IPFIX 2

Paul Aitken

87th IETF Meeting, Berlin, 2013

IPFIX History

- IPFIX BoF, IETF 49 (Dec 2000) + 51 (Aug 2001) 12½ years
- IPFIX WG created September 2001 ~12 years
- draft-ietf-ipfix-reqs-00.txt = November 2001 ~12 years
- draft-ietf-ipfix-architecture-00 = February 2002 11 years
- First IPFIX interop, Paris, July 2005 8 years
- RFC5101 = January 2008 5 years

- 20 RFCs
- 5 more in RFC Editor queue
- 3 WG drafts in progress

IPFIX2 Scope

- Solve IPFIX problems, issues, enhancements
- Want to solve real issues with IPFIX
  - with real use cases which need WG focus
- Not just engineering nice-to-haves
  - ie, improving the protocol
- Be careful of the political / marketing message
  - “IPFIX is somehow inadequate and needs redesigned”

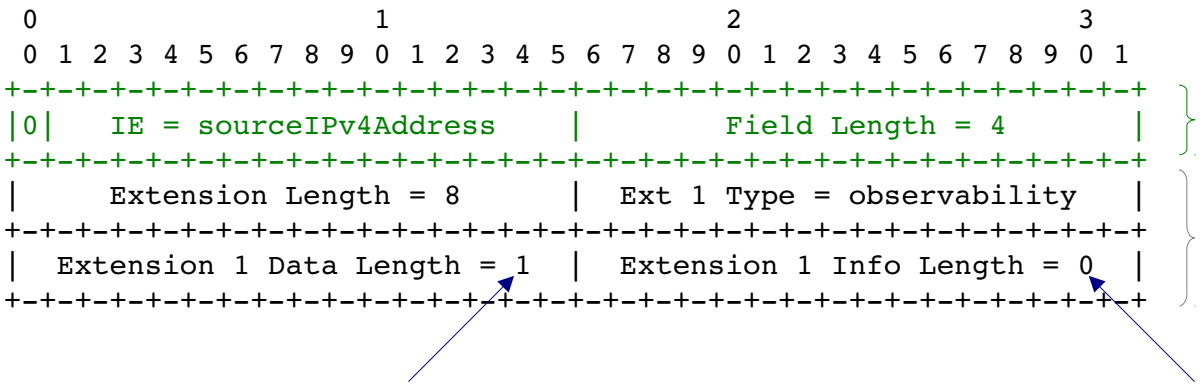
Field attributes: overview

- How can encapsulated protocols be reported?
  - multiple new IEs like the MPLS labels?
- Similar issue for indexing
  - eg, this is the n^{th} instance of this IE
- MIB export
  - Relating the MIB OID, indices, value.
- Creation of EFSF in draft-ietf-ipfix-mib-variable-export-01

Field attributes: IETF86 summary

- Paul presented the MIB Variable Export draft:
  - The draft proposes to introduce Extended Field Specifier Format (EFSF), with 'decorators' that could, for example, be used to handle Unobserved Fields.
  - EFSF generated a long discussion - it could be used for many other things; this concept could provide a new, elegant and concise way of handling attributes of particular IEs.
  - Consensus in the meeting was that EFSF is a new direction for IPFIX - it's really IPFIXv2.
  - Paul will remove it from the MIB Variable Export draft
  - The WG should adopt this as a work item.
  - That will require a new charter; meanwhile work on it can proceed on the IPFIX list.

EFSF: example for unobserved fields

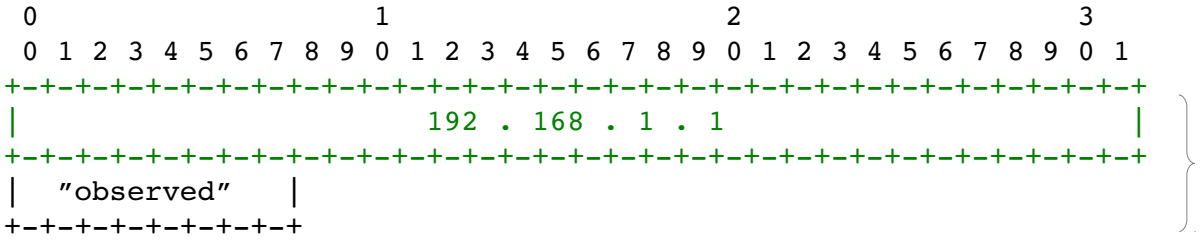


Standard Field Specifier

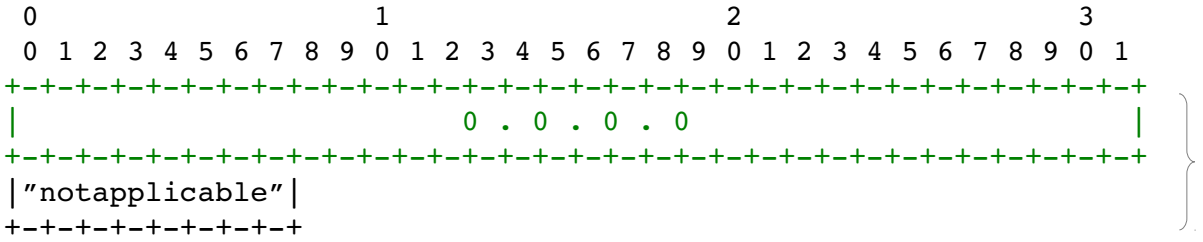
"Observability" Extension

Contributes 1 extra octet to Data Records.

No further information within the template



data record for IPv4 traffic



data record for non-IPv4 traffic

EFSF: what can it be used for? 1/2

Type	Length	Value	Details
Key	0	key / non-key	Key fields distinguish one flow from another.
Non-key behaviour	1	min / max / average / first / last	How the value of a non-key field was determined.
Direction	0	ingress / egress	Whether traffic was ingress or egress.
Observation point	1	OP ID	Location where the traffic was observed. Eg, interface, NAT process, QOS process...
Pre / Post	0	pre / post	Whether the observation was made before (pre) or after (post) packet treatment.
Biflow direction	0	forward / reverse	Forward versus reverse fields, without the clumsy RFC 5103 PEN mechanism.
Biflow strategy	0	initiator / responder	Which side of the biflow is which?
Counter semantics	0	delta / total	deltaCounter versus totalCounter sematics, without requiring duplicate fields.
Aggregation count	4	original / aggregated	How many flows were aggregated together. A value of “1” indicates an unaggregated flow.
Time	1	start / end	Start and end timestamp, without requiring duplicate fields.
MIB	N	MIB OID	The OID of the MIB being exported.
Observability	1	observed / not available / not applicable	Indicates whether a value was observed, and why not.

EFSF: what can it be used for? 2/2

Type	Length	Value	Details
Offset	1	packet offset	The offset of the captured data within a packet section.
Autonomous system	0	peer / origin	Whether the AS ID is from a peer or origin.
Interface type	1	physical / logical / channelised / virtual	The interface type.
Error type	0	absolute / relative	Whether the error is absolute or relative.
Error amount	4	amount of error	
Hash options	tbd	tbd	tbd
Name	N	(string)	informationElementName
Range	N	X, Y	informationElementRangeBegin, informationElementRangeEnd
Semantics	1	(semantics)	informationElementSemantics
Units	1	(units)	informationElementUnits
Index	N	Field index	Field index, eg encapsulation layer.
Enterprise-specific	4	PEN	Indicates the PEN for ES elements.

EFSF: use cases 1/3 : IE equivalence

- Today we export “ingressInterface” and “egressInterface” and assume that “interfaceName” applies equally to both.
- Since interfaceName is directionless, use EFSF with direction, index, and name properties:
- Data record:
  interface.{dir=ingress} = 123
  interface.{dir=egress} = 456
- Option record:
  interface.{index=123}.{name=“eth1”}
  interface.{index=456}.{name=“eth2”}

EFSF: use cases 2/3 : index and encaps

- Indexing multiple instances of an IE within a data record, eg MPLS label stack:
  MPLSlabel.{stackLevel=1} = xxxx
  MPLSlabel.{stackLevel=2} = yyyy
  MPLSlabel.{stackLevel=3} = zzzz

- Reporting traffic hierarchy and inner headers.
  eg, report IPv6 encapsulated in IPv4:

  sourceIPv4address.{encapsLevel=1}
  destinationIPv4address.{encapsLevel=1}

  sourceIPv6address.{encapsLevel=2}
  destinationIPv6address.{encapsLevel=2}

EFSF: use cases 3/3

- Application export:
  app.{id} = 123
  app.{engine} = NBAR
  app.{name} = “http”
  app.{subapp}.{browser} = chrome
  app.{subapp}.{browser}.{version} = 25.0.1364.172
  app.{subapp}.{url} = cisco.com

- MIB export:
  mib.{oid} = 1.3.6.1.2.1.2.2.1.1
  mib.{index} = 5

- (Un)observed fields:
  f.{observed} = observed / not available / not applicable

EFSF: conclusion

- EFSF is an orthogonal mechanism to the IPFIX information model.

- Solves several issues:
  - MIB export
  - indexing, hierarchical, and positional elements
  - inter-relationship between elements (min/max/average, first/last, ingress/egress, pre/post)
  - biflow (RFC5103)
  - exporting type (RFC5610)

IPFIX 2

Paul Aitken

87th IETF Meeting, Berlin, 2013