Standardization Efforts in Computer Security (I)

CVE is orthogonal to other standards

#2 CVE

The hacker exploited vuln. CVE-xxxx.yyyy

This advisory describes vuln. CVE-xxxx.yyyy

This host has vuln. CVE-xxxx.yyyy

How to remove vuln. CVE-xxxx.yyyy

EISPP/DAF

OVAL

IODEF/INCH

XCCDF

This advisory describes vuln. CVE-xxxx.yyyy

How to remove vuln. CVE-xxxx.yyyy

Standardizing System Information

Bernd Grobauer
Siemens CERT
Standardization Efforts in Computer Security (II)
System Information is orthogonal as well

EISPP/DAF
The described vuln. affects OS XYZ

IODEF/INCH
The attacked host ran application XYZ

OVAL
This is a check for a host with OS XYZ

XCCDF
This measure works for OS XYZ

System Information

The described vuln. affects OS XYZ
The attacked host ran application XYZ
This is a check for a host with OS XYZ
This measure works for OS XYZ

How System Information is treated in existng standards. Example: EISPP/DAF (I)

ASN.1 Vulnerabilities

TID: ASW.1 Vulnerabilities

Risk:
very high □ high □ medium □ low □ very low

System:

Description:
Multiple integer overflow vulnerabilities in the Microsoft Windows ASN.1 parser library (version 1.1b) could allow a remote attacker to execute arbitrary code with SYSTEM privileges.

Solution:

Reference:

© Siemens AG, CTIC CERT, B. Grobauer
How System Information is treated in existing standards. Example: EISPP/DAF (II)

- EISPP/DAF treats system information as a list of free-text fields, each associated with a tag describing the content:

<table>
<thead>
<tr>
<th>Content Type:</th>
<th>platform</th>
<th>software ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows Server 2003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Windows XP Professional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Windows 2000 (…)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Content Type:</th>
<th>platform</th>
<th>software ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft ASW.1 Library</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- As a result, no automated handling of system info. is possible
- What is needed is a model of system information that describes how to specify machine-readable system information
- Furthermore, a common model of system information is needed

This talk

This talk presents the results achieved by a working group of German CERTs (CERT-Bund, DFN-CERT, PreCERT, Siemens CERT) under the auspices of the “Deutscher CERT Verbund”

Structure of the talk:
- Definition of a Common Model of System Information (CMSI)
- Constraints on a CMSI
- Using the CMSI: process and examples
- Structure of the CMSI
- Closing remarks
Model of System Information -- A Definition

- System Information must be provided consistently:
  - What is called "Microsoft Explorer v6.0" in yesterday's advisory should not be called "MS Internet Explorer (version 6.000)" in today's advisory.
- A "Model of System Information" specifies, how system information is provided.

Examples:
- Tacit Knowledge: "Unwritten rules" (maybe supported by copy and paste from older advisories) regulate how affected systems are called.
- Tool support: An authoring system for advisories constrains the way in which affected systems are specified, e.g., by providing a list to choose from.

Definition: A model of system information consists of a dictionary of identifiers and (syntactic) rules for expressing information about computer systems (usually a combination of OS and application software).

Constraints on a CMSI

- Machine readable vs. human readable information
  - Should the common model deal with machine-readable information, human-readable information, or both?
- Relationship to existing models
  - How should the common model relate to already existing, proprietary models?
- Maintenance
  - What are the dynamics of system information and how much effort is necessary to keep a common model up-to-date?
Constraints (I)
Machine-readable vs. Human-readable Information

- Often, two models of system information are maintained:
  - Human-readable information
  - Machine-readable information
- Filtering and Correlation require machine-readable information
- Form and shape of human-readable information is highly constituency-dependent

⇒ Common model should include machine-readable information

Constraints (II)
Relationship to Existing Models

- Models of system information exist (in some form) with any provider of system information
- A common model will not be able to satisfy all possible demands

⇒ Proprietary models will continue to exist
⇒ Use of common model requires mappings from/to proprietary models

Mappings are, by nature, proprietary, but structure and contents of model must facilitate mappings!

- Must be possible to give very “coarse” information (e.g., “Windows is affected”)
  - some organizations do not keep much more precise information
  - some organizations may not want to put much effort into mapping a detailed proprietary model into the CMSI
- Must be possible to give very detailed information (e.g., “Apache 1.3.27 on Windows 2000 SP2 is affected”) to allow more sophisticated applications
Constraints (III)

Maintenance Issues

- New products / new versions are issued on a daily basis
- Changes in the product landscape must be mirrored by the common model
- Effort necessary of maintaining a common model depends on
  - level of detail contained in model
  - requirements on accuracy of data contained in model
  - processes/actors defined for maintaining the model
  - tool support provided for maintaining the model

⇒ Maintenance issues must be considered as one of the prime design criteria for a common model

Using CMSI (I)

The process of using CMSI

- CMSI is maintained at a central location
  - a maintainer/group of maintainers handles change requests, additions, ...
  - the model's contents can be viewed online for reference
- Organizations that want to use CMSI
  - regularly download the most recent version (XML-based exchange format for communicating the model contents)
  - adapt their proprietary model:
    • either define mappings from proprietary model into CMSI and vice versa
    • or switch to the CMSI also for internal use
- Organizations uses CMSI by communicating system information by filling in an XML-template with CMSI-compliant data
  • XML-template already part of EISPP/DAF and could be easily integrated into other standards, as well.
Using CMSI (II)
A very simple example

- Message: “Apache (on all platforms) is affected.”
- CMSI provides identifiers “os” and “apache”

```
<system_list>
  <system>
    <system_part type="platform">
      <instance tag="os"/>
    </system_part>
    <system_part type="software">
      <instance tag="apache"/>
    </system_part>
  </system>
</system_list>
```

```
Message: “Apache (on all platforms) is affected.”
```

Snippet from DAF-advisory

Using CMSI (II)
A not so simple example

- Message: “Apache 1.3.x and 2.x on Windows XP, and Apache 2.x on Unix are affected.”
- CMSI provides identifiers “w2k”, “wxp”, “unix”, “apache”, and “version” and syntax rules to give version information such as “1.3.x”, “2.x”

```
<system_list>
  <system>
    <system_part type="platform">
      <instance tag="w2k"/>
      <instance tag="wxp"/>
    </system_part>
    <system_part type="software">
      <instance tag="apache">
        <attribute_value tag="version">
          <value>1.3.x</value>
          <value>2.x</value>
        </attribute_value>
      </instance>
    </system_part>
  </system>
  <system>
    <system_part type="platform">
      <instance tag="unix"/>
    </system_part>
    <system_part type="software">
      <instance tag="apache">
        <attribute_value tag="version">
          <value>2.x</value>
        </attribute_value>
      </instance>
    </system_part>
  </system>
</system_list>
```

```
Message: “Apache 1.3.x and 2.x on Windows XP, and Apache 2.x on Unix are affected.”
```

CMSI provides identifiers “w2k”, “wxp”, “unix”, “apache”, and “version” and syntax rules to give version information such as “1.3.x”, “2.x”

Snippet from DAF-advisory
Structure of CMSI (I)
Overview

Category-Tree

System

OS

Windows

Unix-like

Mainframe

... Server

Client

Application

Category Tree serves several purposes:

- Users of the model should be facilitated in finding their way around
  - Tree should not be nested too deeply!
- Category nodes can be used for (very) coarse system information
- Category nodes such as "Server" and "Client" can be used for creating user profiles (e.g.: "Tell me about vulnerabilities in server products only")
- Implementing and using the category tree is not much effort but already brings benefits: it allows expressing and filtering with respect to information such as
  - "Windows is affected"
  - "Unix is affected"
  - "A web-server product on Windows is affected"
Structure of CMSI (III)
Product Families

Think of a product family as a flashcard:

**MS Windows 2000 (w2k)**

*Products:*
- MS Windows 2000 Workstation (w2k:ws)
- MS Windows 2000 Server (w2k:server)
- MS Windows 2000 Advanced Server (w2k:aserver)
- MS Windows 2000 Datacenter Server (w2k:data)
  
*Attributes for this family:*
- patchlevel: SP[0-9]+  

Structure of CMSI (IV)
Product Families

- **Product family** comprises one or more closely related products. Consequently,
  - the same vulnerability will often affect all members of a product family
  - Version information (version number, patch level, etc.) is given in a similar fashion for all members of a product family
  
⇒ "Product family" is the right level of abstraction for a common model of system information:
  - In many cases, information of type "product family X is affected" will be precise enough
  - Syntactic rules for providing, e.g., version information, can be given on a per-family basis
  
- **One product family can be the child of several category node**
  ⇒ Ambiguities in the tree can be worked around

© Siemens AG, CTIC CERT, B. Grobauer
Constraints on a CMSI -- revisited

- **Common model should include machine-readable information**
  - Unique identifiers and syntactic rules provide for machine-readable information
  - Changes concerning human-readable names are no problem: computer-readable identifier stays the same
  - More than one human-readable name can be given to assure that users of the model find products under the name they are used to

- **Mappings are, by nature, proprietary, but structure and contents of model must facilitate mappings!**
  - Coarse mappings possible by mapping to categories or product families
  - Very fine-grained mappings possible by mapping to products & significant attribute information (version info., etc.)

- **Maintenance issues must be considered as one of the prime design criteria for a common model**
  - Because model treats version information "only" by supplying rules for specifying version information, maintenance effort seems manageable: release of new version usually will not trigger any changes in model.

Status of CMSI

- **Structure of CMSI**
  - An XML schema for describing the contents of the common model (category tree and product families) has been defined. It is described in a FIRST 2005 paper and available from the CMSI home page ([http://www.cert-verbund.de/cmsi](http://www.cert-verbund.de/cmsi)).
  - An XML schema for including system information based on CMSI has been defined and is already part of the EISPP/DAF advisory exchange formats ([see http://www.cert-verbund.de/daf/daf_description.html](http://www.cert-verbund.de/daf/daf_description.html)).
  - CMSI has been tightly integrated into the open-source development of the incident handling system SIRIOS, a project of CERT Bund ([see http://www.cert-verbund.de/sirios](http://www.cert-verbund.de/sirios)). SIRIOS also supports IODEF and EISPP/DAF.

- **Contents of CMSI**
  - Category tree agreed upon within CMSI-working group of the German CERT association (modulo some cleaning up...)
  - At the moment, the category tree is being filled with the most important product families
Before I forget: What's the story behind the logo?

Like bees, we want to communicate useful information with means that are as simple as possible and yet effective...

Closing Remarks: What to remember from this talk?

- Like CVE, system information is orthogonal to all standardization efforts in computer security (and other fields, as well).
- If you need to build a model for providing machine-readable system information in a consistent way: check, whether CMSI meets your demands.
- If you do not want to start from scratch with filling your model, check the status of the model being built by the German CERT working group.
- If you agree that a truly common model of system information in the fashion of CVE should exist, talk to us and let’s join forces.

Further information:

- http://www.cert-verbund.de/cmsi
- bgrobauer@cert.siemens.de