Internet Draft   E. Boschi
draft-boschi-ipfix-implementation-guidelines-02.txt   Hitachi Europe
Expires: December 28, 2006   L. Mark
                                                      Fraunhofer FOKUS
                                                            J. Quittek
                                                        M. Stiemerling
                                                       NEC Europe Ltd.
                                                          Paul Aitken
                                                                 Cisco

June 26, 2006

IPFIX Implementation Guidelines
draft-boschi-ipfix-implementation-guidelines-02.txt

Abstract

The IP Flow Information eXport (IPFIX) protocol defines how IP Flow information can be exported from routers, measurement probes or other devices. This document provides guidelines for the implementation and use of the IPFIX protocol. A set of these guidelines refers to the implementation on middleboxes, such as firewalls, network address translators, tunnel endpoints, packet classifiers, etc.

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

The IPFIX protocol defines how IP Flow information can be exported from routers, measurement probes or other devices. In this document, we provide guidelines for its implementation. At the same time, open issues and unclear definitions are discussed while waiting to be corrected in the standard track document directly.

EDITOR'S NOTE: when clarifications about the open issues are brought up in the corresponding drafts, the open issues should disappear from this draft. If not corrected, the open issues should be treated as clarifications or suggestions for future improvements.

A set of the guidelines contained in this document addresses the IPFIX implementation on middleboxes. Middlebox functions

potentially change properties of traffic flows passing the box. For example, NATs change addresses in header fields and firewalls change the numbers of packets and bytes belonging to a traffic flow. An IPFIX implementation on a middlebox should reflect this by the way it selects and reports information about the Observation Point and by the way it measures and reports traffic flows.

Finally, this document contains a list of common mistakes about issues that were clear in the document but had been misinterpreted in the first IPFIX implementations and created (and still might create) interoperability problems.

1.1 History of IPFIX

Many applications require flow-based IP traffic measurements. In order to transmit IP flow information from an exporting process to an information collecting process, a common representation of flow data and a standard means of communicating them was required.

Several systems were presented at a BOF at IETF 51 (Argus, sFlow, CRANE, NetFlow v9, LFAPv5, DIAMETER) leading to the chartering of the IPFIX WG in the fall of 2001 with the goal of defining the standard.

Evaluation of Candidate Protocols [RFC3955] led to the recommendation to base IPFIX on NetFlow v9, a simple template based export protocol, that was evolved to meet the IPFIX requirements [RFC3917]. Differences between NetFlow v9 and IPFIX are listed in section 7.1 below.

1.2 IPFIX Documents Overview

The IPFIX protocol [IPFIX-PROTO] provides network administrators with access to IP flow information. The architecture for the export of measured IP flow information from an IPFIX exporting process to a collecting process is defined in [IPFIX-ARCH], per the requirements defined in [RFC3917]. This document specifies how IPFIX Data Records and Templates are carried via a congestion-aware transport protocol from IPFIX exporting processes to IPFIX collecting process. IPFIX has a formal description of IPFIX information elements, their name, type and additional semantic information, as specified in [IPFIX-INFO]. Finally [IPFIX-AS] describes what type of applications can use

the IPFIX protocol and how they can use the information provided. It furthermore shows how the IPFIX framework relates to other architectures and frameworks.

1.3 Overview of the IPFIX protocol

In the IPFIX protocol, templates contain { type, length } pairs specifying which { value } fields are present in data records conforming to the template, giving great flexibility as to what data is transmitted.

Since templates are sent very infrequently compared with data records, this results in a significant bandwidth saving.

Different data records may be transmitted simply by sending new templates specifying the { type, length } pairs for the new data format. See [IPFIX-PROTO] for more information.

[IPFIX-INFO] defines a large number of standard Information Elements which provide the necessary { type } information for templates.

The use of standard elements enables interoperability between different vendor s implementations. The list of standard elements may be extended in future through the process defined in section 5 below. Additionally, non-interoperable enterprise specific elements may be defined for private use.

2 Terminology

The terminology used in this document is fully aligned with the terminology defined in [IPFIX-PROTO]. Therefore, the terms defined in the IPFIX terminology are capitalized in this document, like in other IPFIX drafts ([IPFIX-PROTO, IPFIX-INFO, IPFIX-ARCH]).

3 Open issues and action items

[1] Enterprise specific Information Elements types. While enterprise IDs are publicly available and it s therefore straightforward to identify the enterprise, how to obtain the type of the given information element requires some clarification.

How to provide this information to the Collector? Which general mechanism(s) should be used?

[2] If an IPFIX Observation Point is co-located with one or more tunnel endpoints such that it observes packets that will be multiplexed into a tunnel or that have been de-multiplexed out of a tunnel, then the corresponding IPFIX Exporter SHOULD be able to report the corresponding tunnel ID.
Currently there isn't an IPFIX Information Element for TunnelIDs. (Cf. section 5.3.4)

[3] Add section on information exchange between metering and exporting process. How does the exporting process signal congestion? Who initiates the export of flow records? The metering process? The exporting process? Has the exporting process access to FlowRecords of the metering process (e.g. via shared memory)?

4 General Guidelines

An IPFIX message contains sets which in turn are a collection of one or more Records. There are two kinds of Records: Data Records contain Information Elements and Template Records the Information Elements Specifications.

4.1 Sets

A Set is identified by a Set ID [IPFIX-PROTO]. A Set ID has an integral data type and its value is in the range of 0 - 65535. The Set ID values of 0 and 1 are not used for historical reasons [RFC3954]. A value of 2 identifies a Template Set. A value of 3 identifies an Options Template Set. Values from 4 to 255 are reserved for future use. Values above 255 are used for Data Sets. In this case the SetID corresponds to the TemplateID of the used Template.

A Data Set received with an unknown Set ID MAY be stored pending the arrival of the corresponding Template (cf. section 9 of [IPFIX-PROTO]). If no Template soon becomes available the event should be logged and the association reset, since the data cannot be interpreted. The reset will cause Templates to be resent.

The arrival of a Set with a reserved or unused Set ID SHOULD be logged.

4.2 Template and Data Records

[IPFIX-PROTO] and [IPFIX-INFO] define the IPFIX protocol and standard Information Elements which can be exported using the protocol.

4.2.1 Template Management

The Exporter SHOULD send Template Records prior to the related Data Records. However, the Collector MAY store Data Records with an unknown Template ID pending the arrival of the corresponding Template (cf. section 9 of [IPFIX-PROTO]). If no Template soon becomes available the event should be logged and the association reset, since the data cannot be interpreted. The reset will cause Templates to be resent. For SCTP and TCP the Templates MUST only be resent on a connection re-establishment. As specified in [IPFIX-PROTO], when IPFIX uses UDP as the transport protocol, Template Sets and Option Template Sets MUST be re-sent at regular intervals (for more details see Section 4.6.2 below).

In either case, the Exporting Process MUST store all active Templates. This guideline can be ignored in case of simple exporters that have the data format hardcoded.

The Exporting Process is responsible for the management of Template Ids. Should insufficient Template IDs be available, the Exporting Process MUST send Template Withdraw message in order to free up the allocation of unused Template IDs. Note that UDP doesn t use the Template Withdraw message and the Template lifetime on the Collector relies on timeout.

4.2.2 Template Records versus Option Template Records

[IPFIX-PROTO] specifies the use of Template and Options Templates. Templates define the layout of Data Records: flows are exported as defined by (Data) Templates, while Option Templates define extra, additional information that doesn t fit in a flow. Options pertain to the control plane while (Data) Templates pertain to the data plane.

However, the choice of Template versus Options Templates to define the layout for exporting certain information about (or related to) Flows is left to the implementers. Indeed, there is a trade-off between bandwidth and complexity for the use of certain Information Elements in Options Templates. For example, sending information about the Observation Point (typically an interface) to the Collecting Process offers two different possibilities to the implementers.

The first one is to export information about the Observation Point as part of every Flow Record as defined by a Template Record. The advantage is simplicity of decoding at the Collector while the disadvantage is that the same information is sent as part of every Flow Record, wasting bandwidth for the export.

The second choice is to not export information about the Observation Point as part of every Flow Record defined by a Template Record, but to export it only once with Flow Record defined by an Options Template Record. The advantage is an optimization of the bandwidth while the disadvantage is a slightly increased complexity for the Collecting Process that has to combine the information from the Data Records defined by two different Templates: the Template Record and the Options Template Record. Note that, in this case, a unique ID for the Flow Record must be specified as a scope in the Options Template Record (Cf. [IPFIX-RED]).

4.2.3 Using Scopes

There's no concept of scope for exported data except for options data, and there's no default scope for IPFIX options, for which a scope MUST be specified. It is possible to specify specific scopes within a single Option Template which only affects option data corresponding to that Template and does not affect the scope of any other data.

4.3 Information Elements

4.3.1 Multiple Information Elements of same type

Exporters and Collectors MUST support the use of multiple Information Elements of the same type in a single Template [IPFIX-PROTO]. This can be needed for instance in PSAMP, when

multiple Selector IDs need to be exported. In this case, order dependency is crucial. The Exporting Process has to make sure to keep the Information Elements ordering given by the Metering Process. Information Elements of the same type have to be exported and stored maintaining the same order.

4.3.2 Order of Information Elements within the Template

Although it is not explicitly mentioned in the protocol draft the order of Information Elements within the Template CAN only be changed by Exporting or Collecting Processes as long as the processes are able to re-order both the IEs in the Template and the corresponding data values in all the associated Data Records.

If a Template contains multiple Information Elements of the same type, the order of these elements MUST be retained by Exporters and Collectors.

4.3.3 Information Element Coding

[IPFIX-ARCH] does not specify which entities have to do the encoding and decoding of Information Elements to be transferred via the IPFIX protocol. An IPFIX device can do the encoding either within the Metering Process or within the Exporting Process. The decoding of the Information Elements can be done by the Collection Process or by a user process of the data processing application.

If an IPFIX node simply relays IPFIX Records (like a proxy) then no decoding or encoding of Information Elements is needed. In this case the Exporting Process may export Information Elements of unknown type.

[IPFIX-PROTO] specifies: "The Collecting Process MUST note the Information Element identifier of any Information Element that it does not understand and MAY discard that Information Element from the Flow Record.". The Collecting Process MAY accept Templates with Information Elements of unknown types. In this case these data SHOULD be decoded as an octet array.

Alternatively, the Collecting Process MAY ignore Templates and subsequent Data Sets that contain Information Elements of unknown types.

4.3.4 Using counters

IPFIX offers both Delta and Total counters (e.g. octetDeltaCount, octetTotalCount). If information about a flow is only ever exported once, then it's not important whether Delta or Total counters are used. However, if further information about additional packets in a flow is exported after the first export then either:

- the metering system must reset its counters to zero after the first export and report the new counter values using delta counters.

Or

- the metering system must carefully maintain its counters and report the running total using total counters.

At first, reporting the running total may seem to be the obvious choice, but requires that the system accurately maintains the flow over a long time without any loss or error. When reported to a Collector, the previous total values will be replaced with the new information.

Delta counters offer some advantages: flows don't have to be maintained at all, and any loss of information has only a small impact on the total stored at the Collector. Finally, deltas may be exported in less bits than total counters using the IPFIX "Reduced Size Encoding" scheme [IPFIX-PROTO].

Note that delta counters have an origin of zero, and that a Collector receiving delta counters for a new flow must assume the deltas are from zero.

4.3.5 Padding

The IPFIX Information Model defines an Information Element for padding called paddingOctets [IPFIX-INFO]. It is of type

octetArray and the IPFIX protocol allows encoding it as a fixed-length array as well as a variable length array.

The padding Information Element can be used to align Information Elements within Data Records, Records within Sets, and Sets within IPFIX messages, as described below.

4.3.5.1 Alignment of Information Elements within a Data Record

The padding Information Element gives flexible means for aligning Information Elements within a Data Record. Aligning within a Data Record can be useful, because internal data structures can be easily converted into Flow Records at the Exporter and vice versa at the Collector.

Alignment of Information Elements within a Data Record is achieved by inserting an instances of Information Element paddingOctets with appropriate length before each unaligned Information Element. This insertion is explicitly specified within the Template Record or Option template record, respectively, that corresponds to the Data Record.

4.3.5.2 Alignment of Information Elements specifiers within a Template Record

There aren't means for aligning Information Element specifiers within Template Records, but there is a limited need for it and Information Element specifiers are aligned to 32-bit address boundaries anyway.

4.3.5.3 Alignment of Records within a Set

There no means for aligning Template Records or Option Template Records within a Set. However, for these records the need for alignment is limited and they are aligned to 32-bit boundaries anyway.

Data Record can be aligned within a Set by appending instances of Information Element paddingOctets at the end of the Record. Since all Data Records within a Set have the same structure and size, aligning one Data Records implies aligning all Data Records within a single Set.

4.3.5.4 Alignment of Sets within an IPFIX message

If Records are already aligned within a Set by using padding Information Elements, then this alignment is probably already achieved. But for aligning Sets within an IPFIX message, padding Information Elements can be used at the end of the Set so that the subsequent Set starts at an aligned boundary. This padding mechanism is described in section 3.3.1 of [IPFIX-PROTO and can be applied even if the records within sets are not aligned. However, it should be noted that this method is limited by the constraint that the padding length MUST be shorter than any allowable Record in the Set.

4.4 IPFIX Message Header Export Time and Data Record Time

Section 5 of the [IPFIX-PROTO] defines a method for an optimized export of time related Information Elements. This section contains recommendations on when to use this method and when not. Additionally, some general comments how to use timestamps in Data Records are provided.

[IPFIX-ARCH] distinguishes the Metering Process and the Exporting Process. The problem is that the Metering Process does not know when the IPFIX Message leaves the Exporting Process. This implies that the Metering Process has to store timestamp information i.e. in a 64 bit memory cell and has to provide the Exporting Process with the 64 bit data, while the Exporting Process has to convert the data e.g. to a 32 bit offset value. This implies some more CPU consumption by the Exporting Process, with the gain of a reduced bandwidth requirement for the export of Data Records as the timestamp related Information Elements would be coded with a reduced length.

Alternatively, the Exporting Process may send the absolute time related Information Elements. While the Exporting Process' job is simplified, this requires some more bandwidth for the export.

4.5 The Collecting Process's side

Template IDs are generated dynamically by the Exporting Process. They are valid only within the protocol stack. A restart of the Exporting Process will lead to a Template ID renumbering.

The Template IDs are unique per Exporting Process and Observation Domain. Therefore, the IPFIX Collector has to maintain a list of Exporting Processes and per Exporting Process a list of Observation Domains. For each Observation Domain a list of current Templates has to be maintained to decode subsequent data.

Because of the Template feature of IPFIX the Collector does not need any knowledge of the transferred data. All information needed to decode the data is transferred via the Template Records.

4.6 Transport Protocol

IPFIX Messages can be transferred using SCTP, TCP or UDP as bearer protocol. An IPFIX implementation MUST support SCTP-PR whereas support for TCP and UDP is optional [IPFIX-PROTO].

4.6.1 SCTP

Preference to SCTP-PR was given because it is congestion-aware and reduces bandwidth in case of congestion but still has a much simpler state machine than TCP. This saves resources on lightweight probes and router line cards.

One extra advantage of the SCTP-PR association is the notion of streams, for which the reliability mode can be chosen: fully reliable, partially reliable, or unreliable. The different levels of reliability are selected according to the different applications. For example, a billing application might require its Data Records to be sent on a reliable stream, while a security application might require a partially reliable stream, and a capacity planning application might require an unreliable stream.

The Collector may check whether IPFIX Messages are lost by checking the Sequence Number in the IPFIX header. The Collector SHOULD check whether IPFIX Messages are lost when using an unreliable or a partially reliable stream. If this is the case, for an unreliable stream the options are:

- To switch the traffic to a partially reliable stream on the Exporter

- To increase the bandwidth of the links through which the Data Records are exported

- To use sampling or filtering in the Metering Process to reduce the amount of exported data.

For a partially reliable stream, the options are:

- To increase the SCTP buffer size on the Exporter

- To increase the bandwidth of the links through which the Data Records are exported

- To use sampling or filtering in the Metering Process.

If the SCTP association is brought down because the IFPIX Messages can t be exported with the reliable stream, the options are:

- To increase the SCTP buffer size on the Exporter

- To increase the bandwidth of the links through which the Data Records are exported

- To use sampling or filtering in the Metering Process.

Note that Templates must NOT be re-sent when using SCTP (except when the SCTP association restarts), per section 8 of [IPFIX-PROTO]:

> Template Sets and Option Template Sets MUST be only sent once on SCTP stream zero with full reliability.

As of June 2006, to the best of our knowledge, the operating systems supporting SCTP-PR are: Solaris 10, Linux, and BSD (Cf. Section 9).

4.6.2 UDP

UDP is not a reliable transport protocol, and therefore IPFIX messages sent using UDP might be lost. [IPFIX-PROTO] specifies that Templates sent from the Exporting Process to the Collecting Process using UDP MUST be resent at regular intervals. The frequency of Template transmission MUST be configurable.

There are two possible implementations of retransmission intervals: time interval and packet interval. In the former case Templates are resent after a certain amount of time (e.g. every ten minutes). The resend times are fairly arbitrary and certainly depend on the application using it and on the export rate. If the time interval is too short however the Template retransmission would cause additional traffic resulting in overhead. On the other hand, if the time interval is too long it introduces costs due to the need of caching (big amounts of) data and higher risks to loose data if for some reason it cannot be cached or kept.
The Collecting Process SHOULD cache Data Records if the corresponding Template Record hasn't yet been received. The Collecting Process MAY drop cached data if it is holding data for more than 30 minutes.

In case of packet intervals Templates are resent depending on the number of packets sent. Similarly to the time interval, resending a Template every few packets introduces additional overhead while resending after a big amount of packets have been already sent means high costs due to the data caching and potential data loss.

Note that this could become an interoperability problem, e.g. if an Exporter re-sends Templates once per day, while a Collector expires Templates hourly, then they may both be IPFIX-compatible, but not be interoperable.

4.6.3 TCP

The use of TCP can be a fallback if one of the communication endpoints has no support for SCTP but a reliable transport is needed and the intermediate network is susceptible to congestion. TCP is one of the core protocols of the internet and is widely supported.

If the available bandwidth between exporter and collector is not sufficient or the metering process generates more data records than the collector is capable to process then the exporter would block. Options in this state are:

- To increase the TCP buffer size on the Exporter

- To increase the bandwidth of the links through which the Data Records are exported

- To use sampling or filtering in the Metering Process.

5 Guidelines for implementation on Middleboxes

The term middlebox is defined in RFC 3234 [RFC3234] as:

"A middlebox is defined as any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host."

The list of middleboxes discussed in RFC 3234 contains:

1. Network Address Translation (NAT),
2. NAT-Protocol Translation (NAT-PT),
3. SOCKS gateway,
4. IP tunnel endpoints,
5. packet classifiers, markers, schedulers,
6. transport relay,
7. TCP performance enhancing proxies,
8. load balancers that divert/munge packets,
9. IP firewalls,
10. application firewalls,
11. application-level gateways,
12. gatekeepers / session control boxes,
13. transcoders,
14. proxies,
15. caches,
16. modified DNS servers,
17. content and applications distribution boxes,
18. load balancers that divert/munge URLs,
19. application-level interceptors,
20. application-level multicast,
21. involuntary packet redirection,
22. anonymizers.

It is likely that since the publication of RFC 3234 new kinds of middleboxes have been added.

While the IPFIX specifications [IPFIX-PROTO] based the requirements on the export protocol only (as the IPFIX name implies), these sections cover the guidelines for the implementation of the Metering Process by specifying which Information Elements to export for the different middlebox considerations.

5.1 Traffic Flow Scenarios at Middleboxes

Middleboxes may delay, re-order, drop, or multiply packets; they may change packet header fields and change the payload. All these actions have an impact on traffic flow properties.
In general, a middlebox transforms a uni-directional original traffic flow T that arrives at the middlebox into a transformed traffic flow T' that leaves the middlebox.

```
                  +-----------+
           T ---->| middlebox |----> T'
                  +-----------+
```

Figure 1: Uni-directional traffic flow traversing a middlebox

Note that in an extreme case, T' may be an empty traffic flow (a flow with no packets), for example, if the middlebox is a firewall and blocks the flow.

In case of a middlebox performing a multicast function, a single original traffic flow may be transformed into a more than one transformed traffic flow.

```
                                +------> T'
                                |
                      +---------+-+
               T ---->| middlebox |----> T''
                      +---------+-+
                                |
                                +------> T'''
```

Figure 2: Uni-directional traffic flow traversing a middlebox with multicast function

For bi-directional traffic flows we identify flows on different sides of the middlebox: say T_l on the left side and T_r on the right side.

```
         +-----------+
T_l <--->| middlebox |<---> T_r
         +-----------+
```

Figure 3: Bi-directional unicast traffic flow traversing a middlebox

In case of a NAT T_l might be a traffic flow in a private address realm and T_r the translated traffic flow in the public address realm. If the middlebox is a NAT-PT, then T_l may be an IPv4 traffic flow and T_r the translated IPv6 traffic flow.

At tunnel endpoints, flows are multiplexed or de-multiplexed. In general, tunnel endpoints can deal with bi-directional traffic flows.

```
                   +------> T_r1
                   v
         +---------+-+
T_l <--->| middlebox |<---> T_r2
         +---------+-+
                   ^
                   +------> T_r3
```

Figure 4: Bi-directional traffic flow traversing a tunnel endpoint

An example is a traffic flow T_l of a tunnel and flows T_rx that are multiplexed into or de-multiplexed out of a tunnel. According to the IPFIX definition of traffic flows in [IPFIX-PROTO] T and T' or T_l and T_rx, respectively, are different flows in general.

However, from an application point of view, they might be considered as closely related or even as the same flow, for example if the payloads they carry are identical.

5.2 Location of the Observation Point

Middleboxes might be integrated with other devices. An example is a router with a NAT or a firewall at a line card. If an IPFIX Observation Point is located at the line card, then the properties of measured traffic flows may depend on the side of the integrated middlebox at which packets were captured for traffic flow measurement.

Consequently, an Exporting Process reporting traffic Flows measured at a device that hosts one or more middleboxes MUST clearly indicate to Collecting Processes the location of the used observation point(s) with respect to the middlebox(es). This can be done by using Options with Observation Point as Scope and elements like for instance linecard ID or sampler ID. Otherwise, processing the measured flow data could lead to wrong results.

At the first glance, choosing an Observation Point that covers the entire middlebox looks like an attractive choice for the location of the Observation Point. But this leads to ambiguities for all kinds of middleboxes. Within the middlebox properties of packets are modified and it MUST be clear at a Collecting Process whether packets were observed and metered before or after modification. For example, it must be clear whether a reported source IP address was observed before or after a NAT changed it or whether a reported packet count was measured before or after a firewall dropped packets. For this reason, [IPFIX-INFO] requires the use of Information Elements with prefix "post" for Flow properties that are changed within a middlebox.

Only in the case of composed middleboxes with well defined and well separated internal middlebox functions, for example a combined NAT and firewall, MAY an Observation Point be inside a middlebox, but in any case it SHOULD be located in between the middlebox functions.

5.3 Reporting Flow-related Middlebox Internals

While this document recommends IPFIX implementations using Observation Points outside of middlebox functions, there are few special cases where reporting flow-related internals of a middlebox is of interest.

For many applications that use traffic measurement results it is desirable to get more information than can be derived from just observing packets on one side of a middlebox. If, for example, packets are dropped by the middlebox acting as a firewall, NAT or traffic shaper, then information about how many observed packets are dropped may be of high interest.

This section gives recommendations on middlebox internal information that SHOULD or MAY be reported if the IPFIX Observation Point is co-located with one or more middleboxes. Since the internal information to be reported depends on the kind of middlebox, it is discussed per kind.

The recommendations cover middleboxes that act per packet and that do not modify the application level payload of the packet (except by dropping the entire packet) and that do not insert additional packets into an application level or transport level traffic stream.

Covered are the packet level middleboxes of kind 1 - 6, 8 - 10, 21, and 22 (according to the enumeration given at the beginning of section 4). Not covered are 7 and 11 - 20. TCP performance enhancing proxies (7) are not covered because they may add ACK packets to a TCP connection.

Still, if possible, IPFIX implementations co-located with uncovered middleboxes (i.e. of type 7 or 11 - 20) MAY follow the recommendations given in this section if they can be applied in a way that reflects the intention of these recommendations.

5.3.1 Packet Dropping Middleboxes

If an IPFIX observation point is co-located with one or more middleboxes that potentially drop packets, then the corresponding IPFIX Exporter SHOULD be able to report the number of packets that were dropped per reported flow.

Concerned kinds of middleboxes are NAT (1), NAT-PT (2), SOCKS gateway (3), packet schedulers (5), IP firewalls (9) and application level firewalls (10).

5.3.2 Middleboxes Changing the DSCP

If an IPFIX observation point is co-located with one or more middleboxes that potentially modify the DiffServ Code Point (DSCP, see [RFC2474]) in the IP header, then the corresponding IPFIX Exporter SHOULD be able to report both the observed DSCP value and also the DSCP value on the 'other' side of the middlebox (if this is a constant value for the particular traffic flow). Related Information Elements specified in [IFPIX-INFO] are: postClassOfServiceIPv4, and postClassOfServiceIPv6.

Note that the 'other' side of the middlebox can be before or after changing the DSCP value depending on the location of the Observation Point.

Note also that IPFIX doesn't support "pre" elements, only "post" elements, so the OP must be on the "before" (i.e. "pre") side.

Note also that a classifier may change the same DSCP value of packets from the same flow to different values depending on the packet or other conditions. Also it is possible that packets of a single uni-directional arriving flow contain packets with different DSCP values that are all set to the same value by the middlebox. In both cases there is a constant value for the DSCP field in the IP packets header to be observed on one side of the middlebox, but on the other side the value may vary. In such a case reliable reporting of the DSCP value on the 'other' side of the middlebox is not possible by just reporting a single value. According to the IPFIX information model [IFPIX-INFO], the first value observed for the DSCP is reported by the IPFIX protocol in that case.

This recommendation concerns packet markers (5).

5.3.3 Middleboxes Changing IP Addresses and Port Numbers

If an IPFIX Observation Point is co-located with one or more middleboxes that potentially modify the

- IP version field,
- IP source address header field,
- IP destination header field,
- TCP source port number,
- TCP destination port number,
- UDP source port number and/or
- UDP destination port number

in one of the headers, then the corresponding IPFIX Exporter SHOULD be able to report besides the observed value of the particular header fields also the 'translated' value of these fields, as far as they have constant values for the particular traffic flow.

Note that the 'translated' values of the fields can be the fields values before or after the translation depending on the Flow direction and the location of the observation point with respect to the middlebox. We always call the value that is not the one observed at the observation point the translated value.

Note also that a middlebox may change the same port number value of packets from the same flow to different values depending on the packet or other conditions. Also it is possible that packets of different uni-directional arriving flows with different source/destination port number pairs may be mapped to a single single flow with a single source/destination port number pair by the middlebox. In both cases there is a constant value for the port number pair to be observed on one side of the middlebox, but on the other side the values may vary. In such a case reliable reporting of the port number pairs on the 'other' side of the middlebox is not possible. According to the IPFIX information model [IFPIX-INFO], the first value observed for each port number is reported by the IPFIX protocol in that case.

Concerned kinds of middleboxes are NAT (1), NAT-PT (2), SOCKS gateway (3) and involuntary packet redirection (21).

This recommendation MAY also be applied to anonymizers (21), but it should be noted that this includes the risk of losing the effect of anonymisation.

5.3.4 Tunnel Endpoints

If an IPFIX Observation Point is co-located with one or more tunnel endpoints such that it observes packets that will be multiplexed into a tunnel or that have been de-multiplexed out of a tunnel, then the corresponding IPFIX Exporter SHOULD be able to report the corresponding tunnel ID.

Note that currently there isn't an IPFIX Information Element for TunnelIDs.

6 Extending the Information Model

New Information Elements can be added to the protocol in two different ways. If an Information Element is considered of general interest, it SHOULD be added to the base set of Information Elements for IPFIX. The request process for a new IETF Information Element is defined in 6.1 (and cf. also [IPFIX-INFO]). Private Enterprises can in alternative define Proprietary Information Elements for internal purposes (because, for example, they are delivering a pre-standards product, or the Information Element is in some way commercially sensitive [IPFIX-PROTO]). Details on this method are provided in section 6.2.

The [IPFIX-INFO] document contains an XML-based specification of Template, abstract data types and IPFIX Information Elements, which may be used to create consistent machine-readable extensions to the IPFIX information model. This formal description can be used for automatically checking syntactical correctness of the specification of IPFIX Information Elements or for generating code that deals with processing IPFIX Information Elements.

6.1 Adding new IETF specified Information Elements

If the Information Elements are considered of general interest they SHOULD be added to the group of IETF specified IPFIX Information Elements to extend the current IPFIX Information Model [IPFIX-INFO]. The list of IETF specified Information Elements will be administered by IANA.

The introduction of new Information Elements in the IANA registry is subject to review by experts drawn from the IPFIX and PSAMP Working Group Chairs and document editors (cf. [IPFIX-INFO]).

Until IANA has created this registry, the list of IETF specified Information Elements will be administered by the IPFIX working group. During this initial period, the list of allocated IEs will be kept and administered at a web site maintained by the IPFIX WG. The IPFIX Working Group will also take care of the IEs

review process and the administration of the IE-reviewers
mailing list to reach the experts described above.

On the IPFIX web site, the following information will be
available:
1. The list of Information Elements already agreed by the IPFIX
Working Group.
2. Brief overview of the request process.
3. Links to the IPFIX and PSAMP Information Model RFCs.
4. Information Element request form and request template.

When submitting the request, the request template provided on
the request page MUST be used; it ensures that requests match
the template for Information Element specifications defined in
[IPFIX-INFO].
The expert evaluation will be notified not later than two months
after the request has been received. The inclusion on the IE
list will be effective immediately after expert approval.

If a request is turned down, the requestor can treat the
Information Elements as enterprise-specific fields. Every
organisation can request an Enterprise Number at IANA with
minimal overhead. This method is described in the following
section

[TODO: indicate whether there is an "appeal" process, ie what a
requestor can do if they are turned down. Is the expert's
decision final?]

6.2 Adding enterprise-specific Information Elements

A faster way of introducing new Information Elements or the way
for vendors to integrate proprietary Information Elements in
IPFIX is by using enterprise-specific Information Elements (cf.
[IPFIX-PROTO]).

Enterprise Specific Information Elements can be chosen
arbitrarily within the range of 1-32767 and have to be coupled
with an Enterprise Identifier [PEN]. Enterprise identifiers MUST
be registered as SMI network management private enterprise code
numbers with IANA. The registry can be found at
http://www.iana.org/assignments/enterprise-numbers [IPFIX-INFO].

When receiving Information Elements from vendors the following information is directly available to the Collector:
- The vendor specific Information Element identifier
- Its length
- The enterprise ID.

Enterprise IDs are publicly available and it s therefore straightforward to identify the enterprise.

[TODO: more on how to obtain the type of the given information element]

7 Implementation mistakes

It seems useful to list a few things that were clear in the document and not needing clarification that some implementers didn't do correctly. All of these things caused or may cause interoperability problems.

7.1 IPFIX and NetFlow version 9

A large group of mistakes stems from the fact that many implementers started implementing IPFIX from an existing version of NetFlow version 9 [RFC3954]. Despite their similarity, the two protocols differ in many aspects. We list here some of the most important differences.

- Transport protocol: NetFlow version 9 has been initially running over UDP while the IPFIX must have congestion aware transport protocol. IPFIX specifies SCTP-PR as its mandatory protocol, while TCP and UDP are optional.

- IPFIX differentiates between IETF and non-IETF defined Information Elements. Non-IETF Information Elements can be specified by coupling the non IETF Information Element identifier with an Enterprise ID (corresponding to the vendor that defined the Information Element).

- Option Templates: in IPFIX an Option Template must have a scope and the scope is not allowed to be of length zero. The NetFlow version 9 specifications [RFC3954] don t specify that the scope must not be of length zero.

Message header:

- Set ID: Even if the packet headers are different between IPFIX and NetFlow version 9, some of the fields are used in both of them. The difference between the two protocols is in the values that these fields can assume. A typical example is the Set ID values: the Set ID values of 0 and 1 are used in NetFlow version 9, while they are not used in IPFIX.

- Length field: in NetFlow version 9, this field (called count) contains the number of Records. In IPFIX, it indicates the total length of the IPFIX message, measured in octets (including message header and Set(s)).

- Timestamp: NetFlow version 9 has an additional timestamp: sysUpTime. It indicates the time in milliseconds since the last reboot of the Exporter.

- The version number is different. NetFlow version 9 uses the version number 9 while IPFIX uses the version number 10.

7.2 Padding of the Data Set

[IPFIX-PROTO] specifies that the Exporting Process MAY insert some padding octets to align Information Elements within a Data Record. The padding length MUST be shorter than any allowable Record in that set.

It is important to respect this limitation: if the padding length is equal to or longer than the length of the shortest Record, it will be interpreted as another Record.

An alternative is to use the paddingOctets Information Elements in the Template definition.

7.3 Field ID Numbers

If the Information Element identifier in the Data Record has a value such that the first bit is "1", the Collector interprets the fields following the length fields as an enterprise number.

There is no way to tell that this is wrong, if it wasn't meant as an enterprise Data Record.

7.4 Template ID Numbers

Template IDs are generated as required by the Exporting Process. When exporting Templates composed by the same set of Information Elements at different times or using Templates composed by the same set of Information Elements multiple times simultaneously, different Template IDs are generated for each Template.

So the collecting process does not know in advance which Template ID a particular Template will use.

7.5 Information Elements Spelling

The spelling of the data type names dateTimeMilliSeconds, dateTimeMicroSeconds, and dateTimeNanoSeconds in [IPFIX-INFO] requires writing the "s" in seconds upper-case (i.e. "S"). Since usually capital letters are required with wordbreaks, attempting to find the flowStartMilliseconds (with "s" low-case, thought as part of the word Milliseconds) IE in an IE registry would cause an error. The same error might occur when looking for Microsends or Nanoseconds.

8 Security Considerations

This document describes the implementation guidelines of IPFIX. The security requirements for the IPFIX target applications are addressed in the IPFIX requirements draft [RFC3917]. These requirements are considered for the specification of the IPFIX protocol, for which a security considerations section exits [IPFIX-PROTO].

Section 4.3 recommends that IPFIX Exporting Processes report internals about middleboxes. These internals may be security-relevant and the reported information needs to be protected appropriately for reasons given below.

Reporting the packets dropped by firewalls and other packet dropping middleboxes imply the risk that this information is

used by attackers for analyzing the configuration of the packet dropper and for developing attacks that pass the middlebox.

Address translation may be used for hiding the network structure behind an address translator. If an IPFIX Exporting Process reports the translations performed by an address translator, then parts of the network structure may be revealed.

If an IPFIX Exporting Process reports the translations performed by an anonymizer, the main function of the anonymizer may be compromised.

Also information about which packet enters or leaves which tunnel may need protection.

9 Code availability

This section provides links where to gather IPFIX implementations (or code related to IPFIX) that have been made freely available by their implementers.

Link: http://libipfix.sourceforge.net
Organisation: Fraunhofer FOKUS
Description: IPFIX C library, distributed under the BSD license. Full support for SCTP, UDP, TCP, IPv4 and IPv6 over Linux, FreeBSD, Solaris.

Link: http://www.ntop.org/
Organisation: Netikos S.p.A.
Description: distributed under the GPL2 license. Runs over Linux.

Link: http://www.cert.org/netsa/tools/fixbuf/
Organisation: CERT / NetSA
Description: distributed under the GPL or LGPL licenses. This code has been tested on Linux, Free/OpenBSD, and Mac OS X, but should be usable without change on other Unix platforms.

10 IANA Considerations

This document has no actions for IANA.

11 Normative References

[RFC3917] J. Quittek, T. Zseby, B. Claise, S. Zander, Requirements for IP Flow Information Export, October 2004.

[IPFIX-PROTO] B. Claise (Editor), IPFIX Protocol Specification, Internet-Draft <draft-ietf-ipfix-protocol-22.txt>, work in progress, June 2006.

[IPFIX-INFO] J. Quittek, S. Bryant, J. Meyer, Information Model for IP Flow Information Export, Internet-Draft <draft-ietf-ipfix-info-12.txt>, work in progress, June 2006.

[RFC2474] K. Nichols, S. Blake, F. Baker, and D. Black, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC 2474, December 1998.

12 Informative References

[IPFIX-ARCH] G. Sadasivan, N. Brownlee, B. Claise, J. Quittek: Architecture for IP Flow Information Export, Internet-Draft, draft-ietf-ipfix-architecture-11.txt, work in progress, June 2006.

[IPFIX-AS] T. Zseby, E. Boschi, N. Brownlee, B. Claise, IPFIX Applicability, Internet-Draft, draft-ietf-ipfix-as-08.txt, work in progress, May 2005.

[RFC3234] B. Carpenter, and S. Brim, Middleboxes: Taxonomy and Issues, RFC 3234, February 2002.

[RFC3917] J. Quittek, T. Zseby, B. Claise, S. Zander, Requirements for IP Flow Information Export, RFC 3917, October 2004.

[RFC3954] B. Claise, et al. Cisco Systems NetFlow Services Export Version 9, RFC 3954, October 2004.

[IPFIX-RED] E. Boschi, L. Mark, B. Claise, Reducing Redundancy in IPFIX and PSAMP reports, Internet-Draft, draft-boschi-ipfix-reducing-redundancy-02.txt, work in progress, June 2006

[PEN] IANA Private Enterprise Numbers registry http://www.iana.org/assignments/enterprise-numbers

13 Acknowledgements

We would like to thank the MoMe project for organising two IPFIX Interoperability Events in July 2005 and in March 2006 that provided us precious input for this draft. We would also like to thank Benoit Claise, Carsten Schmoll, and Brian Trammell for the technical review and feedback.

14 Author's Addresses

Elisa Boschi
Hitachi Europe SAS
Immeuble Le Theleme,
1503 Route des Dolines
06560 Valbonne, France
Phone: +33 4 89874180
Email: elisa.boschi@hitachi-eu.com

Lutz Mark
Fraunhofer Institute for Open Communication Systems (FOKUS)
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
Phone: +49 30 3463 7306
Email: mark@fokus.fraunhofer.de

Juergen Quittek
NEC Europe Ltd.
Network Laboratories
Kurfuersten-Anlage 36
69115 Heidelberg, Germany
Phone: +49 6221 90511-15
Email: quittek@netlab.nec.de

Martin Stiemerling
NEC Europe Ltd.
Network Laboratories
Kurfuersten-Anlage 36
69115 Heidelberg, Germany
Phone: +49 6221 90511-13
Email: stiemerling@netlab.nec.de

Paul Aitken
Cisco Systems
96 Commercial Quay
Edinburgh
Scotland
EH6 6LX
Phone: +44 131 561 3616
Email: paitken@cisco.com

15 Intellectual Property Statement

16 Copyright Statement

17 Disclaimer