

Network Working Group
INTERNET-DRAFT
Category: Experimental
<draft-dekok-radius-ipfix-00.txt>
Expires: April 5, 2013
8 October 2012

A. DeKok
FreeRADIUS
S. Winter
RESTENA

RADIUS accounting via IPFIX
draft-dekok-radius-ipfix

Abstract

The Remote Authentication Dial In User Server (RADIUS) Protocol provides for a number of simple session-based metrics. There is a need to increase the number of metrics, and to add metrics for individual flows within a session. This document leverages IP Flow Information Export (IPFIX) to address both needs.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 12, 2011

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info/>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Terminology	4
1.2. Requirements Language	4
2. IPFIX in RADIUS	5
2.1. IPFIX-Container Attribute	5
2.2. IPFIX to RADIUS Data Type Mappings	6
2.3. IPFIX ElementID to RADIUS TLV Mapping	7
2.4. Summary of the Mappings	8
3. IPFIX-Container in RADIUS Packets	8
3.1. IPFIX-Container in Access-Accept	8
3.2. IPFIX-Container in Accounting-Request	9
3.3. Administratively Configured Flows	9
4. Caveats	10
5. Examples	10
5.1. Access-Accept	10
5.2. Accounting-Request	11
6. Security Considerations	11
7. IANA Considerations	11
8. References	11
8.1. Normative References	11
8.2. Informative References	12
Appendix A - XML to RADIUS Dictionary Converter	13
Appendix A.1 - Code for XML to RADIUS Dictionary Converter ...	14

1. Introduction

The Remote Authentication Dial In User Server (RADIUS) Protocol defines in [RFC2866] and others metrics for transporting session-based accounting. We wish to extend those capabilities, while having a specification that satisfies the following criteria:

- * follows the RADIUS data model as defined in [RFC6158], and [EXTEN],
- * allows for an increased number of metrics per session,
- * allows for metrics for individual flows within a session,
- * can describe flows in an Access-Accept, for later accounting in an Accounting-Request,
- * requires minimal work from the RADEXT working group to define new metrics

We believe this document satisfies those criteria. We propose to do this by leveraging IP Flow Information Export (IPFIX) entries, defined in [IPFIX]. The benefits of this approach are many. We leverage existing expertise and work, we do not create a new standard to exchange the same information, etc.

1.1. Terminology

This document uses the following terms:

RADIUS client

A device that provides an access service for a user to a network. Also referred to as a Network Access Server, or NAS.

RADIUS server

A device that provides one or more of authentication, authorization, and/or accounting (AAA) services to a NAS.

RADIUS proxy

A RADIUS proxy acts as a RADIUS server to the NAS, and a RADIUS client to the RADIUS server.

1.2. Requirements Language

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 [RFC2119].

2. IPFIX in RADIUS

This section defines how IPFIX information is carried inside of RADIUS attributes.

2.1. IPFIX-Container Attribute

We define a new attribute, called IPFIX-Container. It leverages the "Long Extended" type defined in [EXTEN] Section X.Y. A summary of the Attribute format is shown below. The fields are transmitted from left to right.

0										1										2										3										
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1									
Type										Length										Extended-Type										M	Reserved									
Value ...																																								

Type

TBD - IANA allocation from the "long extended" type space

Length

The Length of the attribute, as previously defined in [RFC2865] Section 5. Permitted values are between 5 and 255. If a client or server receives an IPFIX-Container attribute with a Length of 2, 3, or 4, then that Attribute MUST be considered to be an "invalid attribute", and be handled as per [EXTEN] Section 2.7.

Extended-Type

TBD - IANA allocation from the "long extended" type space

M (More)

The More field is one (1) bit in length, and indicates whether or not the current attribute contains "more" than 251 octets of data. The More field MUST be clear (0) if the Length field has value less than 255. The More field MAY be set (1) if the Length field has value of 255.

If the More field is set (1), it indicates that the Value field has been fragmented across multiple RADIUS attributes. When the More field is set (1), the attribute SHOULD have a Length field of value 255; it MUST NOT have a length Field of value 4; there MUST

be an attribute following this one; and the next attribute MUST have both the same Type and Extended Type. That is, multiple fragments of the same value MUST be in order and MUST be consecutive attributes in the packet, and the last attribute in a packet MUST NOT have the More field set (1).

When the Length field of an attribute has value less than 255, the More field SHOULD be clear (0).

If a client or server receives an attribute fragment with the "More" field set (1), but for which no subsequent fragment can be found, then the fragmented attribute is considered to be an "invalid attribute", and handled as per [EXTEN] Section 2.7.

Reserved

This field is 7 bits long, and is reserved for future use. Implementations MUST set it to zero (0) when encoding an attribute for sending in a packet. The contents SHOULD be ignored on reception.

Value

The value is a series of Type-Length-Values (TLVs), which are the IPFIX information elements as defined in [IPFIX]. One flow MUST be encapsulated in one IPFIX-Container attribute. An IPFIX-Container attribute MUST NOT contain zero, or multiple flows.

2.2. IPFIX to RADIUS Data Type Mappings

The IPFIX specification defines a number of data types. Some can be mapped to RADIUS, others cannot. We document those mappings here. Where a data type is not mapped, IPFIX information elements using that data type MUST NOT appear in a RADIUS packet.

We use the RADIUS data types as defined in [RFC6158]. The following table defines the mappings.

Mapping Table

IPFIX	RADIUS
-----	-----
unsigned8	integer
unsigned16	integer
unsigned32	integer
unsigned64'	integer64
ipv4Address	IP Address
ipv6Address	IPv6 Address
string	text

octetArray	string
dateTimeSeconds	date
dateTimeMilliseconds	integer64
dateTimeMicroseconds	integer64
dateTimeNanoseconds	integer64
macAddress	string
boolean	integer

RADIUS does not permit eight (8) or sixteen (16) bit data types, so IPFIX "integer" types of those sizes are mapped to RADIUS 32-bit integers.

We want to avoid creating new RADIUS data types. Therefore, the IPFIX "dateTime" types are mapped to "integer64". The only exception is "dateTimeSeconds", which maps to the RADIUS "date" data type.

RADIUS does not have a "MAC Address" data type. Therefore the IPFIX "macAddress" data type maps to the RADIUS "string" data type.

2.3. IPFIX ElementID to RADIUS TLV Mapping

The IPFIX ElementIDs are fifteen (15) bits long, while the RADIUS attribute IDs are eight (8) bits. We therefore need a mapping from one to the other. The mapping involves allocating two layers of TLVs, which allows for sixteen (16) bits of mapping. The mapping function is as follows:

```
TLV1 = (ElementId >> 8) & 0xFF
TLV2 = ElementId & 0xFF
```

The resulting RADIUS attribute is then TBD.TLV1.TLV2. (IANA NOTE: Replace TBD with the allocated "long extended" attribute, e.g. 245.1).

We do not allocate or name TBD.TLV1 in the RADIUS attribute space. (IANA NOTE: Replace TBD with the allocated "long extended" attribute, e.g. 245.1). It is simple a container which is never referenced by itself.

We do not provide an exhaustive mapping of which IPFIX ElementID is appropriate (or not) for inclusion in which RADIUS packet. We instead provide general guidelines and recommendations.

IPFIX ElementIDs relating to counters and flow identification are useful in RADIUS. Other ElementIDs are less useful, and SHOULD NOT be used. ElementIDs related to security and packet signing MUST NOT be used in RADIUS.

2.4. Summary of the Mappings

The mapping from IPFIX Information Elements to RADIUS attributes is simple and mechanical. New IPFIX Information Elements can be used by RADIUS without requiring new specifications in the RADIUS space.

A short Perl program is available in Appendix A, below. It provides an example of how the mapping can be done automatically. However, it does not provide a standard of any kind.

3. IPFIX-Container in RADIUS Packets

The IPFIX-Container attribute is permitted in Access-Accept packets, and in Accounting-Request packets. It MUST NOT occur in any other kind of RADIUS packet.

3.1. IPFIX-Container in Access-Accept

The IPFIX-Container can be sent in an Access-Accept packet. This indicates a request from the server that the NAS send Accounting-Request packets containing the requested information. The NAS MAY ignore this request, and send different information, or no information.

The contents of the IPFIX-Container are ElementIDs which describe a particular flow. Multiple IPFIX-Container attributes MAY be sent in an Access-Accept.

The ElementIDs which describe a particular flow are not enumerated here. Instead, we suggest ElementIDs relating to IP addresses, protocols, ports, etc. Where ElementIDs relating to flow identification are omitted, the IPFIX-Container SHOULD be interpreted as applying to all traffic.

The only restriction is that each IPFIX-Container attribute MUST contain a flowID. It is RECOMMENDED that the RADIUS server allocate these flowIDs through a robust process that generates globally and temporally unique values. The method for doing so is not described here.

An Access-Accept MUST NOT contain multiple IPFIX-Container attributes with the same flowID.

This restriction comes about as a result of issues seen with Acct-Session-Id. Deployment experience shows that many NAS implementations choose repeating values for Acct-Session-Id. That repetition makes it difficult for RADIUS servers to track individual user sessions. Allowing for flow-based accounting would make that

problem worse.

This document instead requires that flowIDs are allocated by the RADIUS server. NAS implementations MUST NOT create flowIDs.

3.2. IPFIX-Container in Accounting-Request

The IPFIX-Container can be sent in an Accounting-Request packet. This indicates that the NAS is sending accounting information about a particular flow.

The contents of the IPFIX-Container are ElementIDs which describe a the accounting data for a particular flow. Multiple IPFIX-Container attributes MAY be sent in an Accounting-Request. The IPFIX-Container MAY avoid sending ElementIDs which describe the flow. i.e. ElementIDs which are sent in an Access-Accept. The flowID is generally sufficient to uniquely describe a flow.

However, in situations where the flowIDs are administratively configured, those ElementIDs which describe the flow MUST be included in the IPFIX-Container attribute. This inclusion ensures that the combination of flowID and flow description is unique.

The ElementIDs which describe accounting for a particular flow are not enumerated here. Instead, we suggest ElementIDs relating to packet counting, octet counting, time, etc.

The only restriction is that each IPFIX-Container attribute MUST contain a flowID. The NAS MUST use flowIDs taken from a corresponding Access-Accept packet, or flowIDs which have been administratively configured. It MUST NOT generate flowIDs by itself.

An Accounting-Request packet MUST NOT contain multiple IPFIX-Container attributes with the same flowID.

3.3. Administratively Configured Flows

A NAS MAY permit an administrator to manually configure flows. In that case, it can send flows in an Accounting-Request packet which were not requested in an Accounting-Accept packet. In that case, ElementIDs which describe the flow MUST be included in the IPFIX-Container attribute. The flowID MUST also be configured by the administrator, and not created by the NAS.

For administratively configured flows, a NAS MAY give the administrator the choice of using incrementing flowIDs for each flow in a session, starting from the value one (1).

4. Caveats

The flows in IPFIX are unidirectional. RADIUS uses different attributes to count data for each direction. As a result, where RADIUS would use two attributes Acct-Input-Octets and Acct-Output-Octets, we would require two IPFIX-Container attributes.

Not all of the IPFIX data types are mapped. We have no suggestions for what to do with the more complex data types.

The IPFIX "basicList" data type appears to be unnecessary in RADIUS. Instead of encapsulating multiple ports in a basicList, we can simply send multiple port descriptions in one IPFIX-Container attribute.

Mapping eight and sixteen-bit IPFIX integers to a 32-bit RADIUS integer type is suboptimal. However, it meets the requirements of [RFC6158]. The major benefit of reusing the RADIUS data model outweighs the minor benefit of a slightly more efficient packing.

IPFIX allows for sending data up to 64K in size. The multiple encapsulations define here allows for only a maximum of 249 octets. This limitation does not matter for the bulk of the ElementIDs which are relevant to RADIUS. For the rest, we offer no suggestion other than to administratively limit strings to a particular length.

Not all implementations will support this specification. Therefore, the existing RADIUS accounting attributes SHOULD be sent in addition to the IPFIX-Container attribute. Sending only IPFIX-Container is NOT RECOMMENDED.

5. Examples

This section contains examples. We use names for the attributes taken from the output of the Perl program in Appendix A. We show only the IPFIX-Container attribute and the TLVs related to IPFIX ElementIDs. We do not show intermediate TLVs.

5.1. Access-Accept

```
IPFIX-Container
  IPFIX-flowId = 1
  IPFIX-flowDirection = 0x00
IPFIX-Container
  IPFIX-flowId = 2
  IPFIX-flowDirection = 0x01
```

5.2. Accounting-Request

```
Acct-Input-Octets = 1024
Acct-Output-Octets = 2048
Acct-Input-Packets = 10
Acct-Output-Packets = 20
IPFIX-Container
  IPFIX-flowId = 1
  IPFIX-flowDirection = 0x00
  IPFIX-octetTotalCount = 1024
  IPFIX-packetTotalCount = 10
IPFIX-Container
  IPFIX-flowId = 2
  IPFIX-flowDirection = 0x01
  IPFIX-octetTotalCount = 2048
  IPFIX-packetTotalCount = 20
```

6. Security Considerations

This document defines new RADIUS attributes, but makes no changes to the RADIUS protocol. As such, there are no security considerations.

The IPFIX protocol defines security related ElementIDs such as messageMD5Checksum. These ElementIDs MUST NOT appear in a RADIUS packet. Existing RADIUS security is sufficient to protect the packets.

7. IANA Considerations

This document requests the allocation of a new attribute, "IPFIX-Container", from the "long extended" RADIUS attribute type space.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [IPFIX] <http://www.iana.org/assignments/ipfix/>
- [EXTEN] DeKok, A, Lior, A, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", draft-ietf-radext-radius-

extensions-05.txt, (work in progress)

8.2. Informative References

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

[RFC6158] DeKok, A., and Weber, G., "RADIUS Design Guidelines", RFC 6158, March 2011.

Acknowledgments

The initial suggestion to leverage IPFIX was from Benoit Claise.

Appendix A - XML to RADIUS Dictionary Converter

The following page contains a simple Perl script which converts the [IPFIX] XML registry to a RADIUS dictionary file. It is intended as a guide to show that the conversion is mechanical. It does not specify a standard, and implementors are free to use their own method for converting the IPFIX registry to a RADIUS dictionary.

Appendix A.1 - Code for XML to RADIUS Dictionary Converter

```
#!/usr/bin/env perl
use XML::Simple;
use Data::Dumper;

$prefix = "TBD";          # IANA - to be updated

print "ATTRIBUTE IPFIX-Container $prefix long-extended\n";

$map = {
    'unsigned8'    => 'integer',
    'unsigned16'   => 'integer',
    'unsigned32'   => 'integer',
    'unsigned64'   => 'integer64',
    'ipv4Address'  => 'ipaddr',
    'ipv6Address'  => 'ipv6addr',
    'string'       => 'text',
    'octetArray'   => 'string',
    'dateTimeSeconds' => 'date',
    'dateTimeMilliseconds' => 'integer64',
    'dateTimeMicroseconds' => 'integer64',
    'dateTimeNanoseconds' => 'integer64',
    'macAddress'   => 'string',
    'boolean'      => 'integer',
};

$xml = new XML::Simple;
$data = $xml->XMLin("ipfix.xml");
$elements = $data->{registry}->{'ipfix-information-elements'}->{record};

foreach $record (@{$elements}) {
    next if $record->{unassigned};
    next if $record->{reserved};

    $name = $record->{name};
    $name =~ s/\s+//g;

    $supper = 1 + (($record->{elementId} & ~0xff) >> 8);
    $lower = $record->{elementId} & 0xff;
    $oid = "$prefix.$supper.$lower";

    $type = $record->{dataType};
    if (!defined $map->{$type}) {
        print "# ATTRIBUTE IPfix-$name $oid ", $type, "\n";
    } else {
        print "ATTRIBUTE IPfix-$name $oid ", $map->{$type}, "\n";
    }
}
```

}

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project
<http://freeradius.org/>

Email: aland@freeradius.org

Stefan Winter
Fondation RESTENA
6, rue Richard Coudenhove-Kalergi
Luxembourg 1359
Luxembourg

Phone: +352 424409 1
Fax: +352 422473
EMail: stefan.winter@restena.lu
URI: <http://www.restena.lu>

