Security Content Metadata Model with an Efficient Search Methodology for Real Time Monitoring and Threat Intelligence

Preeti Subramanian

Abstract

The Security Content Automation Protocol (SCAP) federates a number of open standards that are used to enumerate software flaws and configuration issues related to security. They measure systems to find vulnerabilities and offer methods to score those findings in order to evaluate the possible impact. There are a number of SCAP components such as Common Vulnerabilities and Exposures (CVE), Common Configuration Enumeration (CCE), Common Platform Enumeration (CPE), Common Remediation Enumeration (CRE), Extensible Configuration Checklist Description Format (XCCDF), and Open Vulnerability and Assessment Language (OVAL). Malware Attribute Enumeration and Characterization (MAEC) is a standardized language for encoding and communicating high-fidelity information about malware based upon attributes such as behaviours, artefacts, and attack patterns. These standards render data in the form of XML.

Although these standards are linked to each other, there is a lack of commonality in their XML schema definitions. There is a need for a unique common metadata schema to represent important aspects relevant for designing efficient search mechanism. This common metadata supports distribution of data across various repositories that render SCAP content. Across all security content databases unique identification and a short description will be common. In addition, this model makes building of relations to multiple components of SCAP intuitive. Differentiating attributes of security content can be represented as a list of properties, each property being a key-value pair. For example, in the case of CVE, (CVSS, 9.4) represents the key CVSS and a score of 9.4, where CVSS is Common Vulnerability Severity Score. In this model, modifications to the schema of SCAP components can easily be accommodated by just adding or deleting a property key-value pair without changing the model. Searching on this metadata enables fast response to queries and helps interlace various SCAP components; e.g., OVAL references CVE and each CVE depends on various platforms and products denoted by CPEs. This model enables Natural Language Processing (NLP) and renders meaningful responses to queries such as most vulnerable applications, OVAL definitions, vulnerabilities in Adobe Reader in 2014, recent threats etc. 90% of malware attacks make use of an existing vulnerability in the system. This archetype aids to resolve vulnerabilities before an attack happens. In a case where system events are continuously monitored, this model also helps understand an incident in a machine and analyse to determine if it is a malware attack. It will additionally help to scrutinize which vulnerability was exploited by the malware and most importantly, fix the vulnerability to prevent further attacks.

Background

As an organizational security protection mechanism, we need to implement different silos of security technologies, vulnerability management, patch management, incident response or security information and event management (SIEM), malware management, intrusion detection systems. Each of these operates with different sets of data, often only understood by them. However, the underlying common goal is to protect an organization or an entity from adversaries.

This poses two questions:

- **Firstly, why is there a lack of commonality or inter-relationship between these data sets?**
- **Secondly, why don't these products talk to each other and devise a response mechanism which works like a single system?**

These questions lay the foundation of Security Content Automation Protocol, commonly known as SCAP and few other additional standards being developed part of the National Institute of Standards and Technology (NIST) and other standards bodies.

Understanding SCAP

Security Content Automation Protocol, SCAP is a suite of specifications that standardize the format and nomenclature by which software flaw and security configuration information is communicated, both to machines and humans. SCAP is a multi-purpose framework of specifications that support automated configuration, vulnerability and patch checking, technical control compliance activities, and security measurement. Goals for the development of SCAP include standardizing system security management, promoting interoperability of security products, and fostering the use of standard expressions of security content.

SCAP is categorized into Languages, Metrics, Enumeration, Reporting formats and Integrity.

Languages

SCAP provides conventions to express vulnerability assessment, security policies and technical check mechanism in the form of OVAL, XCCDF and OCIL.

Enumeration

SCAP defines a naming format and a list of items that are defined in that nomenclature. It consists of CPE, CCE, CVE and CWE.

Metrics

Measurement and scoring systems in SCAP refers to evaluation of each security weakness and assign a score to each of these weaknesses. Common Vulnerability Scoring System (CVSS) and Common Configuration Scoring System (CCSS) are the scoring system.

Reporting Formats

SCAP describes reporting format that provides necessary constructs to express collected information in standardized formats. The SCAP reporting format specifications are Asset Reporting Format (ARF) and Asset Identification.

Integrity

An SCAP integrity specification helps to preserve the integrity of SCAP content and results. Trust Model for Security Automation Data (TMSAD) is the SCAP integrity specification.

SCAP touches all the requirements for automating information exchange between two entities. It is a synthesis of interoperable specifications derived from community ideas.

SCAP Content Metadata Model

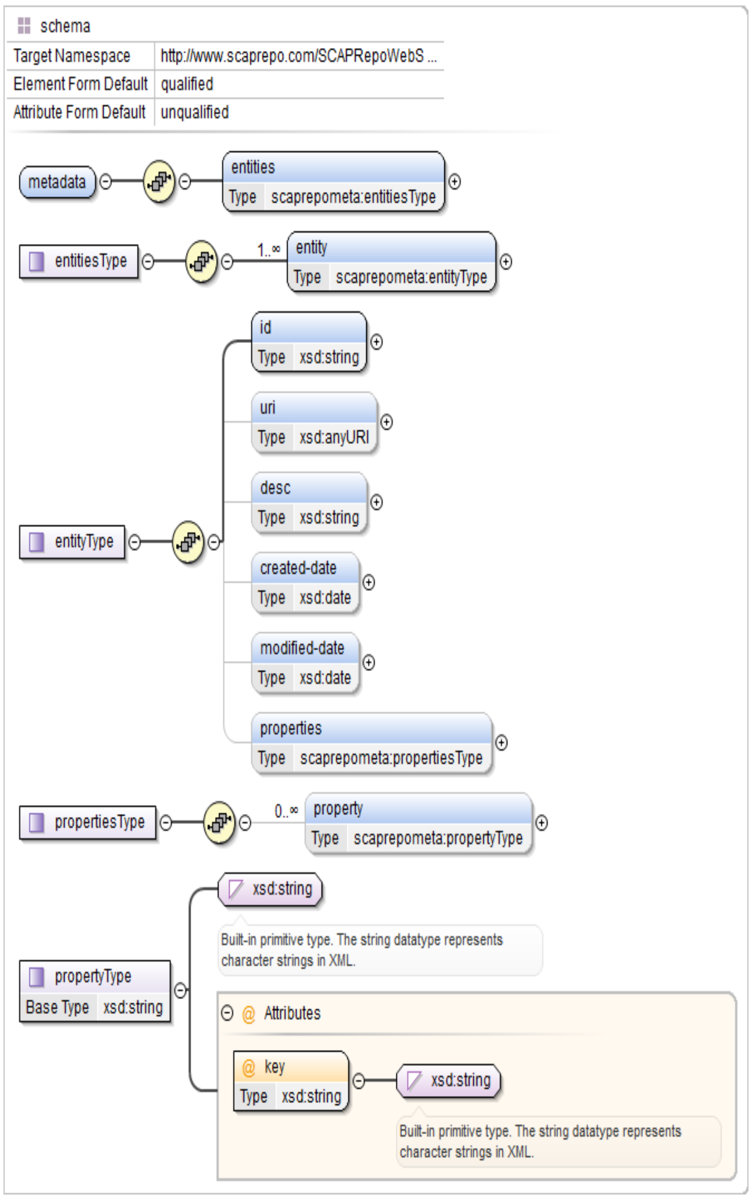
SCAP language and reporting format are expressed in XML format. XML helps exchange of complex information in a simple text structure over the web and it is highly interoperable. Besides these advantages, there are drawbacks in expressing SCAP language and reporting content in XML format. Large content articulated in the form of XML becomes bulky. Since the content is extensive in nature, search and analysis of this content becomes a challenging task.

This gives rise to the need of extracting relevant information in a standardized manner, which we refer to as 'metadata of security content'. This content has to be concrete and should be understood by receiver of information, machine or human to take crucial actions to fix a malware attack on the system.

Metadata of security content can be expressed in the form of key value pair. Each construct of SCAP content has a property; hence the key becomes the property name and value being the data after evaluation.

The diagram below outlines the metadata design of SCAP content:

Figure 1: Metadata schema diagram



Below is the XML schema definition of SCAP metadata:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:scaprepometa="http://www.scaprepo.com/SCAPRepoWebService/schema"targ
etNamespace="http://www.scaprepo.com/SCAPRepoWebService/schema"
attributeFormDefault="unqualified" elementFormDefault="qualified">
    <xsd:element name="metadata">
        <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="entities" type="scaprepometa:entitiesType"
            maxOccurs="1" minOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="entitiesType">
        <xsd:sequence>
            <xsd:element name="entity" type="scaprepometa:entityType"
            maxOccurs="unbounded" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="entityType">
        <xsd:sequence>
            <xsd:element name="id" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
           <xsd:element name="uri" type="xsd:anyURI" maxOccurs="1"
           minOccurs="0"/>
           <xsd:element name="desc" type="xsd:string" maxOccurs="1"
           minOccurs="0"/>
           <xsd:element name="created-date" type="xsd:date" maxOccurs="1"
           minOccurs="0"/>
           <xsd:element name="modified-date" type="xsd:date" maxOccurs="1"
           minOccurs="0"/>
           <xsd:element name="properties"
           type="scaprepometa:propertiesType" maxOccurs="1"
           minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="propertiesType">
        <xsd:sequence>
            <xsd:element name="property" type="scaprepometa:propertyType"
            maxOccurs="unbounded" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="propertyType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="key" type="xsd:string"
use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:schema>
```

Here is an example of vulnerability,

Adobe Flash Player (prior to 13.0.0.262 and 14.x through 16.x before 16.0.0.287 on Windows and OS X and prior to 11.2.202.438 on Linux) does not properly restrict discovery of memory addresses, which allows attackers to bypass the ASLR protection mechanism on Windows, and have an unspecified impact on other platforms, via unknown vectors, as exploited in the wild in January 2015.

Table 1: Impact of CVE-2015-0310

Property	Value
CVSS v2 Base Score	10.0(HIGH) (AV:N/AC:L/Au:N/C:C/I:C /A:C) (legend)
Impact Subscore	10.0
Exploitability Subscore	10.0

Table 2: CVSS Version 2 Metrics of CVE-2015-0310

Property	Value
Access Vector	Network exploitable
Access Complexity	Low **NOTE: Access Complexity scored Low due to insufficient information
Authentication	Not required to exploit
Impact Type	Allows unauthorized disclosure of information; Allows unauthorized modification; Allows disruption of service

Table 3 shows an example of metadata of CVE-2015-0310 that specifies vulnerability in Adobe Flash Player and earlier versions that has high severity score of 10.0.

Table 3: Metadata of CVE-2015-0310

Metadata	
Id	CVE-2015-0310
Desc	Adobe Flash Player before 13.0.0.262 and 14.x through 16.x before 16.0.0.287 on Windows and OS X and before 11.2.202.438 on Linux does not properly restrict discovery of memory addresses, which allows attackers to bypass the ASLR protection mechanism on Windows, and have an unspecified impact on other platforms, via unknown vectors, as exploited in the wild in January 2015.
URI	http://www.scaprepo.com/control.jsp?command=viewXML&id=CVE-2015-0310

Created-Date	2015-01-27
Modified-Date	2015-02-05
Score	10.0
Exploitability_score	10.0
Impact_score	10.0
Access_vector	NETWORK
Access_complexity	LOW
Availability_impact	COMPLETE
Authentication_status	NONE
Confidentiality_impact	COMPLETE
Integrity_impact	COMPLETE
Ext_ref	CONFIRM http://helpx.adobe.com/security/products/flash-player/apsb15-02.html
Published-Date	2015-01-23
Generated-Date	2015-01-26

The above data can be stored in databases, relational databases such as SQL or big data storage. Big data storage is designed to store large amounts of data. It stores information in key-value pair and supports efficient querying and information retrieval techniques. Though big data architecture and use plays an important role in storage of metadata, explanation of big data is out of the scope of the paper.

For information exchange, metadata can be expressed in XML format,

```xml
<metadata xmlns="http://www.scaprepo.com/SCAPRepoWebService/schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"xsi:schemaLocation="http://www.scaprepo.com/SCAPRepoWebService/schema
http://www.scaprepo.com/SCAPRepoWebService/schema/metadata.xsd">
    <entities>
        <entity>
            <id>CVE-2015-0310</id>
            <uri>
                http://www.scaprepo.com/control.jsp?command=viewXML&amp;id=CVE-
2015-0310
            </uri>
            <desc>
                Adobe Flash Player before 13.0.0.262 and 14.x through 16.x before
16.0.0.287 on Windows and OS X and before 11.2.202.438 on Linux does not properly
restrict discovery of memory addresses, which allows attackers to bypass the ASLR
protection mechanism on Windows, and have an unspecified impact on other
platforms, via unknown vectors, as exploited in the wild in January 2015.
            </desc>
            <created-date>2015-01-27</created-date>
            <modified-date>2015-02-05</modified-date>
            <properties>
                <property key="Desc">
                    Adobe Flash Player before 13.0.0.262 and 14.x through 16.x
before 16.0.0.287 on Windows and OS X and before 11.2.202.438 on Linux does not
properly restrict discovery of memory addresses, which allows attackers to bypass
the ASLR protection mechanism on Windows, and have an unspecified impact on other
platforms, via unknown vectors, as exploited in the wild in January 2015.
                </property>
                <property key="Score">10.0</property>
                <property key="Exploitability_score">10.0</property>
                <property key="Impact_score">10.0</property>
                <property key="Access_vector">NETWORK</property>
                <property key="Access_complexity">LOW</property>
                <property key="Availability_impact">COMPLETE</property>
                <property key="Authentication_status">NONE</property>
                <property key="Confidentiality_impact">COMPLETE</property>
                <property key="Integrity_impact">COMPLETE</property>
                <property key="Ext_ref">

                    http://helpx.adobe.com/security/products/flash-player/apsb15-
                    02.html CONFIRM
                    http://helpx.adobe.com/security/products/flash-player/apsb15-
                    02.html
                </property>
                <property key="Publisheddate">2015-01-23</property>
                <property key="Modifieddate">2015-01-26</property>
                <property key="Generateddate">2015-01-26</property>
                <property key="Created-Date">2015-01-27</property>
                <property key="Modified-Date">2015-02-05</property>
            </properties>
        </entity>
    </entities>
</metadata>
```

In this example, CVE-2015-0310 is related to product Adobe Flash Player versions 11.2.202.429, 16.0.0.257, 14.0.0.145, and 14.0.0.125 then we can certainly map them to product enumeration, CPEs: cpe:/a:adobe:flash_player:11.2.202.429, cpe:/a:adobe:flash_player:16.0.0.257, cpe:/a:adobe:flash_player:14.0.0.145, and cpe:/a:adobe:flash_player:14.0.0.125

Further, it can be associated to weakness enumeration CWE-264 - Permissions, Privileges, and Access Controls which contains detection methods and potential mitigation information and to vulnerability assessment signatures articulated in OVAL.

Searching on SCAP Metadata

The creation of SCAP Metadata judiciously reduces the size of security content to be examined. Subsequently, efficient search mechanisms can be implemented on this data to get information for vulnerability assessment and incident response information.

Each SCAP entity in this model is correlated to other relevant entities using references as described in previous section. With this information, tracing the query across different entities of SCAP becomes straight-forward.

For instance, a vulnerability search results in CVEs and each CVE has a detection signature that is written in OVAL. Each vulnerability and detection signature is mapped to a remediation patch or hardening measure. This helps fix an issue before an attack happens, and thus acts as the first line of defence against malware.

Natural Language Processing (NLP) can be implemented that can define linguistics of security content search interactions.

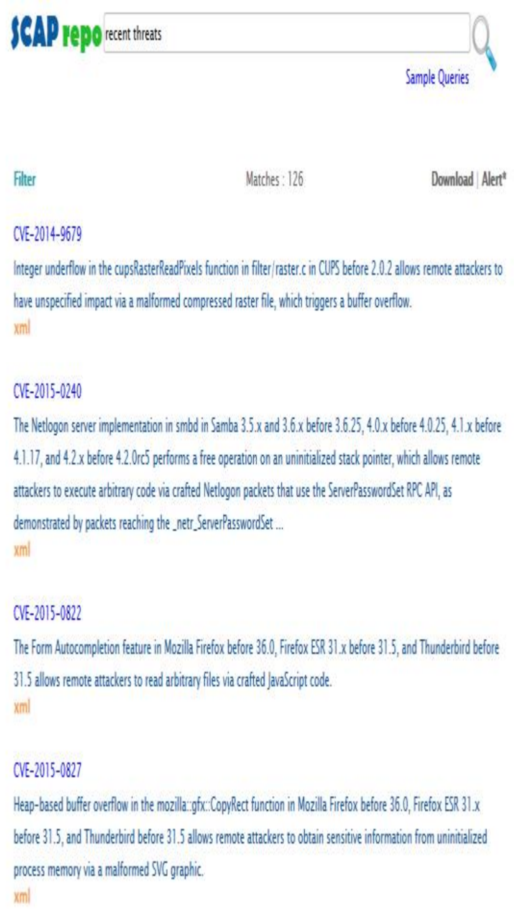
Search implementation can also be enhanced to include date/time related queries.

Examples of Metadata Search

Below are a few examples of metadata searches:

Search query "Recent threats" lists a set of latest vulnerabilities that might be a threat to your organization

Figure 2: Search query 'Recent threats'



CVE-2014-9679

Integer underflow in the cupsRasterReadPixels function in filter/raster.c in CUPS before 2.0.2 allows remote attackers to have unspecified impact via a malformed compressed raster file, which triggers a buffer overflow.

xml

CVE-2015-0240

The Netlogon server implementation in smbd in Samba 3.5.x and 3.6.x before 3.6.25, 4.0.x before 4.0.25, 4.1.x before 4.1.17, and 4.2.x before 4.2.0rc5 performs a free operation on an uninitialized stack pointer, which allows remote attackers to execute arbitrary code via crafted Netlogon packets that use the ServerPasswordSet RPC API, as demonstrated by packets reaching the _netr_ServerPasswordSet ...

xml

CVE-2015-0822

The Form Autocompletion feature in Mozilla Firefox before 36.0, Firefox ESR 31.x before 31.5, and Thunderbird before 31.5 allows remote attackers to read arbitrary files via crafted JavaScript code.

xml

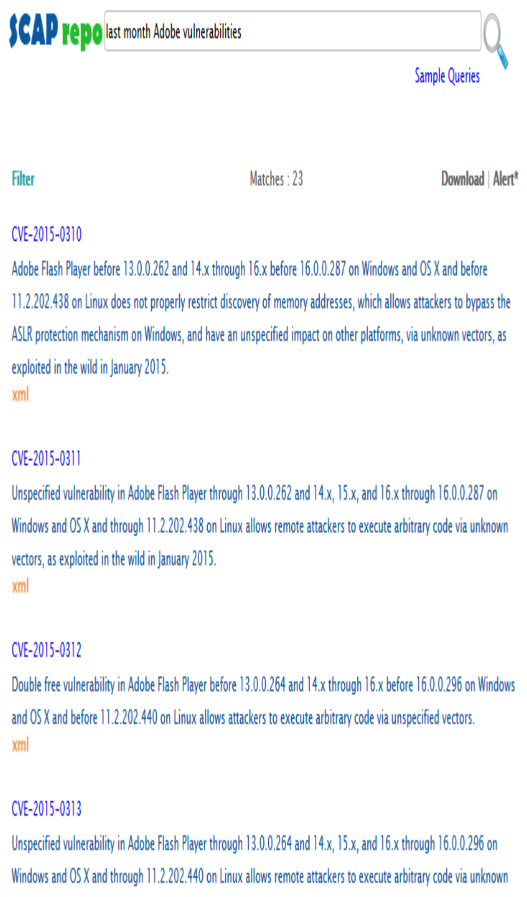
CVE-2015-0827

Heap-based buffer overflow in the mozilla::gfx::CopyRect function in Mozilla Firefox before 36.0, Firefox ESR 31.x before 31.5, and Thunderbird before 31.5 allows remote attackers to obtain sensitive information from uninitialized process memory via a malformed SVG graphic.

xml

Another search query "last month adobe vulnerabilities" lists all vulnerabilities reported in the previous month. (See Figure 3)

Figure 3: Search query 'last month adobe vulnerabilities'


SCAP repo last month Adobe vulnerabilities

Sample Queries

Filter Matches : 23 Download | Alert*

CVE-2015-0310
Adobe Flash Player before 13.0.0.262 and 14.x through 16.x before 16.0.0.287 on Windows and OS X and before 11.2.202.438 on Linux does not properly restrict discovery of memory addresses, which allows attackers to bypass the ASLR protection mechanism on Windows, and have an unspecified impact on other platforms, via unknown vectors, as exploited in the wild in January 2015.
xml

CVE-2015-0311
Unspecified vulnerability in Adobe Flash Player through 13.0.0.262 and 14.x, 15.x, and 16.x through 16.0.0.287 on Windows and OS X and through 11.2.202.438 on Linux allows remote attackers to execute arbitrary code via unknown vectors, as exploited in the wild in January 2015.
xml

CVE-2015-0312
Double free vulnerability in Adobe Flash Player before 13.0.0.264 and 14.x through 16.x before 16.0.0.296 on Windows and OS X and before 11.2.202.440 on Linux allows attackers to execute arbitrary code via unspecified vectors.
xml

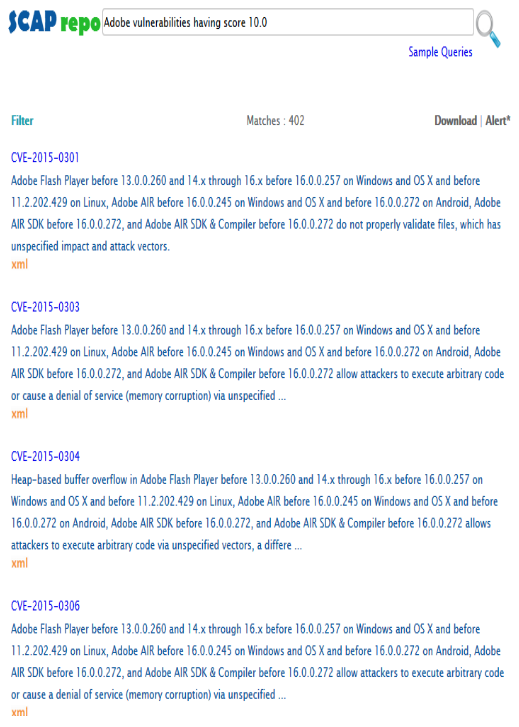
CVE-2015-0313
Unspecified vulnerability in Adobe Flash Player through 13.0.0.264 and 14.x, 15.x, and 16.x through 16.0.0.296 on Windows and OS X and through 11.2.202.440 on Linux allows remote attackers to execute arbitrary code via unknown

The above query lists all vulnerabilities. In order to further refine this search, we can look for vulnerabilities which have a high severity score.

Such queries can assist system administrators to identify which vulnerabilities are critical and need to be addressed quickly.

For a search query "Adobe vulnerabilities having score 10.0", see figure 4)

Figure 4: Search query 'Adobe vulnerabilities having score 10.0'


SCAP repo Adobe vulnerabilities having score 10.0

Sample Queries

Filter Matches : 402 Download | Alert*

CVE-2015-0301
Adobe Flash Player before 13.0.0.260 and 14.x through 16.x before 16.0.0.257 on Windows and OS X and before 11.2.202.429 on Linux, Adobe AIR before 16.0.0.245 on Windows and OS X and before 16.0.0.272 on Android, Adobe AIR SDK before 16.0.0.272, and Adobe AIR SDK & Compiler before 16.0.0.272 do not properly validate files, which has unspecified impact and attack vectors.
xml

CVE-2015-0303
Adobe Flash Player before 13.0.0.260 and 14.x through 16.x before 16.0.0.257 on Windows and OS X and before 11.2.202.429 on Linux, Adobe AIR before 16.0.0.245 on Windows and OS X and before 16.0.0.272 on Android, Adobe AIR SDK before 16.0.0.272, and Adobe AIR SDK & Compiler before 16.0.0.272 allow attackers to execute arbitrary code or cause a denial of service (memory corruption) via unspecified ...
xml

CVE-2015-0304
Heap-based buffer overflow in Adobe Flash Player before 13.0.0.260 and 14.x through 16.x before 16.0.0.257 on Windows and OS X and before 11.2.202.429 on Linux, Adobe AIR before 16.0.0.245 on Windows and OS X and before 16.0.0.272 on Android, Adobe AIR SDK before 16.0.0.272, and Adobe AIR SDK & Compiler before 16.0.0.272 allows attackers to execute arbitrary code via unspecified vectors, a differe ...
xml

CVE-2015-0306
Adobe Flash Player before 13.0.0.260 and 14.x through 16.x before 16.0.0.257 on Windows and OS X and before 11.2.202.429 on Linux, Adobe AIR before 16.0.0.245 on Windows and OS X and before 16.0.0.272 on Android, Adobe AIR SDK before 16.0.0.272, and Adobe AIR SDK & Compiler before 16.0.0.272 allow attackers to execute arbitrary code or cause a denial of service (memory corruption) via unspecified ...
xml

How to use metadata?

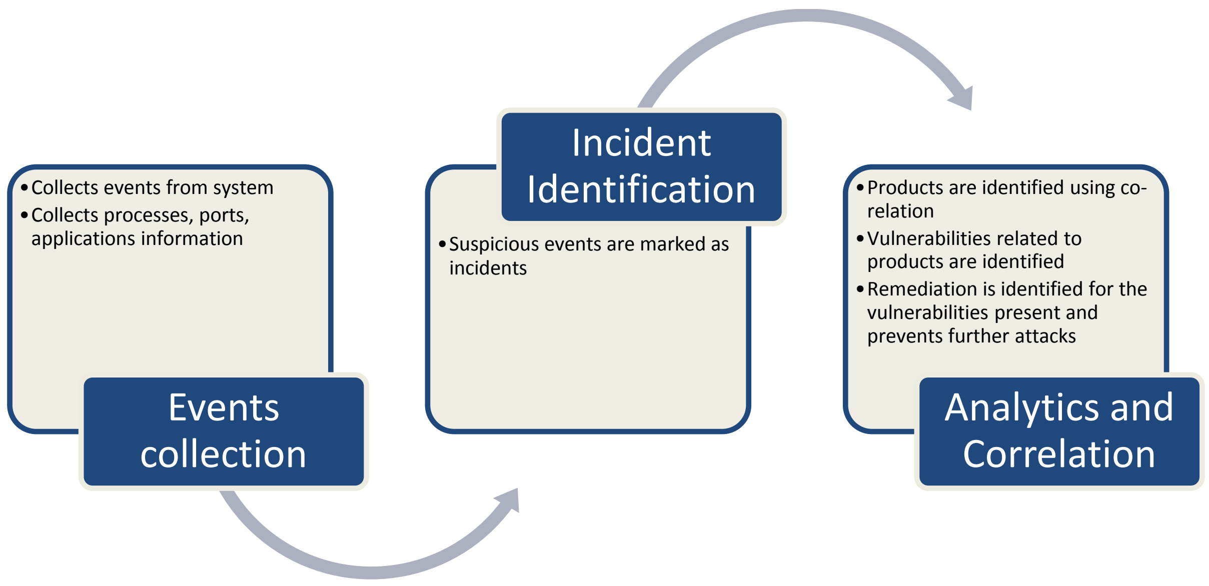
Local and remote scan can be performed on systems to collect system information and produce results in a standard format. Based on system metadata, data analytics can be implemented on results content to understand and co-relate various parameters of data. For instance, risk can be computed based on the count and score of vulnerabilities present in each system of an organization.

This metadata also supports storage of organizations' benchmark, compliance data such as PCI DSS, ISO, and HIPAA and compliance checks that need to be performed on the systems to know if systems are yielding to the benchmark set by the organization.

Once the data is collected from various systems, a holistic view of the security posture of the organization can be observed from metrics, figures and plotting graphs.

Also, if an incident occurs in the system, we can co-relate various entities of SCAP and fix the issue.

Figure 5 shows the flow of incidence response mechanism.



Collects events from system
Collects processes, ports, applications information
Events collection
Incident Identification
Suspicious events are marked as incidents
Products are identified using co-relation
Vulnerabilities related to products are identified
Remediation is identified for the vulnerabilities present and prevents further attacks
Analytics and Correlation

Figure 5: Flow of Incidence Response

Real Time Threat Monitoring using Security Content Metadata Model

We measure the security posture of an organization using various assessments and auditing products. Each product produces large data sets that are limited to the nature of the product.

"How do we identify incidents in a sea of data?"

SCAP metadata along with malware characteristics and events data from systems defines the groundwork of incident response framework. A scan report (see Reporting Format Section) consists of data collected from the system. Once we get the required data, we can arrive at an actionable measure to fix the issue.

This can be explained with an example:

A bizarre exe inj_adb.exe exists in the system and was executed with PID as an argument. This PID is the process ID of a sandboxed Adobe Reader process. As a result, a new file is created named 'adbe'. This appears suspicious and requires some action.

When system information is collected and events are reported, we can co-relate this incident with the product Acrobat Reader. Using system information, it will be clear that the version of Acrobat Reader in Windows is 11.0.8, which is vulnerable to attacks.

The CVE explains that the Acrobat Reader Windows sandbox is vulnerable to NTFS junction attack. An NTFS junction point is a symbolic link to a directory that acts as an alias of that directory. This vulnerability allows malware to write an arbitrary file to the file system under user permissions. This could be used to break out of the sandbox leading to execution at higher privileges. This is marked as CVE-2014-9150.

Remediation suggests removal of adbe file, inj_adb.exe and unwanted .dll present in the location of the sandboxed process, in conjunction to upgrading this Adobe product to fix known vulnerabilities. Common Remediation Enumeration CRE gives us the patch to be installed to upgrade Adobe Acrobat Reader.

This vulnerability is related to weakness enumeration CWE-362, Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition').

From these CWE, we arrive at attack patterns CAPDEC-26 and CAPDEC-29. CAPDEC stands for Common Attack Pattern Enumeration and Classification, a community resource for identifying and understanding attacks.

Additionally, this scenario of ascertaining unwanted behaviour and fixing issue caters to safeguarding us against attacks in future.

Summary

The metadata model stores security intelligence. Due to efficient analytics and correlation of various SCAP entities, it reduces the time of attack detection and remediation of the outbreak. 90% of the attacks make use of vulnerabilities that exist in the computer systems. Due to the complexity of attacks and enormous volume of data it may not be possible to quickly identify vulnerabilities and attacks and take appropriate actions to remediate those weaknesses. If remediation fixes take days and weeks to execute, this opens a large window for attackers to exploit our systems. Hence, it is vital to work on a small set of information crafted as metadata that accommodates meaningful information to trace an attack, resolve and protect our systems from future attacks.

What next?

This metadata model can be used as a foundation to design complex attack detection and prevention mechanisms. This will be inevitable in near future because hackers are designing novel techniques of attacking and compromising the system. Mechanisms consuming this metadata should be able to relate events from a system report them as incidents in case an action is needed, and eventually stop an attack from happening. As the underlying SCAP content evolves, metadata schema should be able to accommodate various attributes of attack detection and protection.

Acronyms and Abbreviations

SCAP	Security Content Automation Protocol
CVE	Common Vulnerabilities and Exposures
CCE	Common Configuration Enumeration
CPE	Common Platform Enumeration
CRE	Common Remediation Enumeration
XCCDF	Extensible Configuration Checklist Description Format
OVAL	Open Vulnerability and Assessment Language
MAEC	Malware Attribute Enumeration and Characterization
SIEM	Security Incidents and Events Management
CVSS	Common Vulnerability Scoring System
CCSS	Common Configuration Scoring System
XML	Extensible Mark-up Language
XSD	XML Schema Definition
ASLR	Address space layout randomization

References

- The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2 (September 2011) http://csrc.nist.gov/publications/nistpubs/800-126-rev2/SP800-126r2.pdf
- National Cyber Awareness System Vulnerability Summary http://web.nvd.nist.gov/
- Metadata repository https://www.scaprepo.com/SCAPRepoWebService
- Adobe Security Bulletin http://helpx.adobe.com/security/products/flash-player/apsb15-02.html

Author:

Preeti Subramanian
Working as Software Architect at SecPod Technologies
Bangalore, India

Email: spreeti@secpod.com