Intrusion Detection Signatures Working Group
Internet Draft   A. Wierzbicki
                                                            J. Kalinski
                                                            T. Kruszona
Document: draft-wierzbicki-cidss-01.txt   Polish-Japanese
                                                           Institute of
                                                            Information
                                                             Technology
Expires: September 2005   March 2005

Common Intrusion Detection Signatures Standard

Status of this Memo

Abstract

The purpose of the Common Intrusion Detection Signatures Standard (CIDSS) is to define a common data format for storing signatures from different intrusion detection systems.

This Internet-Draft describes a common data format to represent information contained in signatures of intrusion detection systems, and explains the rationale for using this common format. The proposed format is a dialect of the Extensible Markup Language (XML). An XML Document Type Definition is developed, and examples are provided.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Table of Contents

1. Introduction

1.1 About CIDSS

Common Intrusion Detection Signatures Standard is intended to be a standard format of signatures used widely in Network Intrusion Detection Systems (NIDS). An IDS is controlled by a set of decision rules. A decision rule of an IDS is composed of two components: a description of specific characteristics of an intrusion attempt (a signature) and an action that has to be carried out when the data provided by IDS sensors matches the signature. This document focuses on the remaining part of an IDS decision rule: the IDS signature.

Currently, every IDS uses a different format of signatures. CIDSS defines a common format of signatures that attempts to express all information contained in signatures of various IDS. The CIDSS signature format is based on XML to facilitate the adaptation and applications of the proposed standard. The CIDSS signature format is designed to be extensible, and therefore it SHOULD be simple to incorporate features of future and current IDS systems that have not been taken into account in the first design.

The main goal of CIDSS is to enable administrators of IDS systems to share, compare, evaluate and criticize signatures used to detect intrusion events. The increasingly dynamic, global, and frequent nature of intrusion attempts is a trend that forces administrators to greater efforts to protect increasingly valuable information. The possibility to disseminate knowledge and experience about IDS systems' operation would be enhanced by the introduction of a common signature format. Therefore the use of a common IDS signature format SHOULD lead to a greater security of information. Other possible applications of CIDSS will be discussed in the next section.

CIDSS Homepage: http://cidss.b59.net

1.2 Potential Applications of CIDSS

One of the main applications of CIDSS is the translation of signatures between various IDS. The ability to translate a signature of an IDS into the common data format and to carry out a reverse translation implies that it SHOULD be possible to translate signatures of different IDS using the common data format as an intermediate form. The development of this standard has been carried out in parallel with the development of an IDS signature translator. Currently, the translator is able to translate signatures of Snort [5] and Dragon [6] IDS into the common format, and among the three systems. It's also partially tested with: Shoki [8], ISS RealSecure(TM) [11], and Cisco NetRanger(TM) [12]. The IDS translator is developed under the GNU General Public License and is available from http://translator.b59.net.

Another possible application of CIDSS would be the creation and maintenance of signature databases by independent providers of IDS signatures. The use of XML as a base for the common signature format enables a simple integration of collections of signatures into a database. This SHOULD improve the searching and management of a signature collection. The business model of an independent signature provider could be the delivery of up-to-date, comprehensive signature collections to increase security of specific services, systems and platforms.

```
   -------------------------------------
   |     First part of a signature     |
   |  <Sources>...</Sources>           |<-------|--|
   |  <Destinations>...</Destinations> |<-------|--|
   |  <Protocols>...</Protocols>       |<-------|--|
   |  <Patterns>...</Patterns>         |<-------|--|
   -------------------------------------        |  |
-------------------------------------------     |  |
|     Second part of a signature          |     |  |
|  <Session>                              |     |  |
|     <Session_Start>...</Session_Start>  |     |  |
|     <Session_End>...</Session_End>      |     |  |
|     <Session_Identification>            |     |  |
|        ...                              |     |  |
|     </Session_Identification>           |     |  |
|     <Session_Filter>...</Session_Filter>|-----   |
|     <Session_Instructions>              |        |
|        ...                              |---------
|     </Session_Instructions>             |
|   </Session>                            |
-------------------------------------------
```

Figure 1 The main components and logical structure of a CIDSS signature

2. XML CIDSS Signatures

This section describes the logical and structural rules for creating signatures in CIDSS format. Each XML element and attribute used in the CIDSS format are described and explained on examples. In appendix A, a full CIDSS signature is provided that has been used to provide the examples used in this section.
CIDSS meets XML ver. 1.0 requirements [9]. CIDSS is defined as a dialect of XML using the XML Schema Definition (XSD). The schema of CIDSS is an appendix to this document (see appendix B: CIDSS XSD schema. cidss.xsd)

2.1 Structure of a CIDSS document

A CIDSS document is a collection of signatures. Each signature is independent, and can be stored in a separate CIDSS document or together with other signatures. The main XML element of a CIDS document is the _Signatures_ element.

2.2 Structure of a CIDSS signature

A CIDSS signature is composed of several XML elements, contained in a common _Signature_ element. A signature can be divided into two basic, logical parts. The first part, that includes (among others) the elements _Sources_, _Destinations_, _Protocols_ and _Patterns_, is used to define building blocks of a signature definition. These blocks are combined in the second part of a signature, and are kept

separate in order to shorten the signature definition and avoid redundancy. For instance, the definition of relevant information about the TCP protocol might be kept inside the _Protocols_ element and might be later combined with several patterns (defined inside the _Patterns_ element). Rather than repeat the definition of the TCP protocol each time a new pattern is used, the signature definition will refer to the information kept inside the _Protocols_ element.
The second part of a signature contains information on how to use the building blocks defined in the first part. The main XML element of the second part of a signature is the _Session_ element. A _Session_ element defines the main signature behavior. A signature MAY use state managed by the IDS for a certain flow of packets (or session). However, it is possible to create stateless signatures, by omitting information REQUIRED for the state mechanisms to work. Then, the information contained in a _Session_ element defines the conditions that MUST be fulfilled by sensor data in order to trigger the signature.
In the second part of a signature, the information contained in the first part is combined using logical expressions. Each element in the first part of a signature that contains a _building block_ for the signature definition MUST have an identifier that is unique in the signature (not necessarily in the CIDSS document that contains the signature). This identifier can be used in the second part of a signature to refer to the _building block_ defined by this element.
The building blocks MAY be combined using logical expressions that are constructed by the _AND_ and _OR_ operators. The logical expressions are contained in special tags, and are treated as strings by the XML parser. CIDSS logical expressions are described in section 2.4.

2.3 Data types used by the Pattern_Content element

The data types used in CIDSS signatures are compatible with the XML[9] and XML Schema (XSD) [10] specification. The following data types are supported:

- String values
You MUST use encoding defined by "encoding" attribute (e.g. <?xml version="1.0" encoding="UTF-8" ?>). UTF-8 RECOMMENDED. Refer to Appendix A and Appendix B
- Hexadecimal values
- Decimal values

2.4 Logical expressions used in CIDSS signature definitions

Some elements in the CIDSS signature contain information that describes how other elements MUST be combined in the signature

definitions. The content of these elements is a String value that contains a logical expression. A translating software MUST be able to process these expressions.
CIDSS logical expressions MUST use operators _AND_, _OR_, and _NOT_ (uppercase). The operators are interpreted as follows:

- AND - logical conjunction
- OR - logical alternative
- NOT - logical negation

The operator precedence in CIDSS logical expressions MUST be interpreted as follows: NOT precedes AND precedes OR.
CIDSS logical expressions MAY contain ordinary braces _(_ or _)_ that are interpreted as in logical expressions.
Apart from braces and operators, CIDSS logical expressions MUST contain unique identifiers of other XML elements in the CIDSS signature definition that MAY be used in logical expressions.

2.5 XML elements and attributes used in CIDSS

In this section, all XML elements defined by the CIDSS standard SHALL be introduced. Each element will be defined using a common template to simplify a presentation. This template is explained below:

Element name

Element description.
Presence: [mandatory | optional, single | multiple]
Location: element name
Attributes: attribute name [type [, unique]]
Contained elements: element names

mandatory _ means that the element MUST exist in the signature
optional _ the element MAY exist in the signature
single _ if the element exists in the signature, then this element MUST exist in exactly one instance
multiple _ if the element exists in the signature, then this element MAY exist many instances
unique _ value of the element MUST NOT be repeated as value of the same element in other place

Signatures

A root element of a CIDSS XML document.

Presence: mandatory, single
Location: root element
Contained elements: Signature [multiple, mandatory]
Example:

```
<Signatures xsi:noNamespaceSchemaLocation="<schema_file.xsd>"\
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

where "<schema_file.xsd>" SHOULD be replaced by the filename of the XSD schema file (e.g. cidss.xsd)

Signature

This element contains all information about a signature. Describes conditions required to identify traffic as suspicious and to take an action.

Presence: mandatory
Location: element Signatures
Attributes: SID [type: integer, single, mandatory, unique]
Contained elements: Enabled [single, mandatory], Sig_Source [single, optional], Description [single, optional], Action [single, optional], Protocols [multiple, mandatory], Sources [multiple, mandatory], Destinations [multiple, mandatory], Patterns [single, mandatory], Logged_Packets [single, optional], Message [single, optional], Comment [multiple, optional]
Example: See Appendix A

Enabled

Defines a current signature state. Setting this to true, the signature will be analyzed by the IDS. Otherwise the signature SHOULD be skipped.

Presence: mandatory
Type: Boolean
Default value: true
Location: element Signature
Example: <Enabled>true</Enabled>

Sig_Source

Optional element for use in translators. Specifies the IDS from which the signature was translated or ported. This element SHOULD contain string that identifies a signature source.

Presence: optional
Type: String
Location: element Signature
Example: <Sig_Source>Snort</Sig_Source>

Description

This element MAY contain a simple description of the signature.

Presence: optional
Type: String
Location: element Signature
Example: <Description>Try to get passwd file using ftp

```
   protocol</Description>
```

Action

This MAY define actions performed by an IDS after intrusion detection
Suggested values: drop, allow, alert, and warning

Presence: optional, single
Type: String
Location: element Signature
Example: <Action>alert</Action>

Protocols

This element contains description of multiple protocols used in potential attack.

Location: Signature
Presence: mandatory, multiple
Attributes: ID[integer,unique]

Protocol

The element used to describe the network protocol. Options of the protocol used in this element depend on a protocol type.
The Proto_ID attribute is used for identification in the Proto_Logic element _ it is REQUIRED only when there is more than one Protocol in the Protocols element.

Presence: mandatory, multiple.
Type: String
Attributes: Proto_ID [integer, unique], Type [enum: tcp, udp, ip, icmp, application]
Location: element Signature
Contained elements: TCP_Ack [single, optional], TCP_State [single, optional], TCP_Seq [single, optional], TCP_Dsize [single, optional], TCP_Flags [single, optional], TCP_Window [single, optional], UDP_Dsize [single, optional], ICMP_Dsize [single, optional], ICMP_Icmp_Id [single, optional], ICMP_Icmp_Seq [single, optional], ICMP_Icode [single, optional], ICMP_Itype [single, optional], IP_Ttl [single, optional], IP_Tos [single, optional], IP_Ipopts [single, optional], IP_Fragbits [single, optional], IP_Id [single, optional], IP_Ip_Proto [single, optional], IP_Dsize [single, optional], Isdataat [single, optional], Rpc [single, optional]

Isdataat

Checks that the data fields in the packet are in the specified offset.

Allowed values: Integer or <integer (less than a given value) or >integer (more than a given value)

```
   Location: Protocol
   Presence: single, optional
   Example: <Isdataat><5</Isdataat>
```

Rpc

This element specifies the RPC application, version, and procedure numbers in SUNRCP call requests. It MUST contain a string in the following format:

Allowed format: <Rpc><application number>, [<version number>|*], [<procedure number>|*]></Rpc>
Location: Protocol, Type==_Application_
Presence: single, optional
Type: String
Example: <Rpc>100000,*,3</Rpc>

TCP_Ack

Checks the specific TCP ack number

Location: Protocol, Type==_TCP_
Presence: single, optional
Type: integer
Example: <TCP_Ack>0</TCP_Ack>

TCP_Seq

Checks the specific TCP seq number

Location: Protocol, Type==_TCP_
Presence: single, optional
Type: integer
Example: TCP_Seq>0</TCP_Seq>

TCP_State

Describes current protocol state e.g. established, stateless

Location: Protocol, Type==_TCP_
Allowed values: [established|stateless]
Presence: single, optional
Type: string
Example: <TCP_State>established</TCP_State>

TCP_Flags

Check if the specific TCP Flags are present

Location: Protocol, Type==_TCP_

Allowed values: [!|*|+][FSRPAU120]
Values description: F-FIN, S-SYN, R-RST, P-PSH, A-ACK, U-URG, 1-Reserved bit 1, 2-Reserved bit 2, 0-No Flags set.

Modifiers description: + - match on the specific bits, plus any others, * - match if any of the specified bits are set, ! _ match if specified bits are not set
Presence: multiple, optional
Type: String
Example: <TCP_Flags>+SA</TCP_Flags>

TCP_Window

Checks value of the TCP window size

Location: Protocol, Type==_TCP_
Presence: single, optional
Type: integer
Example: <TCP_Window>34000</TCP_Window>

TCP_Dsize

Checks the packet data field size in TCP protocol

Allowed signs: <, >, <=, >=, number
Location: Protocol, Type==_TCP_
Presence: single, optional
Type: String
Example: <TCP_Dsize><=40000</TCP_Dsize>

UDP_Dsize

Checks packet data field size in UDP protocol

Allowed signs: <, >, <=, >=, number
Location: Protocol, Type==_UDP_
Presence: single, optional
Type: String
Example: <UDP_Dsize><=33400</UDP_Dsize>

ICMP_Dsize

Checks the packet data field size in ICMP protocol

Allowed signs: <, >, <=, >=, number
Location: Protocol, Type==_ICMP_
Presence: single, optional
Type: String
Example: <ICMP_Dsize>>=64</ICMP_Dsize>

ICMP_Icmp_Id

Checks the value of specific ICMP ID

Location: Protocol, Type==_ICMP_
Presence: single, optional

```
   Type: integer
   Example: <ICMP_Icmp_Id>0</ICMP_Icmp_Id>

ICMP_Icmp_Seq

   Checks the value of ICMP sequence

   Location: Protocol, Type==_ICMP_
   Presence: single, optional
   Type: integer
   Example: <ICMP_Icmp_Seq>0</ICMP_Icmp_Seq>

ICMP_Icode

   Checks the value of specific ICMP code

   Allowed signs: <, >, number
   Location: Protocol, Type==_ICMP_
   Presence: single, optional
   Type: String
   Example: <ICMP_Icode>>25</ICMP_Icode>

ICMP_Itype

   Checks the value of specified ICMP type

   Allowed signs: <, >, number
   Location: Protocol, Type==_ICMP_
   Presence: single, optional
   Type: String
   Example: <ICMP_Itype>>25</ICMP_Itype>

IP_Ttl

   Specifies IP time-to-live value

   Allowed signs: <, >, <=, >=,-, number
   Location: Protocol, Type==_IP_
   Presence: single, optional
   Type: string
   Example: <IP_Ttl>6-8<IP_Ttl> - values between 6 and 8

IP_Tos

   Check the IP ToS field for specified value
   Location: Protocol, Type==_IP_

   Presence: single, optional
```

   Type: integer
   Example: <IP_Tos>2</IP_Tos>

IP_Fragbits

Checks fragmentations bits for the specified value

Location: Protocol, Type==_IP_
Presence: single, optional
Type: String
Example: <IP_Fragbits>DM+</IP_Fragbits>

IP_Id

Checks ID field in IP protocol for the specified value

Location: Protocol, Type==_IP_
Presence: single, optional
Type: integer
Example: <IP_Id>34222</IP_Id>

IP_Ipopts

This element checks if the specified IP option is present.

Allowed values: rr _ Record route, eol _ end of list, nop _ no op, ts _ Time Stamp, sec _ IP security option, lsrr _ Loose source routing, ssrr _ Strict source routing, satid _ Stream identifier, any _ any IP options are set
Location: Protocol, Type==_IP_
Presence: single, optional
Type: String
Example: <IP_Ipopts>lsrr</IP_Ipopts>

IP_Dsize

Checks size of packet data field

Allowed signs: <, >, <=, >=, number
Location: Protocol, Type==_IP_
Presence: single, optional
Type: String
Example: <IP_Dsize>34000</IP_Dsize>

IP_Ip_Proto

Checks IP protocol header for the specified value

Allowed signs: <, >, <=, >=, number, name
Location: Protocol, Type==_IP_
Presence: single, optional
Type: String

```
   Example: <IP_Ip_Proto>igmp</IP_Ip_Proto>
```

Proto_Logic

This element describes logical rules to combine the information in Protocol elements contained in one Protocols element. Logical operators in Proto Logic element are described in section 2.4.

Presence: optional, single
Location: element Patterns
Example: <Proto_Logic>1 OR (2 AND 3)<Proto_Logic>

Sources

This element contains information that describes properties of a source of network communications. If Sources occurs more then once, then every Sourcs MUST have an unique id (attribute) used in Src_Logic that defines logical dependences between each of them.

Presence: mandatory, multiple
Location: element Signature
Attributes: ID
Contained elements: Source[multiple, mandatory], Src_Logic [single, optional]
Example: See Appendix A

Source

This element contains descriptions of source hosts. Src_ID attribute is local (in one Sources element) id for use with the Src_Logic element.

Presence: mandatory, multiple
Location: element Sources
Attributes: Src_ID [presence: mandatory if more than one Source_ in one Sources element, type: integer, unique]
Contained elements: Source_IP[single, mandatory], Source_Port[single, optional]
Example: See Appendix

Destinations

This element contains information that describes properties of a destination of network communications. If Destinations occurs more then once, then every Destination MUST have an unique id (attribute) used in Dst_Logic to define logical dependences between each of them.

Presence: mandatory, multiple
Location: element Signature
Contained elements: Destination [multiple, mandatory]
Example: See Appendix A

Destination

This element contains descriptions of destination hosts. Dst_ID attribute is local (in one Destinations element) id for use with the Dst_Logic element.

Presence: mandatory, multiple
Location: element Destinations
Attributes: Dst_ID [presence: mandatory if more than one Destination_ in one Destinations element, type: integer, unique]
Contained elements: Destination_IP [single, mandatory], Destination_Port [single, optional]
Example: See Appendix A

Source_IP

This element contains an IPv4 or IPv6 network address in any notation. The value "any" means that all addresses will be considered and is an alias for 0.0.0.0 IPv4 address and ::0 for IPv6. If the value of Neg attribute is "true", then the values specified in the Source_IP element are interpreted as addresses that MUST NOT match the source address in order for the signature to apply. Mask attribute defines IP mask for current IP.

Allowed values: Any string
Attributes: Neg [presence: mandatory, type: boolean, allowed values: true|false, default: false], Mask [presence: mandatory, type: string, allowed values: mask in octet or bit notation]
Presence: mandatory, single
Type: String
Location: element Source
Example: <Source_IP Neg=_false_ Mask=_8_>$EXTERNAL_NET</Source_IP>
Variable $EXTERNAL_NET is defined in an IDS. (e.g. $EXTERNAL_NET=1.2.3.4)

Destination_IP

This element contains an IPv4 or IPv6 network address in any notation. The value "any" means that all addresses will be considered and is an alias for 0.0.0.0 IPv4 address and ::0 for IPv6. If the value of Neg attribute is "true", then the values specified in the Destination_IP element are interpreted as addresses that MUST NOT match the source address in order for the signature to apply. Mask attribute defines IP mask for current IP.

Allowed values: Any string
Attributes: Neg [presence: mandatory, type: boolean, allowed values: true|false, default: false], Mask [presence: mandatory, type: string, allowed values: mask in octet or bit notation]

Presence: mandatory, single
Type: String
Location: element Destination

Example: Similar as in Source_IP element

Source_Port

The value of this element is a port number or range of ports expressed by two port numbers divided with a _:_ sign. The _135:139_ expression means that all ports between 135 and 139 will be considered, accounting ports 135 and 139. The value "any" means that all ports will be considered.
Presence: If Protocol Type is set to tcp, udp or ip then mandatory, if Protocol Type value is icmp then MUST NOT be set.

Type: String
Location: element Source
Example: <Source_Port >any</Source_Port>

Destination_Port

The value of this element is a port number or range of ports expressed by two port numbers divided with a _:_ sign. The _135:139_ expression means that all ports between 135 and 139 will be considered, accounting ports 135 and 139. The value "any" means that all ports will be considered.

Presence: If Protocol Type value is set to tcp, udp or ip then mandatory, if Protocol Type value is icmp then MUST NOT be set.
Type: String
Location: element Destination
Example: <Destination_Port>445</Destination_Port>

Src_Logic

Defines logical dependences between each Source description. Logical operators: (,), AND, OR

Location: Sources
Example: 2 OR (1 AND 3)

Dst_Logic

Defines logical dependences between each Destination description. Logical operators: (,), AND, OR

Location: Destinations
Example: 1 AND 2

Patterns

   This element contains multiple Pattern elements.

Presence: mandatory, if Protocol is to tcp, udp, ip or application than element is present.

Location: element Signature
Contained elements: Pattern [multiple, optional]
Attributes: ID[integer, unique]
Example: See Appendix A

Pattern

This element contains information about the content of a packet that is considered as potentially dangerous. Attribute Pat_ID is used in Pat_Logic element to define logical expressions using multiple Pattern elements

Presence: mandatory, multiple
Location: element Patterns
Contained elements: Pattern_Type [single, mandatory], Pattern_Content [single, optional], Pattern_Depth [single, optional], Pattern_Uricontent [single, optional], Pattern_Offset [single, optional], Pattern_Within [single, optional], Pattern_Distance [single, optional]
Attributes: Pat_ID [integer, unique]
Example: See Appendix A

Pattern_Type

Using CIDSS you can specify patterns in hexadecimal, decimal, or string

Presence: mandatory
Type: String
Location: element Pattern
Permitted values: "hex", "dec", "string"
Default value: _string_
Example: <Pattern_Type>string</Pattern_Type>

Pattern_Content

Defines packet content that is interpreted as an intrusion and considered dangerous. To define the content, regular expressions can be used in the Pattern_Content element. Regular expression MUST be in the PCRE format (Perl Compatible Regular Expressions) [13]. If Rawbytes attribute value is _true_ it means pattern is searched in raw packets ignoring decoding that was done by preprocessors.

Presence: mandatory, single

Attributes: CaseSensitive [allowed values: true|false, default value: true], Rawbytes [allowed values: true|false, default value: false]
Type: Same as value of Pattern_Type
Location: element Pattern

Example: <Pattern_Content>RETR passwd</Pattern_Content>

Pattern_Depth

Defines how many bytes of the packet MUST be searched in order to find the content defined in the Pattern_Content element that is contained by the same Pattern element as this element.

Presence: optional, single
Location: element Pattern
Type: Integer
Example: <Pattern_Depth>10</Pattern_Depth>

Pattern_Uricontent

This element describes content of packet in URI format. If content is e.g. URL address it MAY be used in clear form in Pattern_Uricontent without special signs.

Type: string
Location: Pattern
Presence: optional, single
Example:
<Pattern_Uricontent>local/apache/htdocs/</Pattern_Uricontent>

Pattern_Offset

Specifies offset in bytes from beginning of packet to search for the pattern.

Type: integer
Location: Pattern
Presence: optional, single
Example: <Pattern_Offset>5</Pattern_Offset>

Pattern_Within

Used to describe how many packets MUST be at most between two patterns.

Type: integer
Location: Pattern
Presence: optional, single
Example: <Pattern_Within>4</Pattern_Within>

Pattern_Distance

Defines how far the IDS SHOULD look into a packet for the specified pattern relative to last match.

Type: integer
Location: Pattern
Presence: optional, single
Example: <Pattern_Distance>3</Pattern_Distance>

Pat_Logic

This element describes logical rules to combine the information in Pattern elements contained in one Patterns element. Logical operators in Pat_Logic expressions SHOULD be: OR, AND, NOT (as described in section 2.4).

Presence: optional, single
Location: element Patterns
Example: <Pat_Logic>(NOT 1 AND 2) OR 3<Pat_Logic>

Logged_Packets

Number of packets logged when the system detects an intrusion

Presence: optional, single
Location: element Signature
Example: <Logged_Packets>0</Logged_Packets>

Message

Contains the message generated by the IDS when a packet described by this signature was detected.

Presence: optional, single
Location: element Signature
Type: String
Example: <Message>FTP password file GET request</Message>

Comment

This element MAY be used for additional comments and information about the signature. The element MAY contain additional information about signature vendor, vulnerability description, http links etc.

Presence: optional, multiple
Location: element Signature
Example: <Comment>Vendor: Arachnids</Comment>

Session

Defines a session that could be identified by the signature if the state mechanisms of an IDS could be used. Otherwise, the information contained in this element describes the conditions that MUST be satisfied by the sensor data in order to trigger the signature.

Location: Signature

Presence: single, mandatory
Contained elements: Session_Filter [single, optional], Session_Start [single, optional], Session_End [single, optional], Session_Identification [single, optional], Session_Instructions [single, optional]

Session_Filter

Contains IDs of Source, Destination, Protocol and Pattern elements, combined using logical expressions in the format described in section 2.4. The information contained in this element specifies the conditions that MUST be met by sensor data so that the packet will be included in this session.

Location: Session
Presence: single, optional
Allowed values: CIDSS logical expressions.

Session_Start

Contains IDs of Source, Destination, Protocol or Pattern elements, combined using logical expressions in the format described in section 2.4. The information contained in this element specifies the conditions that MUST be met by sensor data so that the packet will define the beginning of a new session. All session state MUST be reset by the IDS when a new session begins.

Location: Session
Presence: single, optional
Allowed values: CIDSS logical expressions.

Session_End

Contains IDs of Source, Destination, Protocol or Pattern elements, combined using logical expressions in the format described in section 2.4. The information contained in this element specifies the conditions that MUST be met by sensor data so that the packet will define the beginning of a new session.
Instead of or in addition to conditions for sensor data, this element MAY include the element Session_Timeout, that specifies a timeout for the session or MAY include Session_Pckt_Count, that defines maximum number of packets considered in current session. When both conditions are specified, then the one that is fulfilled first will terminate the session.

Location: Session
Presence: single, mandatory if the Session_Start attribute is present
Contained elements: Session_Timeout [single, optional],

   Session_Pckt_Count [single, optional]

Session_Pckt_Count

Defines maximum number of packets that are considered during session.

Presence: single, optional
Location: Session_End
Type: Integer
Example: <Session_Pckt_Count>5</Session_Pckt_Count>

Session_Timeout

Defines a timeout for the session. The time MUST be specified in the format: an integer and a single character (the character MUST be one of: ms,s,m,h _ milliseconds, seconds, minutes, hours).

Presence: optional, single
Type: String
Location: Session_End
Example: <Session_Timeout>10s</Session_Timeout>
Example description: The timeout for the session is 10 seconds.

Session_Identification

Defines additional conditions that MUST be met by sensor data so that a packet will be included in this session. These conditions apply after a session has started. For instance, a TCP session will include only the packets that have the same source and destination as the source and destination of packets that started the session. The conditions are specified by including special elements in this element.

Location: Session
Presence: single, mandatory if the Session_Start attribute is present
Contained elements: Same_Source_IP [single, optional], Same_Source_Port [single, optional], Same_Destination_IP [single, optional], Same_Destination_Port [single, optional], Same_Protocol [single, optional], Same_Direction [single, optional]

Same_Source_IP

If this element is present in Session_Identification, packets that will be included in the session MUST have the same source IP address as the starting packet.

Type: boolean
Presence: single, optional
Location: Session_Identification

Same_Source_Port

If this element is present in Session_Identification, packets that will be included in the session MUST have the same source port as the starting packet.

Type: boolean
Presence: single, optional
Location: Session_Identification

Same_Destination_IP

If this element is present in Session_Identification, packets that will be included in the session MUST have the same destination IP address as the starting packet.

Type: boolean
Presence: single, optional
Location: Session_Identification

Same_Destination_Port

If this element is present in Session_Identification, packets that will be included in the session MUST have the same destination port as the starting packet.

Type: boolean
Presence: single, optional
Location: Session_Identification

Same_Protocol

If this element is present in Session_Identification, packets that will be included in the session MUST be of the same protocol as the starting packet.

Type: boolean
Presence: single, optional
Location: Session_Identification

Same_Direction

If this element is present in Session_Identification, packets that will be included in the session MUST have been sent in the same direction as the starting packet.

Type: boolean
Presence: single, optional
Location: Session_Identification

Session_Instructions

This element works like a switch statement for the state mechanism of an IDS. The information contained in this element defines the statefull behavior of an IDS for this session. The element contains several Session_Case elements that include conditions for the case to apply, as well as instructions to be carried out if the case applies. For efficiency reasons, it is assumed that all conditions for state instructions will be brought down into a conjunctive normal form (a logical expression that includes alternatives only at the highest level). That means that in every case element, all case conditions are treated as a logical conjunction (logical AND). This ought to simplify the processing of these instructions.

Location: Session
Presence: single, optional
Contained elements: Session_Case [multiple]

Session_Case

This element contains the conditions and instructions of a case in the switch statement that is defined by the containing Session_Instructions element. For readability, the conditions are split up into three groups: additional conditions for sensor data that MUST be satisfied so that the packet will apply to this case, the direction of the packet, and the conditions that MUST be satisfied by the state variables of a session in order for the case to apply. The instructions of a case are contained in the mandatory Case_State_Instructions element.

Location: Session_Instructions
Presence: multiple
Contained elements: Case_Filter [single, optional], Direction [single, optional], Case_State_Condition [single, optional], Case_State_Instructions [single, mandatory]

Case_Filter

Contains IDs of Source, Destination, Protocol or Pattern elements, combined using logical expressions in the format described in section 2.4. The information contained in this element specifies the conditions that MUST be met by sensor data so that the packet will apply to this case.

Location: Session_Case
Presence: single, optional
Allowed values: CIDSS logical expressions.

Direction

Defines a direction of network traffic, once a source and destination of traffic are specified (e.g. after the start of a session). Allowed values are: _sd_ and _ds_. If direction value is _sd_ it means that the packet has been sent from source to destination. If the value of this element is _ds_ it means that traffic goes from destination to source.

```
   Allowed values: sd|ds
   Default value: sd
   Location: Session_Case
```

Presence: single, optional

Case_State_Condition

This element contains conditional state expressions that MUST all be satisfied (evaluate to a boolean value of _true_) in order for the case to apply. These instructions MAY check the values of state variables stored by the IDS for this session.

Location: Session_Case
Presence: single, optional
Contained elements: Isset_Var, Compare_Var

Case_State_Instructions

This element contains state instructions that MUST all be sequentially carried out by the IDS if the case applies. These instructions MAY set, unset or modify the values of state variables stored by the IDS for this session.

Location: Session_Case
Presence: single, optional
Contained elements: Set_Var, Unset_Var

Isset_Var

A conditional state expression that evaluates to a boolean value of _true_ if the variable of a name that is specified in the _var_ attribute is set in the state of this session.

Location: Case_State_Condition
Presence: multiple, optional
Attributes: var [type: string; single, mandatory]

Compare_Var

Location: Case_State_Condition
Presence: multiple, optional
Attributes: var [type: string; single, mandatory], value [type: string; single, mandatory]

Set_Var

Sets value of _var_ attribute in state of particular session.

Location: Case_State_Instructions
Presence: multiple, optional
Attributes: var [type: string; single, mandatory], value [type:

string; single, mandatory]

Unset_Var

Nullifies value of _var_ used in this session.

Location: Case_State_Instructions
Presence: multiple, optional
Attributes: var [type: string; single, mandatory]

3. Security Considerations

This Internet Draft describes the CIDSS format for storing information about IDS signatures. The applications of this standard can raise security concerns, but there are no security concerns related strictly to the document format.

It is RECOMMENDED that a system for storing CIDSS data SHOULD be protected against unauthorized access and unauthorized use. The means for achieving this protection are outside the scope of this document.

Appendix A
XML CIDSS Document Example

Here we present a sample signature in CIDSS format. Elements of this signature have been used as examples in the previous sections. (This appendix MAY NOT be compatible with Internet Draft formatting).

```
<?xml version="1.0" encoding="utf-8" ?>
<Signatures xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="cidss.xsd">
   <Signature SID="1">
   <Enabled>true</Enabled>
   <Sig_Source>snort</Sig_Source>
   <Action>alert</Action>
   <Description>NETBIOS SMB-DS DCERPC Remote Activation bind attempt;
sid=2252</Description>
   <Message>NETBIOS SMB-DS DCERPC Remote Activation bind
attempt</Message>
   <Comment>reference:cve,CAN-2003-0528; reference:cve,CAN-2003-0605;
reference:cve,CAN-2003-0715;
reference:url,www.microsoft.com/technet/security/bulletin/MS03-
039.mspx;</Comment>
   <Sources ID="1">
      <Source Src_ID="SRC_1">
         <Source_IP neg="false" mask="0">any</Source_IP>
         <Source_Port>any</Source_Port>
      </Source>
      <Source Src_ID="SRC_2">
         <Source_IP neg="true" mask="8">10.0.0.0</Source_IP>
         <Source_Port>any</Source_Port>
      </Source>
      <Source Src_ID="SRC_3">
         <Source_IP neg="true" mask="24">192.168.1.0</Source_IP>
         <Source_Port>any</Source_Port>
      </Source>
      <Src_Logic>SRC_1 AND SRC_2 AND SRC_3</Src_Logic>
   </Sources>
   <Destinations ID="2">
      <Destination Dst_ID="DST_1">
         <Destination_IP neg="false" mask="0">any</Destination_IP>
         <Destination_Port>445</Destination_Port>
      </Destination>
      <Destination Dst_ID="DST_2">
         <Destination_IP neg="true"
mask="24">192.168.1.0</Destination_IP>
         <Destination_Port>445</Destination_Port>
      </Destination>
      <Destination Dst_ID="DST_3">
         <Destination_IP neg="true"
mask="8">10.0.0.0</Destination_IP>
         <Destination_Port>445</Destination_Port>
```

```
      </Destination>
      <Dst_Logic>DST_1 AND DST_2 AND DST_3</Dst_Logic>
   </Destinations>
   <Protocols ID="3">
      <Protocol Proto_ID="PROTO_1" Type="tcp">
         <TCP_State>established</TCP_State>
      </Protocol>
      <Proto_Logic>PROTO_1</Proto_Logic>
   </Protocols>
   <Patterns ID="4">
      <Pattern Pat_ID="PAT_1">
       <Pattern_Type>string</Pattern_Type>
         <Pattern_Content
CaseSensitive="false">|FF|SMB%</Pattern_Content>
         <Pattern_Depth>5</Pattern_Depth>
         <Pattern_Offset>4</Pattern_Offset>
      </Pattern>
      <Pattern Pat_ID="PAT_2">
       <Pattern_Type>string</Pattern_Type>
         <Pattern_Content
CaseSensitive="true">&#038;|00|</Pattern_Content>
         <Pattern_Within>2</Pattern_Within>
         <Pattern_Distance>56</Pattern_Distance>
      </Pattern>
      <Pattern Pat_ID="PAT_3">
       <Pattern_Type>string</Pattern_Type>
         <Pattern_Content CaseSensitive="false">|5C
00|P|00|I|00|P|00|E|00 5C 00|</Pattern_Content>
         <Pattern_Within>12</Pattern_Within>
         <Pattern_Distance>5</Pattern_Distance>
      </Pattern>
      <Pattern Pat_ID="PAT_4">
       <Pattern_Type>hex</Pattern_Type>
         <Pattern_Content CaseSensitive="true">05</Pattern_Content>
         <Pattern_Within>1</Pattern_Within>
      </Pattern>
      <Pattern Pat_ID="PAT_5">
       <Pattern_Type>hex</Pattern_Type>
         <Pattern_Content CaseSensitive="true">0B</Pattern_Content>
         <Pattern_Within>1</Pattern_Within>
         <Pattern_Distance>1</Pattern_Distance>
      </Pattern>
      <Pattern Pat_ID="PAT_6">
       <Pattern_Type>string</Pattern_Type>
         <Pattern_Content CaseSensitive="true">|B8|J|9F|M|1C|}|CF 11
86 1E 00| |AF|n|7C|W</Pattern_Content>
         <Pattern_Within>16</Pattern_Within>
```

```
            <Pattern_Distance>29</Pattern_Distance>
         </Pattern>
         <Pat_Logic>PAT_1 AND PAT_2 AND PAT_3 AND PAT_4 AND PAT_5 AND
   PAT_6</Pat_Logic>
      </Patterns>
      <Session>
         <Session_Filter>(SRC_1 AND SRC_2 AND SRC_3) AND (DST_1 AND
DST_2 AND DST_3) AND PROTO_1 AND (PAT_1 AND PAT_2 AND PAT_3 AND PAT_4
AND PAT_5 AND PAT_6)</Session_Filter>
         <Session_End>
            <Session_Pckt_Count>5</Session_Pckt_Count>
         </Session_End>
      </Session>
      </Signature>
   </Signatures>
```

Appendix B

The schema of CIDSS _ cidss.xsd

Available at http://translator.b59.net/docs/cidss.xsd

References

[1] Bradner S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Curry D., Lynch Merrill, Debar H., "The Intrusion Detection
Message Exchange Format", Internet Draft draft-ietf-idwg-idmef-xml-11, January 2004, expires July 2004.

[3] McLaughlin Brett, Java & XML, 2nd Edition, ISBN: 0-596-00197-5

[4] K. Miakisz, Translator i wspolny jezyk sygnatur systemow wykrywania wlaman (Translator and Common Intrusion Detection Systems Language), Bachelor thesis, Polish-Japanese Institute of Information Technology, 2003

[5] Roesch Martin, Green Chris, "Snort Users Manual", Snort Release 2.1.0, December 2003, http://www.snort.org

[6] "Dragon. Intrusion Detection System. Topics on Writing Signatures" Enterasys Networks, 2002, http://dragon.enterasys.com

[7] arachNIDS - Whitehats Network Security Resource, http://whitehats.com/ids/

[8]   Shoki, "Shoki User's Guide", Release 0.3.0,
      http://shoki.sourceforge.net/

[9]   Extensible Markup Language (XML) 1.0, Third Edition,
http://www.w3.org/TR/2004/REC-xml-20040204/

[10] XML Schema _ Specifications and Development,
http://www.w3.org/XML/Schema#dev

[11] ISS _ Internet Security Systems, Documentation,
http://www.iss.net/support/documentation/

[12] Cisco _ NetRanger, Documentation,
http://www.cisco.com/univercd/cc/td/doc/product/iaabu/\
netrangr/

[13] PCRE _ Perl Compatible Regular Expressions,
http://www.pcre.org/

Author's Addresses

dr Adam Wierzbicki
Polish-Japanese Institute of Information Technology
Koszykowa 86
02-008 Warsaw, Poland
Email: adamw@pjwstk.edu.pl

Jacek Kalinski
Rechniewskiego 6/24
03-980 Warsaw, Poland
Email: jacek@dyski.one.pl

Tomasz Kruszona
Garwolinska 9/83
04-348 Warsaw, Poland
Email: t.kruszona@b59.net

Comments to:

dr Adam Wierzbicki
Polish-Japanese Institute of Information Technology
Koszykowa 86
02-008 Warsaw, Poland
Email: adamw@pjwstk.edu.pl