, \\ & \qquad \mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}) \sqcap \\ & implementedBy(\mathtt{ios}_{\mathtt{DenyInSrcIP127/8DstPort1975}}, \\ & \qquad \mathtt{a}_{\mathtt{UBR7200}}) \end{aligned}$$

For the sake of clarity, a number of property relationships have been excluded from this discussion. Assets, threats, vulnerabilities and countermeasures have additional properties.

IV. SCAP Challenges Revisited

This section revisits the SCAP challenges outlined in Section II and discusses how the inconsistencies may be avoided.

A. Explicit Rationale

The semantic threat graph framework makes the implicit rationale outlined in Section II-A explicit by asserting *why* it is a vulnerability exists. For example, Cisco routers described in the OVAL `oval:org.mitre.oval:def:7123` definition are unable to authenticate the origin (local or external) of UDP 127.0.0.0/8 IP packets. This vulnerability is asserted as individual $\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}$. An OVAL vulnerability definition reference relationship ($hasOVALVulRef$) between this vulnerability and the OVAL $\mathtt{oval}_{\mathtt{oval:org.mitre.oval:def:7123}}$ definition individual is also asserted (Figure 9).

B. Explicit Intra-Definition Relationships

The implicit entities and relationships that can occur within the same SCAP definition are made explicit. For example, it becomes possible to explicitly consider the relationships with the relevant asset and corresponding threat for the $\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}$ vulnerability (Figure 9). That is, vulnerability $\mathtt{vul}_{\mathtt{Unauth127.0.0.0/8Pkt}}$ *isAWeaknessOf* asset $\mathtt{a}_{\mathtt{UBR7200}}$ and as a consequence it $isThreatendedBy$ threat $\mathtt{threat}_{\mathtt{IPCDoSUDP127/8pkt}}$.

C. Explicit Inter-Definition Relationships

Using the semantic threat graphs framework, the semantics of the affected Cisco hardware CPE definitions implicit in Figure 1 are made explicit. For example, the Cisco cable router asset individual $\mathtt{a}_{\mathtt{UBR7200}}$ is asserted to have a *hasCPERef* property relationship with CPE definition individual $\mathtt{cpe}_{\mathtt{cpe:/h:cisco:ubr7200}}$.

D. Explicit Definition Hierarchies

In simple terms, the SCAP repository is effectively a forest of inter-related SCAP definitions that does not have an explicit hierarchical structure. Under the ontology Open World Assumption [15], the semantic threat graphs framework is extensible where further sub-concepts and sub-properties can be defined. For example, concept $CPEDef$ is a sub-concept of the set of all SCAP based asset identifications (concept $SCAPAssetDef$). This concept can be further specialised:$CPEHW, CPEOS, CPEApp \sqsubseteq CPEDef$ where these sub-concepts are representative of CPE hardware, operating system and application definition names respectively. Concept $CPEHW$ in turn may be further specialised: $CPEHWCisco, CPEHWRedhat \sqsubseteq CPEHW$. Figure 10 illustrates a fragment of the CPE concept hierarchy. If individual $\mathtt{cpe}_{\mathtt{cpe:/h:cisco:ubr7200}}$ is asserted as a member of concept $CPEHWCisco$ then it will be inferred as a member of all its parent concepts.

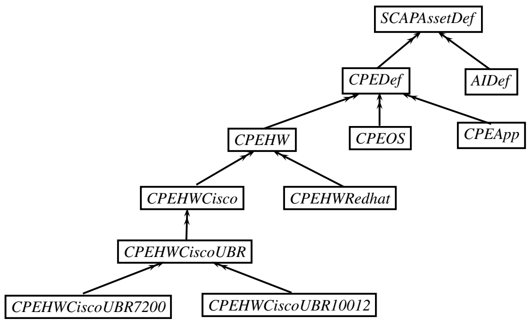

Fig. 10. Fragment of SCAP Definition Hierarchy

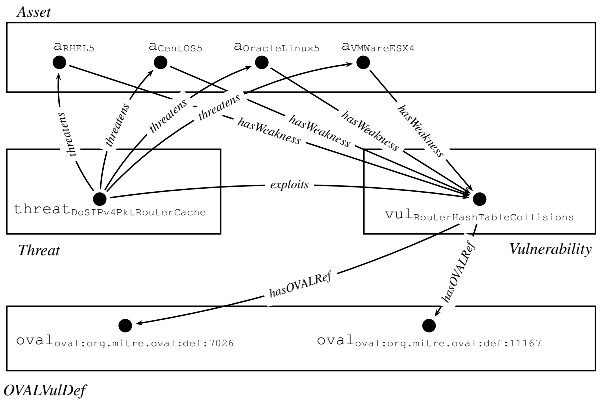

Fig. 11. Example Fragment of SCAP Definition Reuse

E. Explicit Relationship Reuse

The semantic threat graphs framework provides a basis for the explicit reuse of SCAP data within and across SCAP definitions. For example, described within the following OVAL `oval:org.mitre.oval:def:11167` definition is a threat $\mathtt{threat}_{\mathtt{DoSIPv4PktRouterCache}}$ and vulnerability $\mathtt{vul}_{\mathtt{RouterHashTableCollisions}}$ that are explicitly shared amongst a number of assets namely $\mathtt{a}_{\mathtt{RHEL5}}$, $\mathtt{a}_{\mathtt{CentOS5}}$ and $\mathtt{a}_{\mathtt{OracleLinux5}}$. OVAL data defined within the OVAL `oval:org.mitre.oval:def:11167` definition is explicitly reused in the OVAL `oval:org.mitre.oval:def:7026` definition. Asset $\mathtt{a}_{\mathtt{VMWareESX4}}$ is affected by the same threat and vulnerability. Figure 11 illustrates a fragment of SCAP definition reuse.

F. Non-Ambiguous Definition Names

Unique Name Assumption is not supported in Description Logic. As a result, unless explicitly stated otherwise, it cannot be assumed that individuals identified with different names are distinct. Consider the example in Figure 7. A Cisco UBR7200 cable router asset (individual $\mathtt{a}_{\mathtt{UBR7200}}$), amongst other concept restrictions, may only have a relationship ($\exists_{=1} hasCPERef.CPEDef$) to one CPE definition. If individuals of concept $CPEDef$ namely $\mathtt{cpe}_{\mathtt{cpe:/h:cisco:ubr7200}}$, $\mathtt{cpe}_{\mathtt{cpe:/h:cisco:ciscoubr7200}}$ and $\mathtt{cpe}_{\mathtt{cpe:/h:cisco:ciscorouterubr7200}}$ are not explicitly defined as distinct from each other then these CPE definition individuals will be inferred to represent the same individual. However, if these individuals have been asserted as distinct, then an inconsistency will be detected when reasoning over the model.

V. Related Research

By interpreting SCAP in terms of Semantic Threat Graphs, it has been possible to demonstrate that encoding SCAP data within an ontology can help to avoid the kinds of inconsistencies discussed in Section II. Semantic threat graphs have been shown to be effective for modeling and reasoning about high-level best practice security recommendations, such as those from NIST standards, in terms of low-level assets, threats, vulnerabilities and countermeasures [3]. The observations about SCAP in this paper are not confined to Semantic Threat Graphs; we conjecture that similar observations could be arrived at by developing an SCAP ontology from scratch or by using other ontology-based threat models [16], [17].

A general purpose asset management ontology is presented in [18]. In [19], an ontology for CCE is presented and an ontology for CVE is presented in [20]. Mathews et. al. [17] use Notation-3 encodings built from an ontology of National Vulnerability Database data to provide context information for improving intrusion detection.

In [21], [22], ontologies have been shown to be effective at modelling and reasoning about low-level systems security configurations (for example [21], [22]). Thus, we argue an ontology engineering approach to encode existing low-level SCAP repository information is practical.

VI. Discussion and Conclusion

This paper identified a number of consistency challenges that can occur in SCAP repositories. For example, implicit definition relationships. A semantic threat graphs model for SCAP was used as a systematic approach to make the implicit explicit within and across inter-related SCAP definitions, thereby providing a basis to avoid such inconsistencies.

Future research will consider a number of additional inconsistency classifications. For example, *Definition Error Detection* and *Implicit Inter-Definition Vulnerability Cascade*. The OVAL `oval:org.mitre.oval:def:14812` definition (Figure 12) is an example where there is an integer overflow in Adobe Reader on Linux platforms and (in contradiction) this vulnerability affects Microsoft Windows platforms. Security configurations (inclusive of patches) used to mitigate vulnerabilities may in turn themselves introduce vulnerabilities. In SCAP these cyclic dependencies between seemingly disparate vulnerabilities are implicit, giving rise to a further *Implicit Inter-Definition Vulnerability Cascade* inconsistency classification. The OVAL `oval:org.mitre.oval:def:11167` definition outlined in Figure 5 is an example where a Redhat patch gave rise to this particular vulnerability.

Acknowledgement

This research has been supported by Science Foundation Ireland grant 08/SRC/11403. The authors would like to thank Kathleen Moriarty for useful discussions on SCAP.

```
<definition id="oval:org.mitre.oval:def:14812"
version="6" class="vulnerability">
<title>Integer overflow in Adobe Reader 9.x before 9.4.6
on Linux allows attackers to execute arbitrary code via
unspecified vectors.</title>
<affected family="windows">
```

Fig. 12. Fragment of `oval:org.mitre.oval:def:14812` Definition

REFERENCES

[1] http://scap.nist.gov/.
[2] B. A. Cheikes, D. Waltermire, and K. Scarfone, "Common Platform Enumeration: Naming Specification Version 2.3," *NIST Interagency Report 7695, NIST-IR-7695*, August 2011.
[3] S. N. Foley and W. M. Fitzgerald, "Management of Security Policy Configuration using a Semantic Threat Graph Approach," *Journal of Computer Security (JCS), IOS Press, Volume 19, Number 3*, 2011.
[4] https://oval.mitre.org/.
[5] http://cpe.mitre.org/.
[6] http://cve.mitre.org/.
[7] http://cce.mitre.org/.
[8] P. S. S. Council, "Payment Application Data Security Standard: Requirements and Security Assessment Procedures Version 1.2.1," *Payment Card Industry (PCI) Data Security Standard*, July 2009.
[9] BSI, "IT Grundschutz Manual," *https://www.bsi.bund.de*, May 2010.
[10] ISO, "ISO 27001 Standard," *http://www.27000.org*, May 2010.
[11] J. Wack, K. Cutler, and J. Pole, "Guidelines on Firewalls and Firewall Policy," *Recommendations of the National Institute of Standards and Technology, NIST-800-41*, 2002.
[12] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, March 2003.
[13] http://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20080924-ipc.
[14] R. Shirey, "RFC 2828: Internet Security Glossary," *http://ietf.org*, May 2000.
[15] M. K. Smith, C. Welty, and D. L. McGuinness, "OWL Web Ontology Language Guide," *W3C Recommendation, Technical Report*, 2004.
[16] J. Undercoffer, A. Joshi, and J. Pinkston, "Modeling computer attacks: An ontology for intrusion detection," in *Recent Advances in Intrusion Detection, 6th International Symposium, RAID*, 2003.
[17] M. L. Mathews, P. Halvorsen, A. Joshi, and T. Finin, "A collaborative approach to situational awareness for cybersecurity," in *th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom*, 2012, pp. 216–222.
[18] H. Birkholz, I. Sieverdingbeck, K. Sohr, and C. Bormann, "IO: An Interconnected Asset Ontology in Support of Risk Management Processes," *7th International Conference on Availability, Reliability and Security (ARES), Washington, DC, USA*, August 2010.
[19] M. C. Parmelee, "Toward an Ontology Architecture for Cyber-Security Standards," *5th International Conference on Semantic Technologies for Intelligence, Defense, and Security, (STIDS), George Mason University, USA*, October 2010.
[20] J. A. Wang and M. Guo, "OVM: An Ontology for Vulnerability Management," *5th Cyber Security and Information Intelligence Research Workshop (CSIIRW), Oak Ridge, Tennessee, USA*, April 2009.
[21] W. M. Fitzgerald and S. N. Foley, "Management of Heterogeneous Security Access Control Configuration using an Ontology Engineering Approach," *3rd ACM Workshop on Assurable and Usable Security Configuration, (SafeConfig), Chicago, USA*, October 2010.
[22] ——, "Reasoning about the Security Configuration of SAN Switch Fabrics," *4th Symposium on Configuration Analytics and Automation, (SafeConfig), Arlington, VA, USA*, November 2011.
[23] A. Phillips and M. Davis, "RFC 4646: Tags for Identifying Languages," *http://ietf.org*, September 2006.

APPENDIX A
ONTOLOGY MODEL FOR THE CPE STANDARD

This Appendix presents an ontology model for CPE, in order to serve as an illustration. Ontologies for CVE, CCE and OVAL are omitted due to page constraints.

CPE is a standard specification used to provide a CPE name definition for a given IT product ($Asset$). The CPE repository is a collection of CPE definitions (concept $CPEDef$) that have a CPE Well Formed Name ($CPEWFN$), zero or more human-readable descriptive $Note$s and zero or more OVAL diagnostic $Check$s.

$$\begin{array}{rcl} CPEDef & \sqsubseteq & \exists_{=1} hasTitle.String \sqcap \\ & & \exists_{=1} hasCPEName.CPEWFN \sqcap \\ & & \forall_{\geq 0} hasNote.Note \sqcap \\ & & \forall_{\geq 0} hasCheck.Check \end{array}$$

The following ontology fragment:

$$\begin{array}{l} CPEDef(\texttt{cpeDef}_1) \\ \leftarrow \quad hasTitle(\texttt{cpeDef}_1, \texttt{'Cisco Universal Broadband Router (uBR7200)'}) \sqcap \\ \qquad hasCPEName(\texttt{cpeDef}_1, \texttt{cpe}_1) \end{array}$$

is a unique CPE name definition for Cisco uBR7200 universal broadband routers. The following defines a model for the CPE name definition scheme:

$$\begin{array}{rcl} CPEWFN & \sqsubseteq & \exists_{=1} hasName.String \sqcap \\ & & \exists_{=1} hasPart.Part \sqcap \\ & & \exists_{=1} hasVendor.Vendor \sqcap \\ & & \exists_{=1} hasProduct.Product \sqcap \\ & & \exists_{=1} hasVersion.Version \sqcap \\ & & \exists_{=1} hasUpdate.Update \sqcap \\ & & \exists_{=1} hasEdition.Edition \sqcap \\ & & \exists_{=1} hasLanguage.RFC4646Tag \end{array}$$

Concepts $Part$ and $RFC4646Tag$ are defined as enumerated sets of individuals. Individuals of concept $Part$ represent hardware (h), operating systems (o) and applications (a) respectively. Concept $RED4646Tag$ represents the set of RFC defined tags for identifying languages [23].

$$\begin{array}{rcl} Part & \equiv & \{\texttt{h}, \texttt{o}, \texttt{a}\} \\ RFC4646Tag & \equiv & \{\texttt{en-US}, \texttt{zh-tw}, \texttt{ja-JP}, \ldots\} \end{array}$$

The following fragment of the ontology asserts individual $\texttt{cpe}_1$ has name definition `'cpe:/h:cisco:ubr7200'` and is composed of the following hardware `h`, vendor `cisco` and product `ubr7200` individuals.

$$\begin{array}{l} CPEWFN(\texttt{cpe}_1) \\ \leftarrow \quad hasName(\texttt{cpe}_1, \texttt{'cpe:/h:cisco:ubr7200'}) \sqcap \\ \qquad hasPart(\texttt{cpe}_1, \texttt{h}) \sqcap \\ \qquad hasVendor(\texttt{cpe}_1, \texttt{cisco}) \sqcap \\ \qquad hasProduct(\texttt{cpe}_1, \texttt{ubr7200}) \end{array}$$