EISPP Common Advisory Format Description

Identifier: **EISPP-D3-001-TR**

Version 2.0
Date 2004/05/20

Table of Content

Glossary

BNF	Bacchus Naur Form
CERT	Computer Emergency Response Team
DoS	Denial Of Service
DTD	Document Type Description
EISPP	European Information Security Promotion Program
HTML	Hypertext Markup Language
IDMEF	Intrusion Detection Message Exchange Format
IODEF	Incident Object Description and Exchange Format
PCC	Project Coordination Committee
SME	Small and Medium Enterprise
TBC	To Be Completed
TBD	To Be Defined
URL	Uniform resource location
WP	Workpackage
XML	eXtended Markup Language

Related documents

Applicable Documents

Ref.	*Title*
AD01	CONTRACT No IST-2001-35200 and Annexes
AD02	Project Consortium Agreement
AD03	Annex 1 - Description of Work

Reference Documents

Ref.	*Title*
RD01	EISPP Common Format Description: Value Lists (available on-line on www.eispp.org)

1. EXECUTIVE SUMMARY

The European Information Security Promotion Programme (**EISPP**) strives to set up a network of expertise with the aim of providing European SMEs with those IT Security services that give them the necessary trust in e-commerce to develop their businesses in that direction. EISPP is a project fund by the EU through the fifth European Framework Program within the thematic program Information Society Technologies (IST). Further information about EISPP can be found at its website, http://www.eispp.org/.

Probably the most important security service SMEs have to be provided with, is an advisory service, i.e., the distribution of so-called security advisories that provides system administrators with precise and timely information about new vulnerabilities and what can be done against them. Such information is absolutely essential for IT security, because new vulnerabilities are discovered on a daily basis. IT systems can only be kept secure, if they are regularly upgraded or patched such that the latest security holes are closed again.

This document describes a corner stone of the EISPP approach towards supplying SMEs with security advisories: a common advisory format, which will enable an easy exchange of advisory data between the four CERTs participating in EISPP. The advisory format merges the best-practice information regarding security advisories of these four CERTs.

The format is defined using XML, so the various standards and standard tools of the XML-family can be used for advisory processing. The XML data-type description of this (and future versions) of the format, together with sample XSLT style sheets for displaying advisory data, are made publicly available on EISPP's website http://www.eispp.org.

2. INTRODUCTION

2.1. European Information Security Promotion Programme (EISPP)

Adequate IT security is probably the most important aspect of creating a European environment in which an information society can flourish: Deficits in IT security bring risks to an otherwise desirable expansion of Internet-use by businesses and governments, deter potential home users, and generally endanger what already has become the nerve system of our critical infrastructures. The European Commission therefore has increased the importance of IT security within its new action plan eEurope 2005.

Amongst other measures, the action plan envisions a European warning and information system, which should keep all users of IT infrastructure up-do-date with the latest security issues. The impact of newly discovered vulnerabilities would thus be reduced, massive attacks targeted at such vulnerabilities, e.g., through worm programs, could hopefully be contained before much damage is done.

The initial plans for establishing such a European warning and information system conform to the nature of the European Union : there are no plans for one organization in which all activities regarding IT security are to be centralized. Rather, the European Commission envisions an increased networking between national players such as CERT organizations and similar bodies.

The main objective of EISPP is to set-up a European framework aimed at providing European SMEs with the necessary IT Security services in order to give them the necessary trust in e-commerce, which is important in developing their businesses. EISPP thus is a pioneer regarding the European Commission's vision of forming a European warning and information system on the basis of international networks and cooperations within the European Union. The results of EISPP will therefore be significant for all other attempts for creating networks of expertise in IT-security.

2.2. Workpackage 3: Creating a Common Advisory Exchange Format

New vulnerabilities are discovered on a daily basis. IT systems can only be kept secure, if they are regularly upgraded or patched such that the latest security holes are closed again. System administrators therefore need precise and timely information about new vulnerabilities and what can be done against them. Such information is usually provided in form of "security advisories", issued by vendors for their own products and CERTs for the products that are of interest to each CERT's constituency.

The focus of WP3 (workpackage 3) is to create an infrastructure that enables CERTs to cooperate in the production of advisories. To make the cooperation worthwhile for the member CERTs, WP3 takes care of supplying processes and infrastructure for reuse of work (e.g., it should be possible to import an advisory of another CERT into one’s production system to share the work on menial tasks such as collecting links to patches and references, etc.)

This document describes one key element of that infrastructure : the common format. This advisory format is needed to enable automated exchanges of CERTs’ advisories within the EISPP community. This format is formally defined as an XML DTD (which describes the fields and sections that could exist in an advisory). This document also describes the format in plain language, and gives guidance there, on how fields must be completed.

It must be noted however that the common format does not include a description of how advisories must be presented (i.e. the final layout of an advisory as sent by a CERT to a user). The advisory format is XML and it differs from the format in which advisories are presented to the reader such as HTML or ASCII. WP3 has produced basic translation schemes from XML to HTML and ASCII.

3. AN OVERVIEW OF THE EISPP ADVISORY FORMAT

The table on the following page provides an overview of the possible contents of a security advisory in the EISPP format. The long list of fields may at first seem intimidating, but many fields are optional, so the EISPP format can be tailored to the specific needs of an advisory issuer. The most basic use of the EISPP advisory format would include

- complete identification data
- at least basic *vulnerability classification* (filling in only a few of the fields)
- a single field each for *system information*, *problem description*, and *solution*.

Of course, basic use of the EISPP advisory format along these lines opens only few of the many possibilities for processing advisories that become available when using a more complete feature set. However, at least exchange and processing of the advisory data can be carried out with a standard toolset. A very basic use of the EISPP advisory format may also be useful for organizations that plan to make use of additional features of EISPP when dealing with their old advisories: from most proprietary formats, automated conversion into a basic EISPP format should almost always be possible.

In order to allow more advanced advisory handling especially with respect to advisory interchange between several advisory issuers, additionally the following features should be used:

- **Inclusion of standard vulnerability identifiers**
  Standard vulnerability identifiers such as CVE numbers or Bugtrag IDs allow searching and grouping of advisories by vulnerabilities
- **Complete vulnerability classification**
  Vulnerability analysis (what are the preconditions for exploiting a vulnerability, what are the effects of successful exploitation, how imminent is the threat posed by a given vulnerability, etc.) is carried out by almost every advisory issuer to some extend. The EISPP advisory format provides a common language for exchanging the findings about a vulnerability, which can be used for quality control or even the sharing of workload.
- **Division of the *problem description* into logical subfields**
  By dividing the advisory text into logical subfields such as a description of the technical context, diagnostic information, etc., the re-use of advisory parts becomes easier. Additionally, advisories can be tailored to the audience by supressing those fields that are of little or no interest to a given audience when displaying the advisory.

Additional features such as a revision history, information about the relationship between advisories of the same issuer, a standardized way for providing links to external resources, etc., can be used to improve advisory handling both for the issuer and the reader of advisories.

Field		Description
Identification Data		
Issuer		Advisory Issuer
Reference Number		An advisory reference number
Date		The date on which the advisory was published
Language		Default language of the advisory
Title		The advisory's title
Abstract		A short abstract that complements the information given in the title.
History Data		
Version History		Information about the advisory's current version/revision, along with history information.
Update Information		Information about the relation of the advisory to prior/later advisories of the same issuer
Vulnerability Classification		
Vulnerability Identifiers		A list of standard identifiers such as CVE numbers, Bugtraq IDs, etc. for the vulnerability.
Confidence Level		Information about the confidence the issuer puts into the presented information.
Vulnerability Category		Description of the vulnerability's cause.
Attack Requirements		Technical requirements needed by an attacker to exploit the vulnerability.
Current Impact		Rating of vulnerability's current impact on IT security.
Immediacy		Information about how immediate the threat posed by the vulnerability is, based on:
	Vulnerability Status	Current stage of the vulnerability in the vulnerability life cycle
	Propagation Method	Level of automation that has been achieved for exploitation
Vulnerability Impact		Rating of the severity of the vulnerability's effect
	Vulnerability Effects	Effects that successful exploitation has on the attacked system
Current Impact		The current impact gives a general assessment of the threat posed by the vulnerability.
Risk		Overall assessment of the risk, taking into account also constituency-specific factors.
System Information		
The system information contains information about the affected systems. Typical fields for specifying such information are		
	Affected Platform	Information about platforms affected by the described vulnerability.
	Affected Software	Information about software affected by the described vulnerability.
	Affected System	Combined information about affected platform and software (instead of above two fields)
	Remarks	Additional remarks, e.g., information about systems that may be affected, are not affected, etc.
Description		
The description section of the advisory contains information relevant for understanding the vulnerability. Typical fields are:		
	Publication Context	Information that puts the advisory into context.
	Technical Context	Information that helps the user to understand the technical context of the advisory.
	Description	Description of the vulnerability/vulnerabilities treated by the advisory.
	Technical Info.	Detailed technical information, targeted more at security experts than the average reader.
	Diagnostic	Information to help the reader to determine whether his system is vulnerable.
Solution		
Solution Introduction		General information about possible solutions.
Solution Sections		Each section describes a possible solution. Sections may be divided by solution type (patch, workaround, etc.), affected system, or both.
Additional Resources		
Additional Resources		References to relevant material such as other advisories.

4. EISPP ADVISORY FIELDS — DETAILED DESCRIPTION

The EISPP advisory format is presented by describing all the fields that the format comprises. Many of these fields are optional; please refer to Section 6 guidance about using the EISPP advisory format.

In addition to the present document, which describes in great details the advisory fields and sections, another document has been created that lists the possible values that must be used for some of the EISPP advisory fields. These lists of values are maintained in a separate document because there are subject to frequent updates; that document, named *"EISPP Common Advisory Format Description: Value Lists" [RD01]* is available on the EISPP web site (http://www.eispp.org).

Nota bene: The presentation in this Section focuses on the concepts rather than the precise grammar of how the information is represented. Here, we abstract away from details such as questions whether a piece of data should be represented, e.g., as an XML element or an XML attribute. Please refer to Section 5 for the precise definition of the exchange format.

The field template

Each field is presented using the following template:

(x) Field name

Short field description.

Content type

Information about the type of content within the field. Possible choices are

- Language-independent text, free text, or formatted text
- List-of-values
- Structured content (described with a semi-formal grammar).

Content Description

Information about how the field should be used and, if necessary, detailed description of the content type (in the case of structured content).

Further comments

Questions and thoughts about the field.

The semi-formal grammar

As was mentioned above, the content type may sometimes be described in form of a semi-formal grammar. Here is an example:

```
<telephone_list> ::= <person>*
<person>         ::= <name>.<telephone_nr>+ <email>* [birthday]
<name>           ::= [title] <first_name> <last_name>
<telephone_nr>    ::= language-independent text (a telephone number)
<email>          ::= language-independent text (an email)
<birthday>       ::= yyyy-mm-dd
<title>          ::= Mr | Mrs | Ms | Dr
```

The grammatic description uses the mechanisms of extended BNF. The most prominent features are:

- A (possibly empty) list is indicated with an asterix '*' in post-fix notation
- A non-empty list is indicated with a plus sign '+' in post-fix notation
- Options, i.e., zero or one occurrence, is indicated with square brackets '[…]'; if an option contains a single non-terminal, the pointed brackets that are used to enclose non-terminal names are not written.
- Choice between several options is indicated using the binary operator '|'.

Free text vs. formatted text

Fields whose content type is described as "free text" or "formatted text" have no formal restriction of their content. The difference between free text and formatted text is that formatted text may contain formatting tags that serves for pretty printing the text, providing for different font styles, paragraph styles, lists, etc. (See Section 5 for details.) Language-independent text never allows formatting tags.

Multi-language Feature

With the EISPP advisory format, several language versions of the same advisory can be stored within one file. As a result, all language-independent fields have to be maintained only once; it is only the language-dependent fields for which several versions are supplied. Basically, all entries that are described as containing either *free text* or *formatted text* can be supplied in multiple languages (see the XML description in Section 5.4 for further details.)

The next section describes the fields of the EISPP advisory format.

4.1. Identification Data

Both issuers and recipients of advisories have to manage an ever-growing set of advisories. It is therefore important that a given advisory can be uniquely identified and referenced within a set of (EISPP) advisories. To do so, there is a need for information about the issuing CERT and the reference number of the advisory. The advisory's title is also included into the identification data, because it is the most useful field for readers to recognize an advisory.

(a) Issuer

Issuer of the advisory.

Content type

List-of-values.

Content Description

An identifier for the (EISPP) CERT that issued the advisory. For FIRST members, the identifier should coincide with the short-name of the advisory as given in the FIRST member information (http://www.first.org/team-info/). Please refer to the document *"EISPP Common Advisory Format Description: Value Lists" [RD01]* (available from www.eispp.org) for a complete list of EISPP Certs and their issuer identifiers.

(b) Reference Number

An advisory reference number.

Content type

Language-independent text.

Content Description

Each advisory must have a unique reference number that should not change during the life time of the advisory. The format of this field is defined by the policy of every issuer.

Further comments

Usually, the serial number is a combination of

- the year.
- a serial number.
- an identifier about the information type (e.g., Cert-IST uses the same reference number scheme for several document types, marking advisories with the tag `AV`, while Siemens CERT distinguishes four different lines of advisories (basically Windows, Unix, Network Equipment, and Miscellaneous) within the reference number.

Examples of reference numbers are

- *CERT-IST/AV-2002.217:* The 217th advisory by CERT IST in 2002
- *PC 42/02*: The 42nd advisory dealing with PCs (basically, machines running MS Windows) issued by Siemens CERT in 2002.

(c) Date

The date on which the advisory was published.

Content type

```
<date> ::= yyyy-mm-dd
```

Content Description

Somewhere within the advisory, information about the publication date must be given. The canonical place to do so is within the version history (Field (g)). The version history, however, may not be present, as it is an optional field. In that case, the publication date should be entered into the present field. Either version history or the present field should be present. If both a present, the version history has precedence.

(d) Language

Information about the default language of the advisory.

Content type

Identifier conform to RFC 1766 using ISO 639 two-letter codes (e.g., 'en' for English, 'fr' for French, 'de' for German, 'it' for Italian', 'es' for Spanish, etc.)

Content Description

As mentioned above, all fields within the advisory that are described either as *free text* or *formatted* text can be given in several languages. The top-level language field determines which language is chosen as the default language of the advisory: if no language tag is supplied with a given *free text* or *formatted text* field, the contents are assumed to be of the default language specified in the present field.

(e) Title

The advisory's title.

Content type

Free text (preferably less than 80 characters such that the title fits nicely on one line, e.g., the subject line of an email.)

Content Description

The title of an advisory should tell the reader in one sentence what the advisory is about. It should include information about (1) the affected platform and software, and (2) the vulnerability.

(f) Abstract

A short abstract.

Content type

Free text (not more than one or two sentences.)

Content Description

The abstract should complement the information given in the title; the idea is that by reading the title and the abstract, the reader already has a pretty good idea about the contents of the advisory. This may be useful, for example, when displaying a list of advisories on a web page: giving only the title may be too little information, forcing the reader to click on each advisory.

4.2. History Data

Advisories are not issued into a void—usually there is a history of older advisories, some of which may be complemented or even superseded by the new advisory. Also, advisories may be revised, which means that there is a version history to be maintained.

The EISPP common advisory format provides two fields, one for the version history and one for keeping track of how an advisory relates to other advisories of the same issuer.

(g) Version History

Information about the advisory's current version/revision, along with history information and change dates.

Content type

Non-empty list with entries consisting of three (optionally four) fields:

```
<version_info>      ::= (<version_nr><date><change_descr>
                         [internal_comments])+
<version_nr>        ::= <version>.<revision>
<version>           ::= number
<revision>          ::= number
<date>              ::= yyyy/mm/dd
<change_descr.>     ::= free text
<internal_comments> ::= free text
```

Content Description

Version information is given as a non-empty list of a structured entry consisting of the version number (field `<version_nr>`), the date (field `<date>`), a short description of which changes have been carried out (field `<change_descr.>`), possibly some internal comments of the issuing CERT (e.g., author information).

Note the following important points:

- The list must be ordered: earliest changes are listed after latest changes, such that the latest change is always at the beginning of the list, and the information about the advisory creation at the end of the list.
- The following policy for version numbering must be used:
  - Versions of form `0.x` are for draft advisories that have not yet been released
  - The first public release of an advisory to the readers always has version `1.0`.
  - Minor changes within an advisory that do not lead to a re-release of the advisory only increment the revision (e.g., from `1.0` to `1.1`).
  - Major changes within an advisory that lead to a re-release of the advisory lead to a new version, i.e., the version part is incremented by one and the revision part is set to zero (e.g., from `1.1` to `2.0`).
- If the version history (which is an optional field) is present, no top-level date field (see Field (d)) should be present in order to avoid inconsistencies. If both a version history and the top-level date field are present, then the former has precedence.

Further comments

The following information can be extracted from the version information:

- version number and date of the latest version (extractable from the head of the `version_info` list.)
- date of advisory creation (last element of the `version_info` list)
- date of first public release (extractable from the item in the `version_info` list of version `1.0`.)

(h) Update Information

Information about the relation of the advisory to prior/later advisories of the same issuer.

Content type

Non-empty list of reference numbers and associated tags describing the relationship:

```
<relation_info>    ::= (<relation_tag> <ref_num>)*
<relation_tag>     ::= complements | complemented_by
                     | supersedes | superseded_by
<ref_num>          ::= advisory reference number (see field (b))
```

Content Description

If updates to advisories are given in the form of new advisories rather than modifications to an existing advisory, a reference to the updated advisory is needed. At the same time, advisories that have been updated should be marked as such within the advisory database. Otherwise, when browsing the database, it may be difficult to see whether more recent information regarding a given advisory is available:

complements: An update of an advisory can mean that complementary information is published in a separate advisory—the old and new advisory should be read together. In this case, we say that the new advisory *complements* the older advisory (which itself is *complemented by* the newer one).

supersedes: An update of an advisory can mean that the newer advisory *supersedes* the older one—the older advisory can be completely discarded, as it is *superseded by* the newer one.

Information about complementing and superseding advisories is given as tagged lists of reference numbers: the `<relation_tag>` specifies the relation in which the present advisory stands to the advisories referenced by a reference number (`<ref_num>`) that follows the relation tag.

4.3. Vulnerability Classification

The vulnerability classification helps the reader to quickly assess the nature and danger of the described vulnerability. This section presents the information defined by the EISPP common format to classify the advisories.

Note: In many cases, several vulnerabilities are treated in one advisory. There are two options for using the EISPP format: (1) Several vulnerabilities are treated as if they were a single vulnerability. This means that in the rating, the various categories of information such as effects, impact, preconditions, of all vulnerabilities, etc., are combined. (2) A vulnerability classification is carried out for each vulnerability on its own and several classification sections are used. etc., are combined A single advisory may contain classifications of *several* vulnerabilities. In this case, along with a per-vulnerability rating of *current impact* (Field (o)) and *risk* (Field (p)), an overall-rating *current impact* and *risk* that summarize the whole advisory should be given . Please refer to the XML-definition in Section 5 for more precise information.

(i) Vulnerability Identifiers

If the vulnerability or vulnerabilities treated in the advisory have CVE names or identifiers provided through some other de-facto standard such Bugtraq, these identifiers should be supplied here.

Content Type

```
<vuln_ids> ::= <vuln_id>+

<vuln_id>  ::= <issuer> <ref_num>

<issuer>   ::= identifier of the issuer of the vulnerability identifier

<ref_num>  ::= reference number associated with the resource pointed to by the reference
```

Content Description

The `<issuer>` is set to an identifier for the issuer of the standard vulnarbility id. Standardized identifiers for naming systems should be used; please refer to the document *"EISPP Common Advisory Format Description: Value Lists"* [RD01] (available from www.eispp.org) for a complete list of standardized identifiers. Below we provide some examples:

Identifier	Vuln. ID. Naming System
BID	Security Focus Bugtraq ID database entry
CERT-VN	CERT/CC Vulnerability Note
CVE	CVE Vulnerability Identifier
SUNBUG	Sun Bug ID

The `<ref_num>` is set to the given vulnerability identifier (please refer to the document *"EISPP Common Advisory Format Description: Value Lists"* for information regarding the standard reference-number format for the various naming systems.)

(j) Confidence level

A rating of the reliability of the vulnerability classification.

Content type

A list-of-values with an additional, optional free-text field for explaining the rating:

```
<confidence level> ::=  <rating> [explanation]

<rating>           ::=  official_and_tested | official | tested
                      | probable | not_qualified | not_rated

<explanation>      ::= free text
```

Content Description

The confidence level is set according to the following criteria:

- **Official and tested vulnerability:** The vulnerability has been announced by a recognized authority (CERT, CIAC, etc.) or by a vendor. It was also successfully tested by the issuer or somebody trusted by the issuer (e.g., another EISPP CERT).
- **Official vulnerability:** The vulnerability has been announced by an official authority (CERT, CIAC, etc.) or by a vendor.
- **Tested vulnerability:** The vulnerability has not been announced by an official authority or a vendor, but it was successfully tested by the issuer or somebody trusted by the issuer (e.g., another EISPP CERT).
- **Probable vulnerability:** :The vulnerability has not been announced by an official authority or vendor, but is highly probable (cross-checked between several information sources).
- **Not qualified vulnerability:** The vulnerability has not been released by an official authority or a vendor, and could neither be tested nor crosschecked, but its criticality justifies an advisory, which must be taken "with caution".

The optional free-text field can be used to explain the rating to the reader.

(k) Vulnerability Category

A brief description of the vulnerability' s cause.

Content type

Free Text.

Content Description

The vulnerability category informs about the cause of an vulnerability. Frequent examples are *buffer overflow*, *cross-site scripting*, etc.

(l) Attack Requirements and Attack Vector

Technical requirements needed by an attacker to exploit the vulnerability.

Content type

A list-of-values for specifying the attack requirements. More explanation regarding possible attack vectors can be given in an optional free-text field.

```
<requirements> ::=  <type> [explanation]

<type>         ::=   remote_no_account
                   | remote_account (user | service)?
                   | victim_interaction (content | contact)?
                   | local (interactive | physical)?
                   | packet_access (sniff | manipulate)?
                   | other
                   | not_rated

<explanation> ::= free text
```

Content Description

The attack requirements are rated according to the following criteria:

- **`remote_no_account`:** The system can be attacked remotely, i.e., via a routable protocol, without requiring either an account (or some other form of authentication) or any victim interaction such as opening some file, accessing some service, etc. (see below).
- **`remote_account`:** The system can be attacked remotely. No victim interaction is required (see below), but the attacker needs an account or has to be authenticated in some other way. We may distinguish between a regular account as some **user** of the attacked system or an account/successful authentication for some **service**, e.g., for a data base server, a mail server, etc.
- **`victim_interaction`:** An attack may be carried out remotely, but is not possible without some form of victim interaction, e.g., a user has to download and open some file, access some service, etc. We may distinguish between cases where the victim must in some form access **content,** which theoretically could be delivered both via a file and a service, and cases where it is enough for the victim to contact some **service** to be compromised.

- **`local:`** The attacker needs local access to the victim. We may distinguish between attacks that require an **interactive** login and attacks that are based on **physical** manipulation of the victim.
- **`packet_access:`** The attacker needs access to packets not addressed to his machine. We may distinguish between situations where it is enough for the attacker to **sniff** packets vs. situations where the attacker needs to **manipulate** packets in some way.
- **`not_rated:`** The issuer chose not to (or was not in a position to) rate the attack requirements in this advisory.

Further comments

The field "Attack Requirements and Attack Vector" allows a choice regarding the level of granularity with which information is specified. For example, the fact that a *remote* attack is possible can be augmented with information about the necessity of an account, but must not. Similarly, the information about what kind of account might be necessary can be left away.

(m) Vulnerability Status, Propagation Method and Immediacy

From information about the vulnerability's status in the vulnerability life cycle and the propagation method, a rating of how immediate a threat the vulnerability poses is derived.

Content type

```
<immediacy> ::= [vuln_status] [prop_method] [explanation]
                [rating]

<vuln_status> ::=  theoretical
                 | exploitable
                 | currently_exploited
                 | exploit_published
                 | not_rated

<prop_method> ::=  manual
                 | automated
                 | replicating
                 | not_rated

<rating>      ::= very_high | high | medium | low | very_low | not_rated

<explanation> ::= free text
```

Content Description

The propagation method describes the degree of automation with which a vulnerability is exploited:

- **manual:** Exploiting the vulnerability requires steps that either have not been automated yet or cannot be automated
- **automatic**: Exploiting the vulnerability can be automated or has been automated such that "push button" exploits are possible
- **replicating**: The vulnerability either is exploited by self-replicating code or is a likely candidate for exploitation by self-replicting code.

The definition of the vulnerability status is influenced by the rating of the propagation method, if such a rating can be given:

- **theoretical**: There are indications that a vulnerability could exist, but this has not been established beyond doubt. If the advisory author can make an educated guess about what level of automation would be likely – in case the vulnerability indeed exists – the propagation-method field can be set accordingly.
- **exploitable**: It has been established (either by the issuer or somebody else) that the vulnerability exists and therefore is exploitable. Such information can be gained theoretically (e.g., through code analysis) or practically (e.g., by creating a proof-of-concept exploit). If there are indications as to what level of automation will be achieved for exploiting the vulnerability, the propagation-method field can be set accordingly.

- **currently exploited**: There are indications that the vulnerability may be actively exploited, e.g., because actual exploit attempts have been witnessed or exploits are suspected to circulate within the black-hat community. If there are indications as to the level of automation with which the exploits are carried out, the propagation-method field can be set accordingly.
- **exploit published**: Exploits for the vulnerability are publicly available, e.g., through full disclosure lists. The propagation-method field should be set to the level of automation of the published exploit(s) or – if it is foreseeable that better exploits are likely to follow shortly – to the expected maximum level of automation.
  Remark: An exploit does not necessarily have to be a piece of code. For vulnerabilities that are so simple to exploit that no coding is necessary, already the information about how the vulnerability can be exploited counts as exploit.

The following table gives an overview over all possible combinations and suggests an rating of the *immediacy* of the threat posed by the vulnerability. If nothing can be said about the propagation method, then 'automated' should be assumed for calculating the immediacy.

Immediacy	<propagation method>		
vuln. status	manual	automated	replicating
theoretical	very low	low	medium
exploitable	low	medium	high
currently exploited	medium	high	high
exploit published	medium	high	very high

Examples

- The RPC DCOM vulnerability of summer 2003 shows, how the immediacy of a vulnerability changes with time, as events progress:
  - The vulnerability became known through a vendor announcement, according to which the vulnerability was indeed *exploitable*. Right from the beginning, experts warned, that the vulnerability was a likely candidate for exploitation by a worm program, so the propagation method would have been set to *replicating*. Thus, immediately, the RPC DCOM vulnerability had an immediacy rating of *high.*
  - A few days after the vulnerability became public, exploits were published, so the vulnerability status changed to *exploit published*. The first exploits were not very well designed and required a bit of tinkering, but it was clear that at least working automated exploits would follow. Thus the propagation method should have been set at least to *high* or – taking into account that the threat of a worm was still imminent – to *replicating*. Thus, at this stage, the immediacy would be judged as *high* or *very high*.
  - At the latest with the appearance of the Blaster worm, a *replicating* exploit was publicly available – after all, it took only a few minutes to catch a specimen of the Blaster worm by connecting a vulnerable computer to the Internet. Obviously, with the Blaster outbreak, the immediacy of the RPC DCOM vulnerability was *very high*.
- Consider a CGI-script vulnerability that allows an attacker to view information not intended for the public by supplying the CGI-script with maliciously crafted arguments. In this case, the information about how to exploit the vulnerability is equivalent to an exploit: as soon as the vulnerability is widely known, the vulnerability status is *exploit published*. The distinction between *manual* and *automated* depends on the level of "customization" with respect to the attacked web server that is necessary. For example, if a generic argument can be given that in many cases will reliably retrieve a password file, then the propagation method probably should be rated as *automated*. If, on the other hand, some tinkering with the argument strings is necessary to make the exploit work for a given web site, then the propagation method is *manual*.

(n) Vulnerability Effect and Impact

Information about the effect on a targeted system if exploitation succeeds: specified are the type of loss of security and the scope in which a violation can occur.

Content type

```
<impact>          ::= [effects] [rating] [explanation]

<effects>         ::= <effect>+

<effect>          ::= <loss> <scope>?

<loss>            ::=  take_control
                     | take_partial_control
                     | modification
                     | disclosure
                     | availability
                     | circumvention
                     | not_rated

<scope>           ::=  person
                     | service
                     | system
                     | network
                     | not_rated

<rating>          ::=  very_high | high | medium | low | very_low
                     | not_rated

<explanation>     ::= free text
```

Content Description

The main information regarding the effect is with respect to the kind of security loss that may be suffered:

- **Take control:** By exploiting the vulnerability, the attacker can gain total control of the attacked system within the specified scope:
  - **Person:** The attacker can take control over a user account (where 'user' is to be understood as a 'real' person rather than a service for which a dedicated user account has been configured).
  - **Service:** The attacker can take control over a service. For example, on a Unix machine, the dedicated user account for the web server demon may be compromised or, on a Windows machine, the web-server process can be controlled.
  - **System:** The attacker can control the complete system, i.e., the adminstrator/root account can be compromised
  - **Network:** The attacker can gain control over some part of a network: all packets can be read, modified, re-routed, etc.
- **Take partial control:** The attacker can gain control over some aspects but falls short of total control. (If the partial control is limited to writing or reading data, then the loss types *modification* and *disclosure* should be used instead).
  - **Person:** The attacker can act in place of a user (where 'user' is to be understood as a 'real' person), e.g., accessing certain services as the user.
  - **Service:** The attacker can partially control a service. For example, the compromise of a mail service might enable the attacker not only to read and modify mails of all users, but to send mails as any user, etc.
  - **System:** The attacker can partially control the complete system, i.e., carry out certain actions with administrative rights.
  - **Network:** The attacker can gain partial control over some part of a network
- **Modification:** By exploiting the vulnerability, the integrity of data can be violated. If such modification leads directly to (partial) control within some scope of the system, *take (partial) control* should rather be used as loss type, together with the appropriate scope. Similarly, if modification allows the attacker the *circumvention of security measures* within some scope (e.g., the deletion of log files), *circumvention* should be used as loss type.
  - **Person:** The attacker can modify user data (where 'user' is to be understood as a 'real' person), e.g., writing user files.
  - **Service:** The attacker can modify data under the control of some service such as a database, a web server, etc.

  - **System:** The attacker can modify system data.
  - **Network:** The attacker can modify packets within some part of a network
- **Disclosure:** By exploiting the vulnerability, the confidentiality of data can be violated.
  - **Person:** The attacker can read user data (where 'user' is to be understood as a 'real' person), e.g., read user files, read a user's email, etc.
  - **Service:** The attacker can read data under the control of some service such as a database, a web server, etc.
  - **System:** The attacker can read system data.
  - **Network:** The attacker can read packets within some part of a network
- **Availability:** By exploiting the vulnerability, an attacker can negatively affect the availability of some resource.
  - **Person:** The attacker can lock out a user (where 'user' is to be understood as a 'real' person) from certain resources such as a service, the whole system, etc.
  - **Service:** The attacker can impair the availability of some service, e.g., crash the web server, etc.
  - **System:** The attacker can slow down, crash, or otherwise impair the availability of a complete system.
  - **Network:** The attacker can impair the availability of some network part.
- **Circumvention of Security Measures** By exploiting the vulnerability, the some security protection can be circumvented. Typical examples are the deletion/modification of log files, the removal/penetration of a (personal) firewall, etc. If, however, the circumvention leads directly to one of the loss types outlined above, then that loss type should be used instead.
  - **Person:** The attacker can circumvent a security measure applied within a user's context (where 'user' is to be understood as a 'real' person), e.g., a personal fire wall.
  - **Service:** The attacker can circumvent security measures applied for a certain service, e.g., modify web-server log files.
  - **System:** The attacker can circumvent a security measure applied for a whole system, e.g., modify system log files, disable an anti-virus product installed for the whole system, etc.
  - **Network:** The attacker can circumvent a security measure on the network level, e.g., penetrate a firewall product.

The following table gives an overview over all possible combinations and *suggests* a rating of the *impact* each effect has on IT security.
Remark: *The rating is only a suggestion that may have to be adjusted to the actual situation. For example, if modification of almost irrelevant data is possible (e.g., changing the default language for an application), then the impact is probably lower than the table suggests. On the other hand, if disclosure affects data that is very likely to be extremely sensitive (e.g., the contents of a program for managing logins and passwords), then the impact is probably higher than suggested in the table. Similarly, the impact concerning the scope service is very much dependent on both the nature of a service and the privileges granted to the service process/account.*

Impact	<scope>			
<loss>	person	service	system	network
Take Control	high	high	very high	very high
Take Partial Control	medium	medium	high	high
Modification	low	medium	high	high
Disclosure	very low	low	medium	high
Availability	very low	low	medium	high
Circumvention of security measures	very low	low	medium	high

(o) Current Impact

The current impact gives a general assessment of the threat posed by the vulnerability. It is rated by combining the immediacy of the vulnerability and its impact.

Content type

List-of-values:

```
<current_impact> ::=  very_high | high | medium | low | very_low
                    | not_rated
```

Content Description

The following table combines immediacy and impact into the current impact:

Current Impact	Impact				
Immediacy	very low	low	medium	high	very high
very low	very low	very low	low	low	medium
low	very low	low	low	medium	high
medium	low	low	medium	high	high
high	low	medium	high	high	very high
very high	medium	high	high	very high	very high

(p) Risk

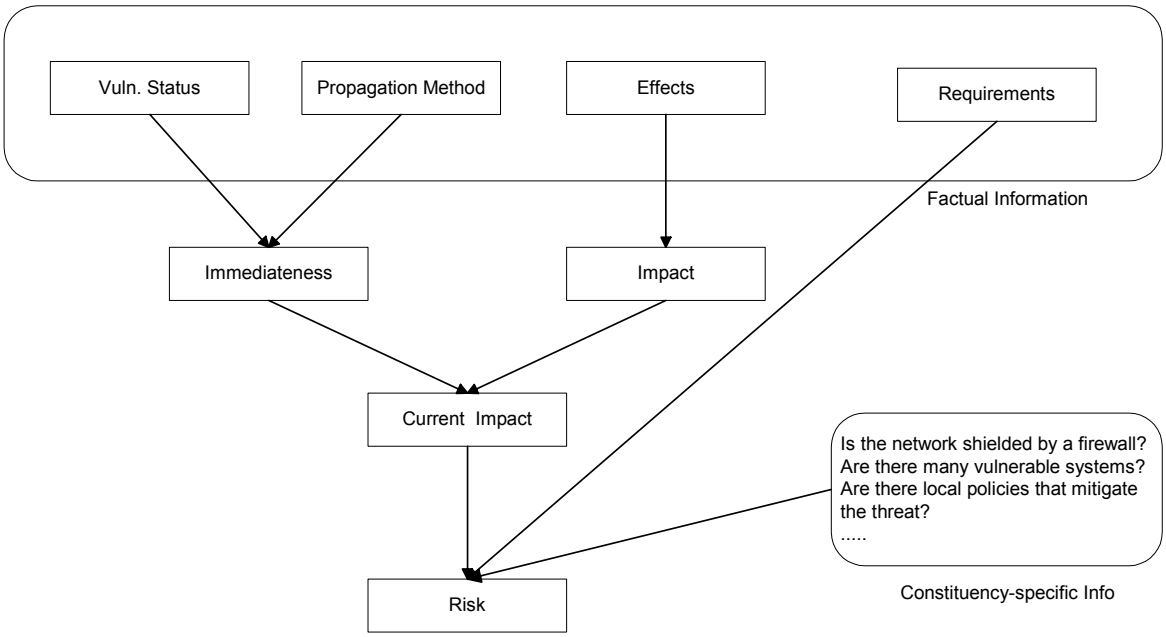
The risk assessment for a vulnerability combines the factual assessment of the current impact of a vulnerability with constituency specific considerations.

Content type

```
<risk_ratings>  ::=  <risk>+

<risk>          ::= [schema] [group] <rating>

<schema>        ::= tag for identifying a rating schema for risk that has been used

<group>         ::= tag  for identifying a group defined within the schema for which the
                    risk has been rated

<rating> ::= very_high | high | medium | low | very_low | not_rated
```

Content Description

Risk assessment requires some idea about the probability that one's assets are successfully attacked. Therefore, risk assessment can only be carried out on the basis of constituency-specific knowledge. Information about the current impact of a vulnerability and attack requirements may be useful for the risk assessment as the following diagram shows, but ultimately, the process of risk assessment can only be defined locally.



There may, however, be attempts to define risk rating schemas for 'standard' constituencies. To support such initiatives, several risk ratings can be given, which can be tagged with an identifier that specifies the rating schema that has been used and a risk group (e.g., home users, firewalled servers, etc.) for which the rating has been carried out.

4.4. System Information

The crucial question for assessing whether an advisory may be relevant for a given environment depends on the system affected by the vulnerability described in the advisory.

(q) System Information

Information about affected platforms and systems.

Content type

A structured field, which gives both informal (i.e., formatted text) and formal information about the affected systems.

```
<system_info>        ::= (<info_type> <information>)+
                         <system_id_list>?

<content_type>       ::=  affected_platform
                        | affected_software
                        | affected_system
                        | remarks

<information>        ::= formatted text

<system_list>        ::= machine-readable system information. See Section 5 for details
```

Content description

More than for any of the other fields, system information is important for advisory distribution: if system information is kept in a machine-readable format, then filtering mechanisms can be used to distribute security advisories only to readers with systems that might actually be affected. On the other hand, machine-readable system information is probably unsuitable for displaying it directly within the advisory, which will be read by humans after all.
The EISPP advisory format tries to strike a balance by providing (1) formatted text fields to inform the reader about which systems are affected, and (2) a field for machine-readable information about affected systems.

System information is usually provided by informing about the affected platform platform (either an operating system,e.g., SuSE Linux, or MS Windows XP, a list of operating systems, a family of operating systems or hardware, etc.) and the affected software (e.g., MS Excel, Apache, etc.). Whether this is done using a single field or two separate fields–one for the platform, one for the software–is up to the advisory issuer:

- For specifying the affected platform and the affected software in two separate fields, two `<information>` fields should be used that are tagged as `'affected_platform'` and as '`affected_software'`, respectively.
- For specifying the affected platform and the affected software in one single field, one `<information>` fields should be used that is tagged as `'affected_system'`.

Additional information, e.g., about systems that may be affected or are known to be not affected can be given in a `<information>` field that has been tagged as `remarks`.
Note: The choice to use a single field or two fields for specifying system information should be made consistently for all issued advisories.

For specifying system information in a machine-readable way, the field `<system_list>` can be used. For machine readable information, however, a well-defined, formal model of system information is necessary. Because the EISPP format could be used together with several such models, the definition of `<system_list>` is held rather general; please see Section 5 for details.

4.5. Description

Vulnerability classification and system information enable the reader to quickly assess whether (1) the advisory might be relevant in his environment and (2) how quickly he should react.

(r) Description

Information relevant for understanding the vulnerabilities treated in the advisory.

Content type

A structured field, which gives both informal (i.e., formatted text) and formal information about the affected systems.

```
<descriptions>        ::= <description>+

<description>         ::= <content_type> <information>

<content_type>       ::=  publication_context
                          | technical_context
                          | description
                          | technical_information
                          | diagnostic
                          | complete_advisory

<information>         ::= formatted text
```

Content description

There are several aspects in informing about a vulnerability: providing information about the technical context of a vulnerability, which may be of more interest to inexperienced users, technical information for security experts, information about how to diagnose whether one's system is really affected, etc. If desired, these different aspects can be treated in separate fields, where each field is tagged with a `<content_type>` that informs about the contents of the field. The following content categories have been identified:

- **publication_context:** Examples for possible entries regarding the publication context are
  - information as to what triggered the release of the present advisory (e.g., the release of a patch by a vendor)
  - information about the relation of the present advisory to former advisories of the same issuer (e.g., for an advisory that complements an older one, what the new bit of information is).
- **technical_context:** Examples for possible entries regarding the publication context are
  - information as to what triggered the release of the present advisory (e.g., the release of a patch by a vendor)
  - information about the relation of the present advisory to former advisories of the same issuer (e.g., for an advisory that complements an older one, what the new bit of information is).
- **description:** This description should be understandable by any readers and does not require extended knowledge in IT and security. More technical description should be put in a "Technical information" field.
- **technical_information:** Detailed technical information, targeted more at security experts than the average reader
- **diagnostic:** Information to help the reader with diagnostics, i.e., to determine whether his system has the described vulnerability.
- **complete_advisory:** For migrating from a proprietary advisory format to EISPP, it may be helpful to be able to start with very simple EISPP advisories. In the most simplified case, the complete advisory could be dumped into a single description field of category *complete_advisory*. See also Section 6.

4.6. Solution

After the reader has understood the vulnerability and established which of his systems are affected, the question is how the vulnerability can be removed or at least alleviated.

(s) Problem Solution

Description of how the vulnerability can be removed/alleviated.

Content Type

List of structured solution fields:

```
<solutions>   ::=  [sol_intro] <sol_section>*

<sol_section> ::=  [sol_type] <sol_title> [sol_descr] <reference>*

<sol_type>    ::=  code_fix (patch | software_upgrade)?
                 | workaround
                 | other

<sol_intro>   ::= formatted text

<sol_title>   ::= free text

<sol_descr>   ::= formatted text
```

The `<reference>` field is used also in other sections; please refer to Section 4.8 for a detailed description.

Content Description

The contents of the solution section following the semi-formal grammar given above is explained

`<solutions>` An advisory may present several solutions, e.g., patch information and work arounds. If several solutions are presented, it may be desirable to start the solutions section with some introductory information (optional field `<sol_intro>`), followed by a list of solution entries.

`<sol_section>` a solution has a short solution title (field `<sol_title>`), followed by a description of the solution (field `<sol_descr>`). After the description, a list of references can be given. The solution itself can be classified using the field `<sol_type>`—a distinction is made between solutions by patches and software upgrades (summarized as code_`fix`), and workarounds.

`<reference>` A reference provides a pointer to some resource "outside" the advisory such as patches, etc. A reference can be given a (preferably short) title (field `<ref_title>`); the pointer itself is given as one or more URIs (field `<uri>`); all URIs should point to the same resource—this provides the possibility to point, for example, to local copies of the resource or different language versions of the resource (e.g., for advisories). Further information regarding size, checksum, etc. can also be given; this may be most useful when referencing patches. See Section 4.8 for a detailed description.

4.7. Vulnerability Identifiers and Additional Resources

The EISPP participants tend to issue short and concise advisories, hence additional references to additional sources of information will often be useful. Standardised identifiers for vulnerabilities such as CVE names do not provide additional information themselves (even though automatic links could be created), but help the reader to correlate vulnerability information from sources such as local or remote scanners with a collection of advisories.

(t) Additional Resources

References to relevant material such as advisories.

Content Type

We use the `<reference>` structure (described in Section 4.8) and provide pointers to additional resources as a list of reference:

```
<additional_resources > ::= <reference>*
```

Content Description

The main use of the *additional references* field is to provide references to relevant material such as advisories published by other bodies. See Section 4.8 for further details about the `<reference>` structure.

4.8. Description of the reference-structure

4.8.1. Format of reference structure

A reference provides a pointer to some resource "outside" the advisory such as other advisories, patches, etc. A reference can be given a (preferably short) title (field `<ref_title>`); the pointer itself is given as one or more URIs (field `<uri>`); all URIs should point to the same resource—this provides the possibility to point, for example, to local copies of the resource or different language versions of the resource (e.g., for advisories). Further information regarding size, checksum, etc. can also be given; this may be most useful when referencing patches. The reference structure described is used in Fields (s) and (t).

```
<reference>     ::= [ref_type] [ref_name] [issuer] [ref_num]
                    [ref_title] <uri>*

<ref_type>      ::=  code_fix (patch | software_upgrade)?
                   | advisory [vendor]?
                   | technical_information
                   | vuln_id
                   | other

<ref_name>      ::= identifier of this reference local to the present advisory, which may be used to refer
                    to this reference from a formatted-text field via an html <a>-tag.

<issuer>        ::= identifier of the issuer of the resource pointed to by the reference

<ref_num>       ::= reference number associated with the resource pointed to by the reference

<ref_title>     ::= free text

<uri>           ::= [size] [checksum] [checksum_alg] [language] standard URL

<size>          ::= size of the resource pointed to in bytes

<checksum>      ::= checksum of the resource pointed to

<checksum_alg>  ::= identifier for the checksum algorithm used to calculate the checksum

<language>      ::= identifier for the language of the resource (e..g., an advisory) pointed to.
```

`<ref_type>`: The type of the resource pointed to by the reference:

- A `fix` is a piece of code that fixes the vulnerability. If of interest, a distinction between a `patch`, and a `software_upgrade` can be made. The difference is, that a patch does not affect the version number of the patched software, while an software upgrade does.
- For an `advisory`, the special case of a vendor advisory may be closer specified
- The `vuln_id` is described in Field (i).

`<ref_name>`: As will be shown below when describing the XML format, there is a possibility to refer to a reference from within a formatted-text field using the HTML-tag `<a>`. The `<ref_name>` serves as unique identifier for a reference within the advisory.

`<issuer>`: An identifier of the source that provides the resource.the `<issuer>`. Standardized identifiers for naming systems should be used; please refer to the document *"EISPP Common Advisory Format Description: Value Lists"* [RD01] (available from www.eispp.org) for a complete list of standardized identifiers. Below we provide some examples:

Identifier	Source
`AUSCERT`	AUSCERT advisory
`CERT`	CERT/CC Advisories
`CIAC`	DOE CIAC (Computer Incident Advisory Center) Bulletins
`CISCO`	Cisco security advisory

DEBIAN	Debian Linux Security Information
EEYE	eEye security advisory
FREEBSD	FreeBSD security advisory
ISS	ISS Security Advisory
MANDRAKE	Linux Mandrake advisory
MS	Microsoft Security Bulletin
REDHAT	Security advisories
SGI	SGI Security Advisory
SUN	Sun Security Bulletin
SUNALERT	Sun Security Allert
SUSE	SuSE Linux: Security Announcements
XF	X-Force Vulnerability Database

`<ref_num>`: The reference number (if any) given to the resource by its issuer (e.g., for advisories of other references, the reference number of that advisory, or, for patches from Microsoft, the patch number.) Please refer to the document *"EISPP Common Advisory Format Description: Value Lists" [RD01]* for information regarding the standard reference-number format for various resources.

`<uri>`: This is the location where the resource can be found. The `<uri>` can be omitted in the <reference> structure only when it can be derived easily from the `<issuer>` and `<ref_num>` information. URIs that are explicitly given have preference over derived URIs.

`<size>`: Information about the size of the resource that is pointed to (in bytes).

`<checksum>`: The checksum of the resource that is pointed to (as output by the checksum algorithm that was used).

`<checksum_alg>`: An identifier for the checksum algorithm that was used to calculate the checksum of the resource that is pointed to.

`<language>`: An identifier for the language of the resource that is pointed to (e.g., for advisories); language identifiers are given using ISO 639 two-letter codes.

5. XML FORMAT

5.1. Introduction

The advisory format, as presented in the previous section, has been translated into an XML DTD. In most cases, the translation from the semi-formal grammar given above into the XML DTD is straightforward; points that required special consideration were

- where to use XML elements and where to use XML attributes
- how to implement the multi-language feature
- how to treat lists-of-values (for XML-attributes, lists-of-values can be constrained via the DTD)
- which HTML-tags can be used within the *FormattedText* elements

The DTD is extensively commented; we hope that together with the description given in Section 4, it can be easily understood.

5.2. Notes on Presentation

In order to make use of an advisory written in the EISPP XML format, display mechanisms need to be implemented that convert the XML format into a readable presentation, e.g., in ASCII or HTML. A possible implementation mechanism is XSLT, a transformation language especially designed for XML.

In order to make full use of the EISPP format, a presentation mechanism should be able to do the following with an XML advisory:

- extract advisories for all languages for which information is contained in the XML advisory
- convert standard values out of fixed lists of values into human-readable form, e.g., turning impact "`remote_no_account`" (see Field (l)) into "*The vulnerability can be exploited remotely; no account on the system is neccessary for exploitation*." for the English advisory version, a corresponding German sentence for the German version, etc.
- use information contained in the vulnerability identifiers (Field (i)) and the reference structure such as the 'issuer' and 'ref_num' information to automatically derive a reference title and uri. For example, a reference from issuer `CVS` with reference number `CAN-2002-0008` could be displayed *automatically* as
  "*CVE Number CAN-2002-0008 (http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0008)"*
- update information (field (h)) should be used to display warnings in outdated advisories and information about outdated advisories in the the more recent advisory. In both cases, links to the referenced advisories could be generated.

5.3. XML-signature

So far, security concerns regarding mainly authenticity of data contained in an EISPP XML-advisory must be treated by securing the *transport* of the advisory, e.g., by signing the email with which the advisory was sent. An alternative would be to include security features into the XML-document, using for example XML-signature schemes, so that proof for the authenticity of the advisory data can be stored along with the data. Research and experiments indicated that for the moment, XML signature schemes are not mature enough to be included into the EISPP standard.

5.4. XML DTD

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!--====================================================================-->
<!--======EISPP Common Advisory Format, version 2.0        ================-->
<!--====================================================================-->


<!-- ======== First, the type definition for open lists of values =============-->
<!-- Please refer to the comment at the end of this DTD for information
     regarding the currently supported open lists of values -->
```

```
<!ENTITY % oLOV "NMTOKEN">


<!-- ======== Next, we specify rules for types not expressible with LOVs =======-->

<!-- The generic type for attributes that are defined using DTD-external rules
     is called 'token' -->

<!ENTITY % token "CDATA">


<!-- A language code, as per [RFC1766] -->

<!ENTITY % LanguageCode "CDATA">


<!-- Date information, with format yyyy-mm-dd -->

<!ENTITY % Date "CDATA">


<!--==========Finally, definitions of lists of values ============================-->


<!--Values for generic ratings from 'very low' to 'very high'  -->
<!ENTITY % attvals.rating_attr "
   (not_rated | very_low | low | medium | high | very_high)
 ">


<!--Values for confidence level rating  -->
<!ENTITY % attvals.confidence_rating_attr "
   (not_rated | not_qualified | probable | tested | official | official_and_tested)
 ">

<!--Values for requirements type  -->
<!ENTITY % attvals.requirements_type_attr "
   (not_rated | remote_no_account | remote_account | victim_interaction | local |
packet_access | other)
 ">

<!--Values for requirements subtype  -->
<!ENTITY % attvals.requirements_subtype_attr "
   (user | service | content | contact | interactive | physical | sniff | manipulate )
 ">


<!--Values for vuln status   -->
<!ENTITY % attvals.vuln_status_attr "
   (not_rated | theoretical | exploitable | currently_exploited | exploit_published)
 ">

<!--Values for propagation method   -->
<!ENTITY % attvals.vuln_status_propagation_attr "
   (not_rated | manual | automated | replicating)
 ">


<!--Values for effect   -->
<!ENTITY % attvals.loss_attr "
   (not_rated | take_control | take_partial_control | modification | disclosure |
availability | circumvention)
 ">
```

```
<!--Values for effect scope   -->
<!ENTITY % attvals.scope_attr "
   (not_rated | person | service | system | network)">


<!--Values for the relate attribute within the relation information    -->
<!ENTITY % attvals.relation_attr "
   ( complements | complemented_by | supersedes | superseded_by)
 ">

<!--Values for the sol_type attribute within the sol_sec    -->
<!ENTITY % attvals.sol_type_attr "
   (code_fix | workaround | other)
 ">

<!--Values for the sol_subtype attribute within the sol_sec    -->
<!ENTITY % attvals.sol_subtype_attr "
   (patch | software_upgrade | other)
 ">


<!--Values for the ref_type attribute within the reference information    -->
<!ENTITY % attvals.ref_type_attr "
   (  code_fix  | advisory | technical_information | vuln_id | other )
 ">

<!--Values for the ref_subtype attribute within the reference information    -->
<!ENTITY % attvals.ref_subtype_attr "
   (  patch | software_upgrade | vendor )
 ">

<!--Values for the type attribute within the <a>-element     -->
<!ENTITY % attvals.a_type__attr "
   ( vulnerability | reference )
 ">



<!--============================================================================-->
<!--==================Free text and formatted text==========================-->
<!--============================================================================-->
<!--
We define two kinds of elements for storing language-dependent content:
"FreeText" for text without markup, and "FormattedText" for text with markup.

Language-dependent fields such as "Title" can have several FreeText or
FormattedText-entities as content, one for each language.

-->

<!ELEMENT FreeText (#PCDATA)>
<!ATTLIST FreeText
        xml:lang            %LanguageCode;          #IMPLIED
>


<!--
Before defining what is formatted text, we need to make some
definitions: We want to be able to use a few very basic html-elements
in formatted text, namely:

 - font changes (emphasis, strong emphasis, code font)

 - linebreaks

 - paragraphs
```

```
 - lists (unnumbered, numbered, and definition lists)

 - tables (very simple ones)

 - anchors and links

-->


<!--
 First we define some categories of markup:
 -->


<!ENTITY % heading "h1 | h2 | h3 | h4 | h5 | h6">

<!ENTITY % phrase "em | strong | code"> <!-- Font changes -->

<!ENTITY % inline "%phrase; | br | a"> <!-- Stuff within text: font change, line break
and links -->

<!ENTITY % Inline "(#PCDATA | %inline;)*"> <!-- normal text -->

<!ENTITY % lists "ul | ol | dl"> <!-- unnumbered, numbered, and definition lists -->

<!ENTITY % block "p | %heading; | %lists; | table | pre"> <!-- text blocks:
paragraphs,
                                        preformatted text, lists, and tables -->



<!ENTITY % ListBody "(#PCDATA | %lists; | %inline; | p | table | pre)*">

<!-- Now we are in a position to define formatted text: -->

<!ELEMENT FormattedText (#PCDATA | %inline; | %block;)*>
<!ATTLIST FormattedText
        xml:lang            %LanguageCode;          #IMPLIED
>

<!-- What follows are the definitions for lists, tables, etc. -->

<!ELEMENT table (tbody)>
<!ELEMENT tbody (tr)+>
<!ELEMENT tr (th|td)+>
<!ELEMENT th %Inline;>
<!ELEMENT td %Inline;>


<!-- horizontal alignment attributes for cell contents -->
<!ENTITY % cellhalign
  "align      (left|center|right) #IMPLIED"
  >

<!-- vertical alignment attributes for cell contents -->
<!ENTITY % cellvalign
  "valign     (top|middle|bottom|baseline) #IMPLIED"
  >

<!ATTLIST tbody
  %cellhalign;
  %cellvalign;
  >
```

```
<!ATTLIST tr
  %cellhalign;
  %cellvalign;
  >


<!-- th is for headers, td for data and for cells acting as both -->

<!ATTLIST th
  rowspan     CDATA       "1"
  colspan     CDATA       "1"
  %cellhalign;
  %cellvalign;
  >

<!ATTLIST td
  rowspan     CDATA       "1"
  colspan     CDATA       "1"
  %cellhalign;
  %cellvalign;
  >




<!ELEMENT em %Inline;> <!-- emphasis -->

<!ELEMENT strong %Inline;> <!-- strong emphasis -->

<!ELEMENT code %Inline;> <!-- program code -->

<!ELEMENT br EMPTY>


<!-- Preformatted text is always understood as code, ie., it
     should be typeset with a fixed-width font. We allow no
     markup whatsoever within "pre" -->

<!ELEMENT pre (#PCDATA)>

<!-- In a heading we allow only font changes -->

<!ELEMENT h1  (#PCDATA | %phrase;)*>

<!ELEMENT h2  (#PCDATA | %phrase;)*>

<!ELEMENT h3  (#PCDATA | %phrase;)*>

<!ELEMENT h4  (#PCDATA | %phrase;)*>

<!ELEMENT h5  (#PCDATA | %phrase;)*>

<!ELEMENT h6  (#PCDATA | %phrase;)*>


<!-- EISPP v2.0 now also allows anchors and links within FormattedText.
     Additionally, vulnerabilities and references can be linked to:

     The a-element has two non-standard attributes 'type' and 'iref'
     'type' can be set to 'vulnerability' or 'reference'.

     * By setting 'type' to 'vulnerability' and 'iref' to the value given in
       a vulnerability's name-attribute (see below), a specific vulnerability
       can be referenced.

     * By setting 'type' to 'reference' and 'iref' to the value given in
```

```
       a reference's name-attribute (see below), a specific reference
       can be referred to.
 -->

<!ENTITY % URI "CDATA">

<!ELEMENT a (#PCDATA | %phrase;)*>
<!ATTLIST a
  name        NMTOKEN        #IMPLIED
  href         %URI;          #IMPLIED
  type       NMTOKEN          #IMPLIED
  iref       NMTOKEN          #IMPLIED
  >



<!-- In a paragraph we allow normal text, no lists, tables, etc. -->

<!ELEMENT p %Inline;>

<!-- Unordered list -->

<!ELEMENT ul (li)+>

<!-- Ordered (numbered) list -->

<!ELEMENT ol (li)+>

<!-- list item -->

<!ELEMENT li %ListBody;>

<!-- definition lists - dt for term, dd for its definition -->

<!ELEMENT dl (dt|dd)+>

<!ELEMENT dt %Inline;>

<!ELEMENT dd %ListBody;>









<!--============================================================================-->
<!--================== EISPP advisory format ===================================-->
<!--============================================================================-->
<!--
Here we define the EISPP advisory format.

In its attributes we find some fields that the informal description
lists as identification data:
 - language :
   Making the language-information an attribute is standard for XML;
   putting it into the top element makes most sense. Because the
   format supports multiple-language content, the top-level language
   attribute defines the default language.

 - (EISPP-)issuer:
   Will not be displayed by most presentation-engines (which are for
   readers) and applies to the whole advisory
    ===> attribute in top-element.
```

```
-->

<!ELEMENT EISPP-Advisory (Id_Data,
                          History_Data?,
                          Vulnerability_Class,
                          System_Information?,
                          Description?,
                          Solution?,
                          Additional_Resources?)>

<!ATTLIST EISPP-Advisory
      version           CDATA                #REQUIRED
       issuer CDATA #REQUIRED
       xml:lang          %LanguageCode;        #REQUIRED
       date %Date; #IMPLIED

>


<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - -  Identification Data - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->


<!ELEMENT Id_Data (ref_num, title, abstract?)>


<!-- - - - - - - - - Ref_num- - - - - - - - - - - - - - - - - - - - - - - - -->
<!ELEMENT ref_num (#PCDATA)>

<!-- - - - - - - - - Title- - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- Here is the first occurrence of using FreeText to implement a multi-language
     feature.  -->

<!ELEMENT title (FreeText+)>

<!ELEMENT abstract (FreeText+)>

<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - -  History Data - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<!ELEMENT History_Data (version_history?, update_information?)>

<!-- - - - - - - - - version_history- - - - - - - - - - - - - - - - - - - - -->
<!-- Several fields identified in the informal description make excellent
     attributes, namely:
      - version
      - date

  If we want to have multi-language features, one question is whether
  those should be used for the change_descr. Probably yes, because I might
  want to display this info also to my readers, in which case I have
  to give it in all languages that I support in my CERT. The internal
  comments, on the other hand, will be in the working language of the
  issuing CERT.
  -->

<!ELEMENT version_history (change_descr+)>

<!ELEMENT change_descr (FreeText+, internal_comment?)>
<!ATTLIST change_descr        version  CDATA  #REQUIRED
        date            %Date;        #REQUIRED
>

<!ELEMENT internal_comment (#PCDATA)>
```

```
<!-- - - - - - - - - - update_information- - - - - - - - - - - - - - - - - - - - -->
<!-- Here is the first occurrence, where an element occurs that is not
     explicit in the informal description. There we find

    <update information> :== (<relation_tag> <ref_num>)*

     What "update_pointer" does is to give a name to the group
             (<relation_tag> <ref_num>)

     We have turned the relation_tag into an attribute; it fits its
     role, and this way we can contrain it with the DTD.
 -->

<!ELEMENT update_information (update_pointer*)>

<!ELEMENT update_pointer (#PCDATA)>
<!ATTLIST update_pointer
        relation %attvals.relation_attr; "complements"
>



<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - -  Vulnerability Classification- - - - - - - - - - - - -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->


<!ELEMENT Vulnerability_Class (vulnerabilities?,current_impact?,risk_ratings?)>



<!ELEMENT vulnerabilities (vulnerability+)>


<!ELEMENT vulnerability
(vuln_ids?,confidence_level?,vuln_cat?,requirements?,immediacy?,impact?,current_impact
?,risk_ratings?)>
<!ATTLIST vulnerability
        name NMTOKEN  #IMPLIED
>

<!ELEMENT vuln_ids (vuln_id+)>

<!ELEMENT vuln_id EMPTY>
<!ATTLIST vuln_id
        issuer %oLOV; #REQUIRED
       ref_num %token; #REQUIRED
>


<!ELEMENT confidence_level (explanation?)>
<!ATTLIST confidence_level
        type %attvals.confidence_rating_attr; #IMPLIED
>



<!ELEMENT requirements (explanation?)>
<!ATTLIST requirements
        type %attvals.requirements_type_attr; #IMPLIED
        subtype %attvals.requirements_subtype_attr; #IMPLIED
>


<!ELEMENT vuln_cat (explanation?)>
```

```
<!ELEMENT requirements (explanation?)>
<!ATTLIST requirements
        type %attvals.requirements_type_attr; #IMPLIED
        subtype %attvals.requirements_subtype_attr; #IMPLIED
>



<!ELEMENT immediacy (explanation?)>
<!ATTLIST immediacy
        vuln_status %attvals.vuln_status_attr; #IMPLIED
        prop_method %attvals.vuln_status_propagation_attr; #IMPLIED
        rating %attvals.rating_attr; #IMPLIED
>


<!ELEMENT impact (effects?, explanation?)>
<!ATTLIST impact
        rating %attvals.rating_attr; #IMPLIED
>


<!ELEMENT effects (effect+)>

<!ELEMENT effect EMPTY>
<!ATTLIST effect
        loss %attvals.loss_attr; #REQUIRED
        scope %attvals.scope_attr; #IMPLIED
>



<!ELEMENT current_impact (explanation?)>
<!ATTLIST current_impact
        rating %attvals.rating_attr; #REQUIRED
>


<!ELEMENT risk_ratings (risk+)>

<!ELEMENT risk (explanation?)>
<!ATTLIST risk
        schema %oLOV; #IMPLIED
        group %oLOV; #IMPLIED
        rating %attvals.rating_attr; #REQUIRED
>



<!ELEMENT explanation (FreeText*)>




<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - -  System Info - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<!ELEMENT System_Information (information+, system_list?)>

<!ELEMENT information (FormattedText+)>
<!ATTLIST information
        type %oLOV; #IMPLIED
>
```

```
<!-- - - - - - - - - -system_list- - - - - - - - - - - -->
<!-- For  machine-readable system information, a model
     of system information is necessary, which standardizes
     identifiers for platforms, software, version information, etc.

     A model of system information is orthogonal to
     an advisory format: various system models could be
     used together with the EISPP advisory format. The following
     definition for specifying a list of affected systems should
     be general enough to use with any model ofs system information.
     Which model is used must be communicated using the attribute
     'cat_model' in the top-level element 'system_list'
  -->

<!-- A system list contains one or more affected systems -->

<!ELEMENT system_list
    (system+)>
<!ATTLIST system_list
    cat_model CDATA #IMPLIED
>


<!-- A system may be specified by one or more system parts of
     different types; which types such as 'platform' and
     'software' must be defined in the model of system information.
-->


<!ELEMENT  system
    (system_part+)>


<!ELEMENT system_part
    (instance+)>

<!ATTLIST system_part
    type CDATA #REQUIRED
>

<!-- An instance of a system part is best understood by example.
     Consider a system part of type 'platform'. Instances could
     be 'Windows 2000', 'Windows XP', and 'RedHat Linux'.
     A standardized tag to identify such an instance must be
     defined by a model of system information
-->



<!ELEMENT instance
    (attribute_value*)>

<!ATTLIST instance
    tag CDATA #REQUIRED
>

<!-- Some models of system information may want to associate additional
     information such as version information with single instances. This
     can be done using the attribute_value element. The tag of the
     attribute_value element is used to communicate the kind of information
     (e.g., 'version', 'patchlevel', etc.) and must be standardized by
     the model of system information that is used. One or more values
    can be provided.

     Consider the following example, in which an instance 'w2k' (Windows 2000)
```

```
     is associated with patchlevel information:

          <instance tag = "w2k">
            <attribute_value tag = "patchlevel">
             <value>SP1</value>
             <value>SP2</value>
            </attribute_value>
          </instance>

-->

<!ELEMENT attribute_value
    (value+)
>
<!ATTLIST attribute_value
    tag CDATA #REQUIRED
>

<!ELEMENT value
    (#PCDATA)>


<!-- Here is a complete example of how machine-readable system information
  could be used to communicate that Apache 1.3.x and 2.0 is vulnerable
  on Windows 2000 and Windows XP, while on unix machines only
  Apache 2.x is vulnerable.


   <system_list cat_model="german_cert_wg">
       <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="os">
      <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>

-->




<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - -  Description - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
```

```
<!ELEMENT Description (description+)>

<!ELEMENT description (FormattedText+)>

<!-- In addition to the 'content type' mentioned in the informal description,
     an attribute for specifying a content 'subtype' has been added. Co-operating
     CERTs can define a proprietary use for this attribute.
-->

<!ATTLIST description
        type %oLOV; #IMPLIED
        subtype %oLOV; #IMPLIED
>

<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - -  Solution - - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->


<!ELEMENT Solution (sol_intro?, sol_section*)>

<!-- - - - - - - - - -sol_intro- - - - - - - - - - - - - - - - -->
<!ELEMENT sol_intro (FormattedText+)>

<!-- - - - - - - - - -sol_section  - - - - - - - - - -->
<!-- sol_type makes a good attribute -->

<!ELEMENT sol_section (sol_title, sol_descr?, reference*)>
<!ATTLIST sol_section
        type  %attvals.sol_type_attr; #IMPLIED
        subtype  %attvals.sol_subtype_attr; #IMPLIED
>

<!-- - - - - - - - - -sol_title - - - - - - - - - -->
<!ELEMENT sol_title (FreeText+)>

<!-- - - - - - - - - -sol_descr - - - - - - - - - -->
<!ELEMENT sol_descr (FormattedText+)>


<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - -  Additional resources- - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<!ELEMENT Additional_Resources (reference+)>


<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - -  reference- - - - - - - - - - - - - - - - - - - - - -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<!-- Many of the fields for "reference" identified in the informal
     description make good attributes -->

<!ELEMENT reference (ref_title?, uri*)>

<!ATTLIST reference
        name NMTOKEN #IMPLIED
        ref_type %attvals.ref_type_attr; #REQUIRED
        issuer   %oLOV; #IMPLIED
        ref_num  %token; #IMPLIED
>

<!-- - - - - - - - - -ref_title - - - - - - - - - - - - - - -->
<!ELEMENT ref_title  (FreeText+)>
```

```
<!-- - - - - - - - - -uri - - - - - - - - - - - - - - - - - -->
<!ELEMENT uri   (#PCDATA)>


<!ATTLIST uri
        xml:lang      %LanguageCode; #IMPLIED
        size          %token; #IMPLIED
        checksum      %token; #IMPLIED
        checksum_alg  %oLOV; #IMPLIED
>



<!-- ====================================================================

At last, some information regarding open Lists of Values.

- recognized values for the information type within the system information are

 platform_info
 software_info
 system_info
 remark


- recognized values for the information type within the system information are

  publication_context
  technical_context
  description
  technical_information
  diagnostic
  complete_advisory


For other oLOVs (issuers, etc.) see the EISPP documentation.

==================================================================== -->
```

6. USING THE EISPP FORMAT: FROM "EISPP LIGHT" TOWARDS FULL COMPLICANCE

The specification of the EISPP format describes how the format should be used: the description section should be split into logical subsections, references should be specified using an issuer name and a reference number, etc. Reaching full compliance in a single step may not always be possible. In the following, some suggestions are made about what lightweight versions of EISPP could look like. Such lightweight versions could ease the process of switching to the EISPP.

6.1. Minimal use of the EISPP Format

The following, minimal use of EISPP features could be especially useful for converting old advisories from a proprietary format into EISPP. Because EISPP is bound to be more structured than the proprietary format in question (which might just be an ASCII- or HTML-file), an automated conversion into fully compliant EISPP will not be possible. On the other hand, also an unstructured ASCII or HTML-format usually offers enough implicit structure in the form of standard headers, etc, that it should be relatively easy to extract at least the date of issue, a reference-number, a title, some system information, and a risk rating. Thus, the result of automated conversion from an old advisory in HTML could look like this:

```
<EISPP-Advisory version="2.0" issuer="ACME-CERT" xml:lang="en" date="2004-01-01">
  <Id_Data>
    <ref_num>ACME-2004-0001</ref_num>
    <title>
      <FreeText>Buffer Overflow in Foo package on Bar system</FreeText>
```

```
    </title>
  </Id_Data>
  <Vulnerability_Class>
    <vulnerabilities>
      <vulnerability>
       <risk_ratings>
         <risk schema="ACME" rating="high"/>
       </risk_ratings>
      </vulnerability>
    </vulnerabilities>
  </Vulnerability_Class>
  <System_Information>
   <information type="system_info">
    <FormattedText>Foo v1.3 on BAR OS</FormattedText>
   </information>
  </System_Information>
  <Description>
    <description type="complete_advisory">
      <FormattedText>
        A <em>buffer overflow</em> vulnerability has been discovered in the
        Foo package on the Bar system. Find more information
        <a href="http://www.foo-vendor.org/security/advisories/04/001">here</a>.
      </FormattedText>
    </description>
  </Description>
</EISPP-Advisory>
```

Thus, one important step has been achieved: legacy advisories can be handled with any infrastructure that understands the EISPP-advisory format. An advisory issuer that has switched to EISPP therefore does not have keep maintaining the old infrastructure for legacy advisories.

6.2. "EISPP Light"

In order to leverage at least some of the possibilities that EISPP offers, at least the following elements of the EISPP format should be used:

- **Inclusion of standard vulnerability identifiers**
  Standard vulnerability identifiers such as CVE numbers or Bugtrag IDs allow searching and grouping of advisories by vulnerabilities
- **Complete vulnerability classification**
  Vulnerability analysis (what are the preconditions for exploiting a vulnerability, what are the effects of successful exploitation, how imminent is the threat posed by a given vulnerability, etc.) is carried out by almost every advisory issuer to some extend. The EISPP advisory format provides a common language for exchanging the findings about a vulnerability, which can be used for quality control or even the sharing of workload.
- **Division of the *problem description* into logical subfields**
  By dividing the advisory text into logical subfields such as a description of the technical context, diagnostic information, etc., the re-use of advisory parts becomes easier. Additionally, advisories can be tailored to the audience by supressing those fields that are of little or no interest to a given audience when displaying the advisory.

Thus, a lightweight use of EISPP could look like this:

```
<EISPP-Advisory version="2.0" issuer="ACME-CERT" xml:lang="en" date="2004-01-01">
  <Id_Data>
    <ref_num>ACME-2004-0001</ref_num>
    <title>
      <FreeText>Buffer Overflow in Foo package on Bar system</FreeText>
    </title>
  </Id_Data>
  <Vulnerability_Class>
    <vulnerabilities>
      <vulnerability>
       <vuln_ids>
         <vuln_id issuer="CVE" ref_num="CAN-2004-0000"/>
```

```
        <vuln_id issuer="BID" ref_num="12345"/>
      </vuln_ids>
      <requirements type="remote_no_account"/>
      <immediacy vuln_status="exploitable" prop_method="automated" rating="medium"/>
      <impact rating="very_high">
        <effects>
          <effect loss="take_control" scope="system"/>
        </effects>
      </impact>
      <current_impact rating="high"/>
      <risk_ratings>
        <risk schema="ACME" rating="high"/>
      </risk_ratings>
     </vulnerability>
   </vulnerabilities>
  </Vulnerability_Class>
  <System_Information>
    <information type="system">
      <FormattedText>Foo v1.3 on BAR OS</FormattedText>
    </information>
  </System_Information>
  <Description>
    <description type="technical_context">
      <FormattedText>
       The Foo package is a standard component of the Bar OS. It implements
       the proprietary Wiggle-protocol.
      </FormattedText>
    </description>
    <description type="description">
      <FormattedText>
        A <em>buffer overflow</em> vulnerability has been discovered in the
        Foo package on the Bar system.
      </FormattedText>
    </description>
  </Description>
  <Solution>
    <sol_intro>
      <FormattedText>
       Patch your system. Patches can be accessed from
        <a href="http://www.foo-vendor.org/security/advisories/04/001">here</a>.
      </FormattedText>
    </sol_intro>
  </Solution>
</EISPP-Advisory>
```

For full compliance with EISPP, some things still are missing, e.g., the use of solution sections, reference structures for handling links, etc. Nevertheless, "EISPP light" already offers many opportunities for better advisory handling and co-operation on advisories between issuers.

7. ISSUES AND CONCLUSION

The exchange format for security advisories described in this document will be put to the test within EISPP: it will serve as a common basis for exchanging information, and collaborating on security advisories. Further revisions of the format will incorporate lessons learned through this cooperation.

The XML data-type description of this and future versions of the format, together with sample XSLT stylesheets for displaying advisory data are made publicly available on EISPP's website www.eispp.org.

The EISPP consortium invites everybody interested in the common format to download and use the materials. Comments and questions can be addressed via email to info@eispp.org

Document management information

Title	EISPP Common Advisory Format Description
Identifier	EISPP-D3-001-TR
Confidentiality[1]	PU
Status	A (Approved)
Creation Date	2002/07/30
Version	2
Revision	0
Revision Date	2004/05/20
Deliverable Reference	D3.01
Authors	Philippe Bourgeois, Oscar Conesa, Bernd Grobauer, Gareth Price
Keywords	EISPP, XML, Advisory, Format, Exchange

Document History

Version	*Date*	*Reason for modification*
1.0	2002-10-21	Approbation by EISPP partners
1.1	2003-02-05	Updated format description after discussions within WP3
1.2	2003-03-19	Factored out fast changing lists of values (issuer identifiers) into separate document. Rewrote some impact descriptions. Added a few sentences about <uri> in reference description. Added some notes about how to present XML advisories.
2.0	2004-05-20	Included changes in format descriptions that resulted from experiences within the EISPP trial period, feedback from interested CERTs and discussions and experiences collected within a working group of "German CERT working group" (Deutsche CERT AG)

[1] Confidentiality indicator according to the following table

PU	Public
PP	Restricted to other programme participants (including the Commission Services)
RE	Restricted to a group specified by the consortium (including the Commission Services)
CO	Confidential, only for members of the consortium (including the Commission Services)