

# Towards a Common Model of System Information



Bernd Grobauer  
Siemens CERT





# History: From EISPP to CMSI (I)

For internal use only!

Siemens CERT Security Telegram PC 005/04

**ASN.1 Vulnerabilities**

Date: 2004-02-11

Priority: 1

SCCS: ADPR

**PLATFORM:**  
 Microsoft Windows Server 2003  
 Microsoft Windows XP Professional  
 Microsoft Windows 2000  
 Microsoft Windows NT 4.0  
 Microsoft Windows NT 4.0 Terminal Server Edition

**SOFTWARE:** Microsoft ASN.1 Library

**DESCRIPTION:**  
 Multiple integer overflow vulnerabilities in the Microsoft Windows ASN.1 parser library (msasn1.dll) could allow an remote attacker to execute arbitrary code with SYSTEM privileges.

**PATCHES/WORKAROUNDS:**

- Patch for Microsoft Server 2003 (32bit Version, English)  
 Intranet [EN32\\_PC00504\\_Q828028\\_MS04007\\_DS.EXE](#)  
 Internet [WindowsServer2003-KB828028-x86-ENU.exe](#)

**STANDARD VULN-IDS:**

CVE Number: CAN-2003-0818 →

**Title:** ASN.1 Vulnerabilities

**Risk:**  very high  high  medium  low  very low

**System:**



**Description:**

**Content Type:**  description  diagnostic ...  \_\_\_\_

Multiple integer overflow vulnerabilities in the Microsoft ASN.1 Library ...

**Solution:**

...

**Reference:**

**Ref. Type:**  vuln. id.  advisory ...

**Issuer ID:**  CVE  BID ...  \_\_\_\_

**Ref. Num:** CAN-2003-0818





## History: From EISPP to CMSI (II)

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

|  |  |
|--|--|
| <b>System:</b>   |  |
| <b>Content Type:</b>   | <input checked="" type="checkbox"/> platform <input type="checkbox"/> software ... <input type="checkbox"/> ____ |
| Microsoft Windows Server 2003<br>Microsoft Windows XP Professional<br>Microsoft Windows 2000 (...) |  |
| <b>Content Type:</b>   | <input type="checkbox"/> platform <input checked="" type="checkbox"/> software ... <input type="checkbox"/> ____ |
| Microsoft ASN.1 Library  |  |

- As a result, no automated handling of system info. is possible
- EISPP envisions the addition of a model for specifying system information in a machine-readable way.





## History: From EISPP to CMSI (III)

- **In September 2003, the “Advisory Working Group” is established within the “German CERT Working Group”**
  - active members: CERT-Bund, DFN-CERT, PreSecure, Siemens-CERT
  - feedback (amongst others) from Bayern-CERT, Dt. Telekom, S-CERT, Secunet
- **Tasks:**
  - give input to EISPP consortium for design of EISPP advisory format
  - work towards a common model of system information (CMSI)
    - that allows exchange of useful information about affected systems
    - that can be used together with EISPP (chosen as a basis for German CERT-cooperation) or any other structured advisory format
- **Results:**
  - structure of CMSI has been designed and agree upon
  - process of “filling” the model has started





## This talk

- **Definition of a CMSI**
- **Possible Applications of a 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:

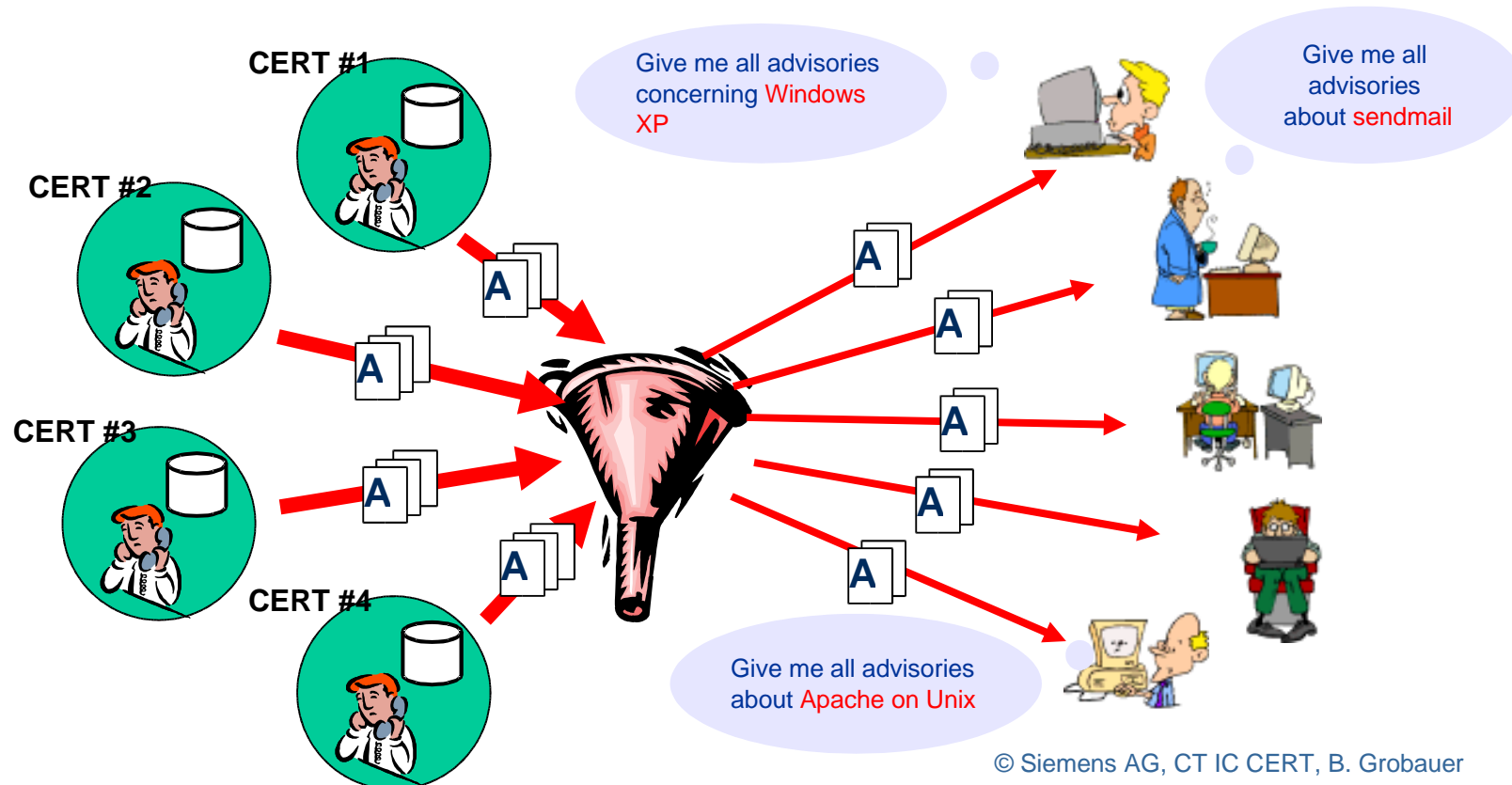
- **Tacit Knowledge:** "Unwritten rules" (maybe supported by *copy and paste* from older advisories) regulate how affected systems are called
- **Tool support:** An authoring system for advisories constrains the way in which affected systems are specified, e.g., by providing a list to chose from
- **Definition:**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





## Applications of CMSI (I) Filtering

- System information is one of the prime criteria for establishing whether an advisory is applicable
- ⇒ Filtering with respect to system info. as prime application

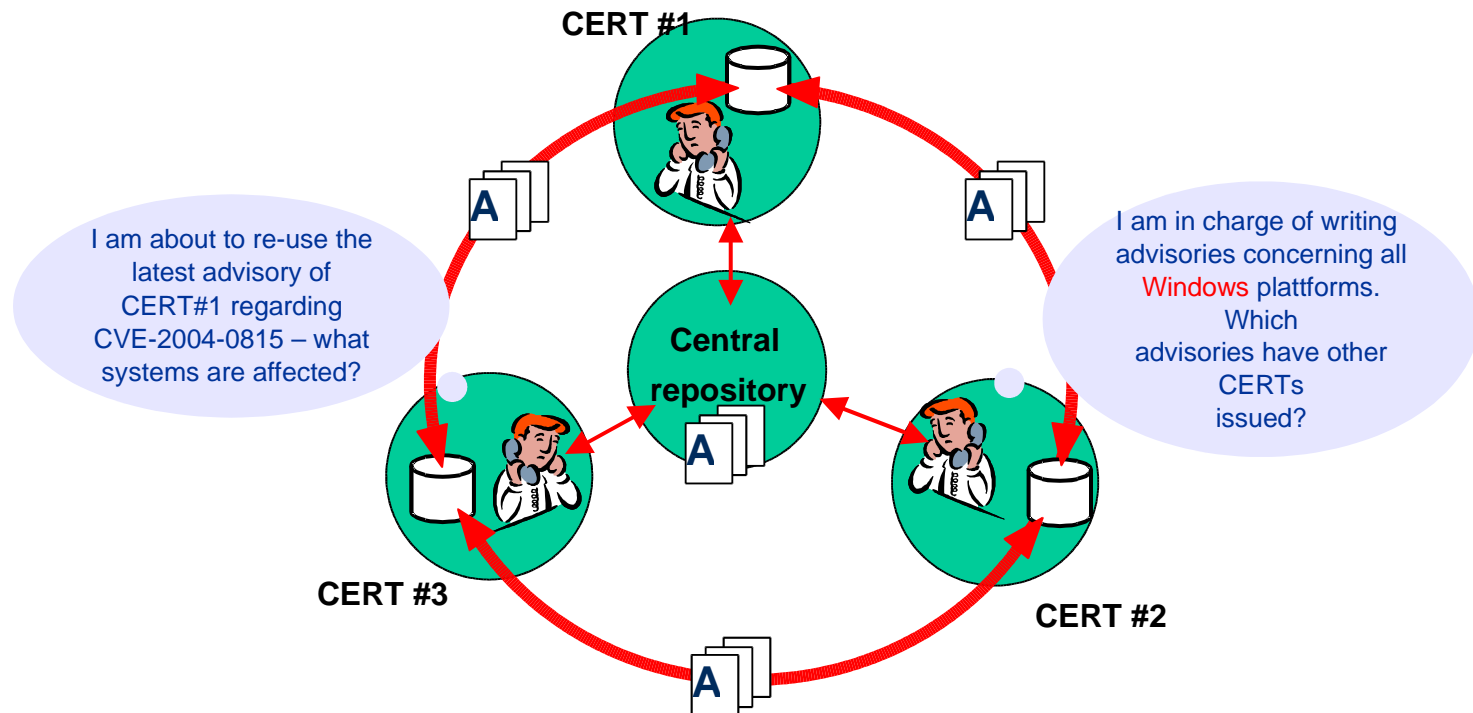




## Applications of CMSI (II)

### Correlation and co-operation on advisories

- Closer co-operation between CERTs regarding advisories requires a "common language" for correlating their findings
- ⇒ Correlation of system information as future application

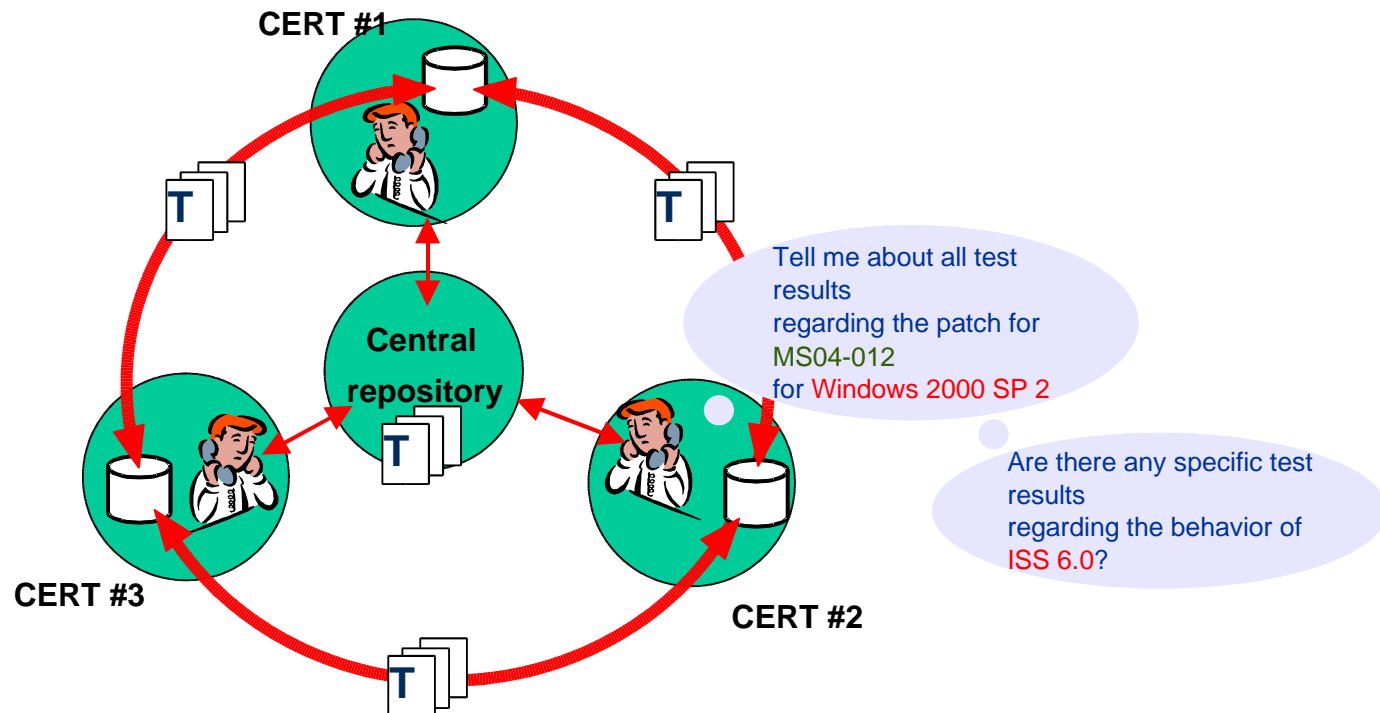




## Applications of CMSI (III)

### Correlation and collections of patch-test results

- Testing patches is a serious bottle neck in the patch process
- ⇒ Collection and correlation of test results could help





## Constraints on a CMSI

- **Applications of a common model**

- **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 treat machine-readable information**

```

Microsoft Internet Explorer 5.0
- Microsoft Windows 2000 Workstation
- Microsoft Windows 2000 Workstation
  SP1
- Microsoft Windows 2000 Workstation
  SP2
- Microsoft Windows 95
- Microsoft Windows 98
+ Microsoft Windows 98SE
- Microsoft Windows NT 4.0 SP3
- Microsoft Windows NT 4.0 SP4
- Microsoft Windows NT 4.0 SP5
- Microsoft Windows NT 4.0 SP6
- Microsoft Windows NT 4.0 SP6a
Microsoft Internet Explorer 5.0.1 SP3
Microsoft Internet Explorer 5.0.1 SP2
- Microsoft Windows 2000 Advanced
  Server
- Microsoft Windows 2000 Advanced
  Server SP1
- Microsoft Windows 2000 Advanced
  Server SP2
- Microsoft Windows 2000 Datacenter
  Server
(...)
  
```





## 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 CERTs do not keep much more precise information
  - some CERTs 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 the 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
- **CERTs who 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
- **CERT uses CMSI by communicating system information by filling in an XML-template with CMSI-compliant data**
  - XML-template part of advisory format such as EISPP or somehow embedded linked to other data, e.g., test result for patches





## Using the CMSI (II)

### A very simple example

```
<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”
- **CMSI provides identifiers “os” and “apache”**





## Using the CMSI (III)

### A not so simple example

```

<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>

```

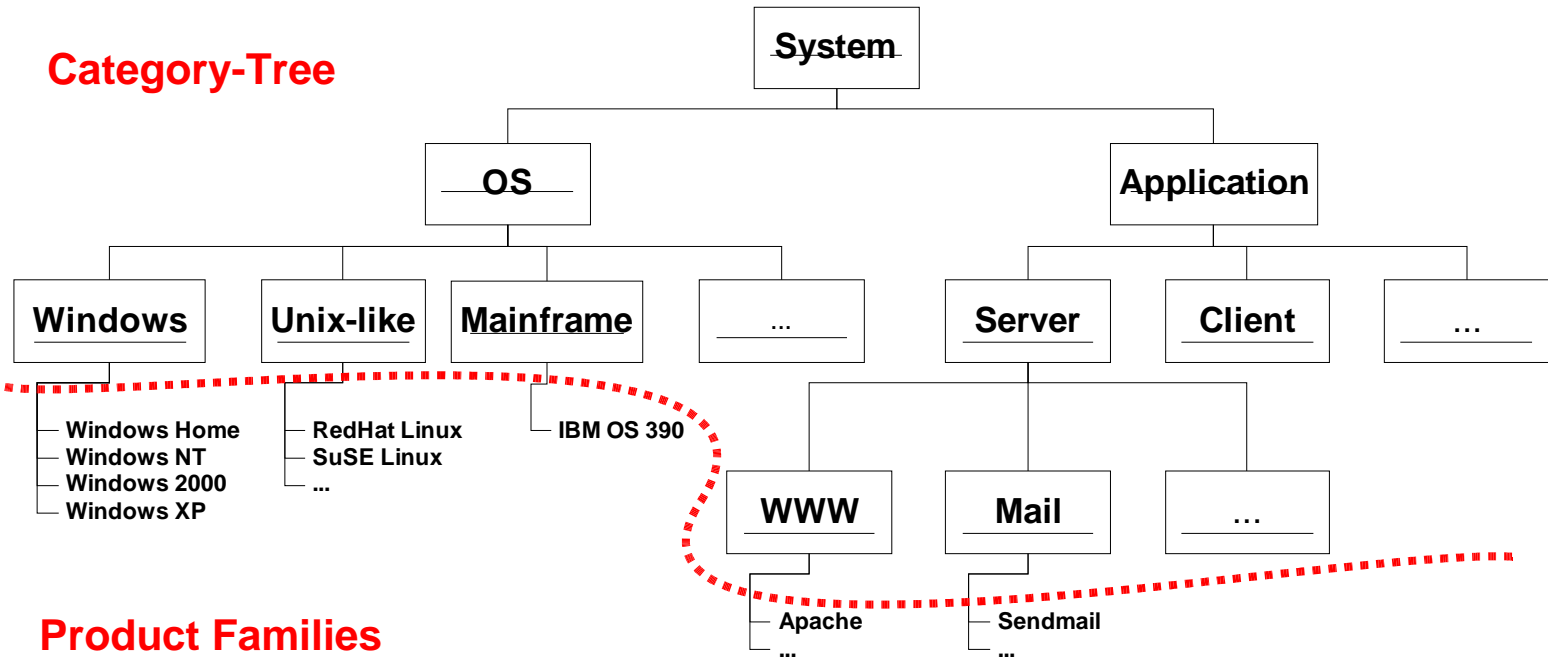
- **Message:**  
 “Apache 1.3.x and 2.x on Windows 2000 and Windows XP, and Apache 2.x on Unix are affected.”
- **CMSI provides**
  - identifiers “w2k”, “wxp”, “unix”, and “apache”
  - identifier “version” and syntax rules to give version information such as “1.3.x”, “2.x”





# Structure of CMSI (I) Overview

## Category-Tree

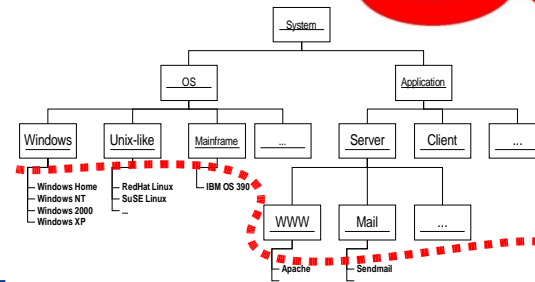


## Product Families





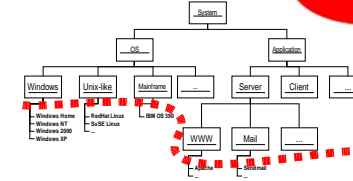
## Structure of CMSI (II) Category Tree



- **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 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]+

Language: [A-Z][A-Z] (ISO-649 lang. codes)

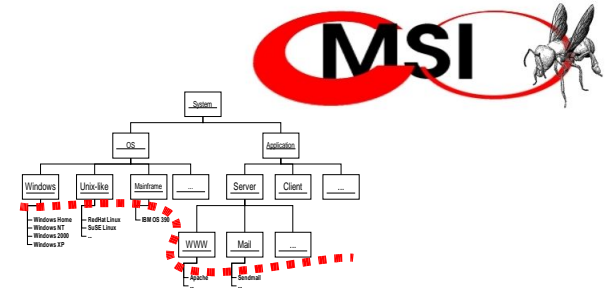
Unique identifier

Regular expressions

Explanation how to use attribute / attr. semantics



# Structure of CMSI (IV) Product Families

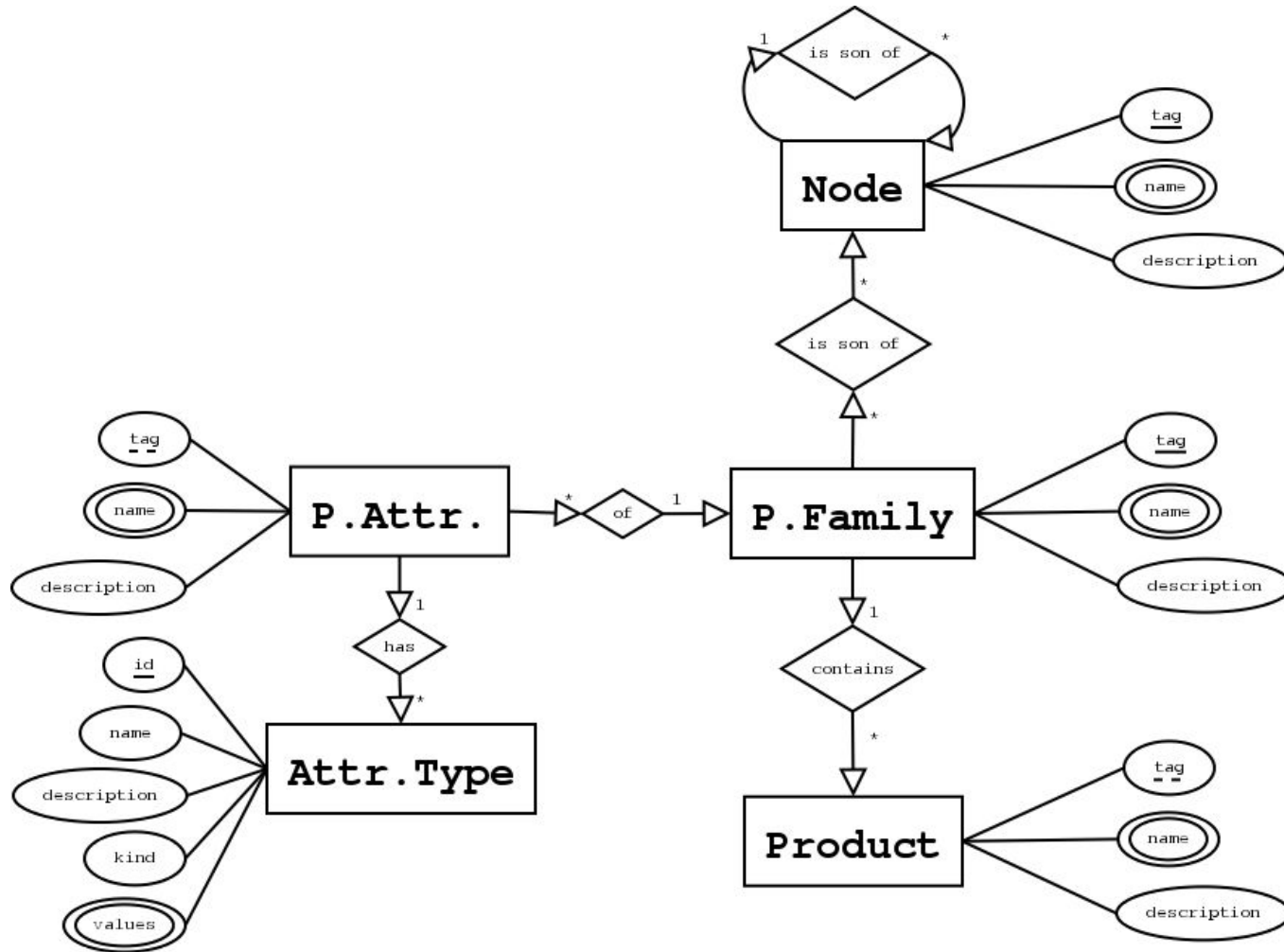


- **Product family comprises one or more closely related products. Consequently,**
  - the same vulnerability will often affect all members of a product family
  - Version information (version number, patch level, etc.) is given in a similar fashion for all members of a product family
- ⇒ **"Product 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





# Structure of CMSI (V) Datamodel





## Constraints on a CMSI -- revisited

- **Common model should treat 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

- Agreement on structure of CMSI (category tree, concept of product families and attributes, ...) has been agreed upon within CMSI-working group
- Specification of CMSI (including XML-format for communicating the model) in preparation

### • Contents of CMSI

- Category tree agreed upon within CMSI-working group (modulo some cleaning up...)
- Process of defining product families and suitable attributes at the very beginning

### • Processes for maintaining CMSI

- Plans of creating a CMSI-server (in conjunction with a server for facilitating co-operation on the basis of the EISPP format) under the auspices of the “Deutscher CERT Verbund”
- Drafts of maintenance processes





## Likely Arguments against CMSI ... ... and how to refute them (I)

- **Argument: “I don't like this or that aspect of the category tree!”**
- **Answer: “You know you have reached a good compromise, if nobody is 100% happy with it”**
- **In other words:**
  - there will always be points that can be argued ad infinitum
  - the alternative is not to have a common model





## Likely Arguments against CMSI ... ... and how to refute them (II)

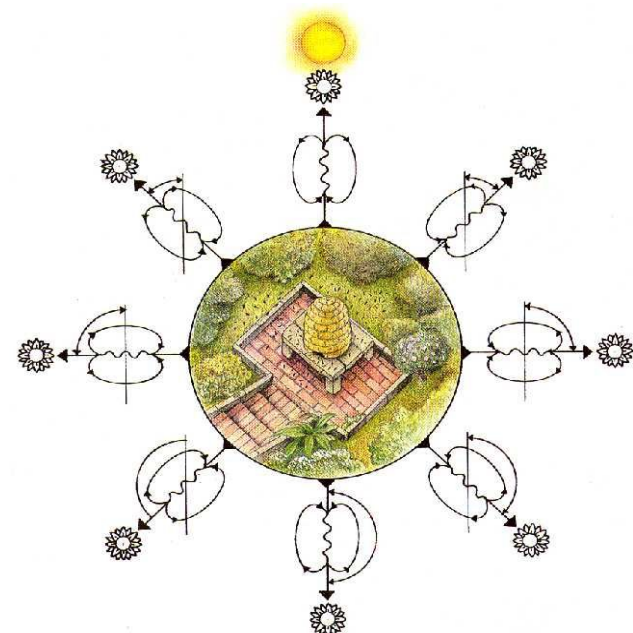
- **Argument: “Maintenance of the model will always be too slow: what happens if I want to send an advisory for a product that is not part of the model, yet?”**
- **Answer: “Striving for absolute perfection is the first step towards failure”**
- **In other words:**
  - Send your advisory anyway, either with coarser information or with information that is not (yet) part of the model.
  - Make a suggestion to the maintainers of the model to include the product”
  - Keep in mind: Even if only 80% of your advisories can be sent with precise system information according to the CMSI, a lot will be gained!



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...**





## Conclusion

- **A working group within the “Deutscher CERT Verbund” is working towards a common model of system information**
  - structure defined
  - content partly defined:
    - category tree almost useable
    - initial set of product families yet to be defined
  - central maintenance of model via dedicated server planned
- **CMSI working group would like to spread use of CMSI within European CERT landscape.**
  - We need your feedback
  - EISPP+CMSI as basis for VEDEF working group?
- **Further information: [bgrobauer@cert.siemens.de](mailto:bgrobauer@cert.siemens.de)**

