W-IPFIX Implementation

Workstation based IPFIX

User’s Manual

Authors: João Olival, Rui Valadas, Paulo Salvador, António Nogueira

Affiliation: University of Aveiro / Institute of Telecommunications Aveiro
   {jolival, salvador, nogueira}@av.it.pt; rv@det.ua.pt

Last updated: March 6, 2006
1 Introduction

Flow measurements in the Internet are becoming increasingly important for a number of applications, including traffic engineering, network planning, accounting and attack detection.

Monitoring of flows typically involves two separate entities: a meter and a collector. The former may be a sniffing probe or dedicated HW/SW in the routing equipment, while the collector is the recipient of the measurement data, transforming and/or storing it before it is processed by an application.

Currently, a variety of IP flow export systems are used in the practice, making it difficult to achieve an acceptable level of interoperability among tools and hindering the development of generalized flow analysis tools.

Exporting network traffic information from a probe and viewing the statistics on a per-flow basis gives network managers information they can use to make key decisions. Administrators who know how many packets and bytes are sent to and from certain IP addresses or across specific network interfaces can create usage-based departmental charge-back systems. They also can use the information to traffic-engineer their networks for optimum performance.

IPFIX defines the format by which IP flow information can be transferred from a meter to a collector. Applications that support IPFIX will understand and display statistics received from any meter that also supports the standard.

W-IPFIX is a workstation-based implementation of the Internet Protocol Flow Information eXport (IPFIX) extended with a centralized control platform. IPFIX has proposed by an IETF Working Group (IPFIX) as an effort to standardize the flow export process.

The IPFIX collecting process (server) should be able to receive the flow information passing through multiple network elements (probes) within the data network.

![Figure 1 - Generic connections between server and "n" probes.](image-url)
2 Extended control platform

The W-IPFIX has an optional probe control mechanism to the standard IPFIX, maintaining the compatibility with it.

The probe is the device (client) that will establish a TCP connection with server. For each probe connected to server, it creates two sessions (figure 2). Probe sends the IPFIX data messages through Session_2 and exchanges control messages with server through Session_1.

IPFIX protocol only supports the use of Session_2, to send data to collecting process. Session_1 is a W-IPFIX extension. It is beyond IPFIX protocol, because allows the devices to exchange control messages, as described ahead in section 3. W-IPFIX is a result of putting all together, IPFIX protocol along with both sessions.

![Figure 2 - Connection detail between server and probe.](image)

The observation point, metering process and exporting process, constitutes the probe device.

The observation point is a location in the network to observe IP packets. The metering process generates flow records. Inputs to the process are packet headers observed at an observation point and packet treatment at the observation point, for example the selected output interface.

A flow is defined as a set of IP packets passing an observation point in the network during a certain time interval. The metering process consists of a set of functions that includes packet header capturing, timestamping, sampling, classifying, and maintaining flow records.

![Figure 3 - Probe representation.](image)

The maintenance of flow records may include creating new records, updating existing ones, computing flow statistics, deriving further flow properties, detecting flow expiration, passing flow records to the exporting process, and deleting flow records.
Figure 4 - Functions of the metering process.

The exporting process sends flow records to the collecting process. The collecting process receives flow records from one or more exporting processes and might store received flow records or further process them.

Each terminal where is executed the application probe.exe (http://www.av.it.pt/wipfix/index_files/probe.exe) must have installed the program WinPcap (http://www.winpcap.org/install/bin/WinPcap_3_1.exe). For obvious reasons, any client must have connectivity with the server, through the network.

3 W-IPFIX Application

3.1 W-IPFIX Description

3.1.1 General Description

The application creates two TCP sessions, to exchange information (data and control/configuration messages) between server and probe, as described previously. Server uses one thread (getClient) to accept a probe.

Client waits for the configuration message while the server receives in its port one acknowledge, indicating that the probe is active and it is ready to receive a control message (its configuration determines when to start and stop captures and which packets to save).

Server application initiates a thread to verify if a probe stops sending IPFIX messages for a certain period (maximum time that server waits without receiving data). Reaching that period without receiving data from probe, server sends a request to client. If the client is inactive, the server resets the connection, closing the sessions, otherwise, reestablishes it.
Once the two sessions are created and after user sets its preferences, server sends a control message to probe, where is created a thread (threadToCaptureExpiredFlow) to detect the expired flows (flows reaching expiration time) and launching the function callback to capture packets.

The filter definition for capturing packets can be set as a promiscuous capture or according to the filter commands in TCPDump and can be set as following:

- IP Address (source and destination)
- Ports (source and destination)
- Protocol
- Host name
- Any combination of the previous

If a flow record is not updated during a configurable period, the referred thread “threadToCaptureExpiredFlow” detects that fact, subtracting the current time by the time when the last packet was captured, and moves its element to the list of expired flows. It is immediately emptied if the list of expired flows is full, moving its information to a buffer and sent to server.

Concerning the probe, its metering process must be able to detect flow expirations. A flow is considered to be expired if no packet of this flow has been observed for a given timeout interval. The metering process may support means for detecting the expiration of a flow before a timeout occurs, for example by detecting the FIN or RST bits in a TCP connection.

The probe adds every new flow to an existent list of flows. When an element of this list expires, it is moved to the list of expired flows. This list is limited to a certain number of elements, removing it, when complete and placing in a buffer, in accordance with the IPFIX message format, preparing to send to server.

When a new flow record appears, function AddFlow initiates the current time in a variable and adds its information to the list of flows. If the probe captures a packet belonging to a flow that exists in the flow list, a function (UpdateList) updates the time variable, number of packets, bytes, etc…

To export data, probes represent each network traffic flow based on seven key fields:

- Source IP address
- Destination IP address
- Source port
- Destination port
- Layer 3 protocol type
- Type-of-service byte
- Input logical interface
If all seven key fields in two different packets match, both packets are designated as belonging to the same flow. Packets in that flow are compared against the same match criteria and counted.

### 3.1.2 IPFIX Message Format

The information sent from probe to server has the IPFIX message format, as described:

- *Message header*
- One or more {template, option template, data} sets.

The implemented module built the IPFIX message with the *Message Header, Template Set* and *Data Set*. A buffer stores these elements to send it to the Collecting Process.

The definition of *Template Set* agrees with IPFIX norm, where it is possible to see its parameters:

![Figure 5 - Message Header structure.](image)

![Figure 6 – Template Set Structure of IPFIX message.](image)

### 3.2 Practical Example

As an example, to begin a capture from terminal ‘A’ (193.136.92.228), sending its data to terminal ‘B’ (193.136.92.191), the probe application is launched in terminal ‘A’ and server in terminal ‘B’.
3.2.1 W-IPFIX Configuration Text Files

Before executing the applications (server and probe), there is an associated configuration text file with each application: `COLLECT_CONFIG.txt` (http://www.av.it.pt/wipfix/index_files/COLLECT_CONFIG.txt) and `PROBE_CONFIG.txt` (http://www.av.it.pt/wipfix/index_files/PROBE_CONFIG.txt), where it is possible to configure some important parameters. The configuration text file of probe has the following information:

```
IP 193.136.92.191
PORT0 17388
PORT1 17364
TCP_FLAGS 0
// IP->IP of the server
// PORT0->port of the server to listen IPFIX message
// PORT1->port of the server to listen CTRL message
// TCP_FLAGS->checks RST or FIN bits in TCP connection, if active (1->active, 0->disable)
```

Figure 7 - Configuration text file of probe.

It’s possible to modify the values of the ports, through which the server is to listen (data messages – `PORT0` and control messages – `PORT1`) and its `IP` address.

The metering process must be able to detect flow expirations. The metering process detects `FIN` and `RST` bits, when the flag `TCP_FLAG` is active, as shown in figure 5.

For the server, the configuration text file will have the following information:

```
PORT0 17388
PORT1 17364
EXPIRATION_TIME 10
WAITING_TIME 45
// PORT0->port that server uses to listen IPFIX message from probe
// PORT1->port that server uses to listen CTRL message from probe
// EXPIRATION TIME->flow record expiration time (in seconds)
// WAITING TIME->maximum time (in seconds) waiting for IPFIX messages
```

Figure 8 - Configuration file of the server.

Here, the configuration text file allows configuring:

- Ports (MUST be the same ones inserted previously in the configuration text file `PROBE_CONFIG.txt`) through which the server listens.
- Expiration time of the flow (defined for 15 seconds in the example).
- Maximum time that server waits without receiving data (defined for 45 seconds in the example).
3.2.2 Running W-IPFIX

After defining the parameters in each one of the configuration text files, it is the execution of both applications: server.exe (http://www.av.it.pt/wipfix/index_files/collect.exe) and probe.exe.

3.2.2.1 User Interface

Probe shows in the console a list of the available devices to capture and communicate with server, for user selection:

![Figure 9 - Probe console, with the list of the available devices.](image)

While server waits for a new probe connection, it presents in the console the following information:

![Figure 10 - Server console, waiting for a probe connection.](image)

The application creates two TCP sessions, to exchange information (data and control messages) between server and probe, as described previously. Server uses one thread (getClient) to accept probe. User selects the device in the terminal where is the probe:

![Figure 11 - Pobe console, waiting for the control message.](image)

Client waits for the configuration message while the server receives in its port (17353 in the example) one acknowledge, indicating that the probe is active and it is ready to define the filter, beginning and ending measure times, to send to probe.
Following the console menu, the application server.exe waits the user to proceed with the configuration, introducing ‘1’ to continue:

A new menu shows up, after choosing the desired device, where user can select:
- Filter definition for capture
- Promiscuous capture
- Closing sessions
Figure 16 gives an example of how to define a filter:

![Figure 17 – Server console, with filter definition interface.](image)

The filter commands are accordingly to `TCPDump`. After defining the filter, it is possible to configure the beginning and ending captures. It can begin measuring freely or accordingly to a specified date and time until a certain time or indefinitely.

![Figure 18 – Server console, with time interface.](image)

‘Date’ refers to ‘date/hour’, as a simplification in this manual. In addition, the application assume the current year, accessing to the operating system, which is why the user is not ask by the year.

The following shows the time interface to begin and end measures:

- To just begin measuring during an uncertain time, must insert ‘33’, in option ‘Month >’.
- To begin immediately with a desired ending time, must insert ‘44’, in option ‘Month >’.
- To define the beginning date, must insert month, day, hour and minutes.
Imagining that user chooses an immediately beginning time, with a desired ending time:

![Command Prompt - collect.exe](image)

**Figure 19 - Consola do server, com a interface para escolha da data de fim de medição.**

Inserting ‘44’, the server application shows the time interface, asking the user whether he wants to define an ending time (choosing the month, day, hour and minutes) or an undefined ending time (‘33’).

![Command Prompt - collect.exe](image)

**Figure 20 – Server console, with the desired configuration.**

The server application makes the beginning and ending time validation, sending an error message to the console if the values are incorrect: ‘Bad Parameters’.

After setting the ending measure, the console shows its information, as the values defined at the configuration text file, presented previously in this manual.
Finally, a control message is sent to the probe device. The program WinPcap captures the packets and the application saves only the ones that agrees with the specified filter. In the example, the capture begins immediately and finishes at 18:09 of 2006/01/11.

![Figure 21 – Probe console, preparing to capture.](image)

Server will continue listening to other possible probes, whose could try to connect and while simultaneously receives IPFIX messages from the connected probe.
3.2.2.2 Console Messages

Not updating a flow record during its 15 seconds (as defined in the configuration text file), the thread `threadToCaptureExpiredFlow` detects that fact, subtracting the current time by the time when the last packet was captured and moves its element to the list of expired flows. It is immediately emptied if the list of expired flows is full, moving its information to a buffer and sent to server.

Server has a thread to detect if probe reaches a certain time (45 seconds in the example), without sending data to server. When that fact occurs:

![Command Prompt - collect.exe](image)

Figure 22 – Server console, informing that sockets with handle 4393 have closed.
Not reaching that time, probe finishes as defined previously:

![Probe console, has finished sending messages to server.](image)

**Figure 23** – Probe console, has finished sending messages to server.
4 Results

Server creates two XML files, storing data received from probe. As the example, the files ‘3546.xml’ and ‘CollectLog.xml’ exists.

Figure 25 shows the structure of file ‘4551.xml’:

```
- <root>
  <Log_from_Probe>193.136.92.228</Log_from_Probe>
  - <info>
    <sourceIPv4Address>193.136.092.165</sourceIPv4Address>
    <destinationIPv4Address>193.136.092.255</destinationIPv4Address>
    <inPacketTotalCount>0000001</inPacketTotalCount>
    <inOctetTotalCount>0000229</inOctetTotalCount>
    <flowEndTime>0010</flowEndTime>
  </info>
  - <info>
    <sourceIPv4Address>193.136.092.183</sourceIPv4Address>
    <destinationIPv4Address>228.001.002.003</destinationIPv4Address>
    <inPacketTotalCount>0000001</inPacketTotalCount>
    <inOctetTotalCount>0000151</inOctetTotalCount>
    <flowEndTime>0010</flowEndTime>
  </info>
  - <info>
    <sourceIPv4Address>193.136.093.147</sourceIPv4Address>
    <destinationIPv4Address>193.136.093.255</destinationIPv4Address>
    <inPacketTotalCount>0000003</inPacketTotalCount>
    <inOctetTotalCount>0000234</inOctetTotalCount>
    <flowEndTime>0012</flowEndTime>
  </info>
```

Figure 24 – XML file “4551.xml”.

Analyzing figure 25, it has information about three flows, with 12 seconds maximum expiration time (flowEndTime). In the first flow, one packet came from IP Address 193.136.92.165 to 193.136.92.255, with size of 229 octets. Each new capture session of a probe creates a file like this one described.
Figure 26 shows the file ‘CollectLog.xml’, where the first capture began at 18:03 of 11/01/2006 and ended at 18:05 of 11/01/2006, from terminal 193.136.92.228, with socket handle 4393, and sent to server ‘zero’ IPFIX messages. The second capture began at 18.07 of 11/01/2006, ended at 18.09 of 11/01/2006, also from terminal 193.136.92.228, with socket handle 4551, and sent to server 14 IPFIX messages.

For each new capture, server updates this file, as an historic log file:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <root>
    - <collecting>
      <date_begin>Probe began capturing at: 18:03 [11/01/2006]
        <ip>193.136.92.228</ip>
        <handle>4393</handle>
        <msgs>0</msgs>
      </date_begin>
      <date_end>Probe finished capturing at: 18:05 [11/01/2006]
        <ip>193.136.92.228</ip>
        <handle>4393</handle>
        <msgs>0</msgs>
      </date_end>
    </collecting>
    - <collecting>
      <date_begin>Probe began capturing at: 18:07 [11/01/2006]
        <ip>193.136.92.228</ip>
        <handle>4551</handle>
        <msgs>14</msgs>
      </date_begin>
      <date_end>Probe finished capturing at: 18:09 [11/01/2006]
        <ip>193.136.92.228</ip>
        <handle>4551</handle>
        <msgs>14</msgs>
      </date_end>
    </collecting>
  </root>
```

Figure 25 – XML file “Collect.xml”.

The XML files can be viewed as a user-friendly way (sorted in columns) if opened, for example, with Microsoft Excel, selecting it to open as an ‘XML list’.

```
Open XML
```

Figure 26 - Opening a xml file in Excel to view information in columns.
Figure 28 shows the XML file ‘3546.xml’ openned as an ‘XML list’:

![Figure 27 - Xml file openned as a xml list in Excel.](image-url)
5 References

- “WinPcap”, http://www.winpcap.org/
- “TCPdump”, http://www.tcpdump.org/
- “Network Sorcery”, http://www.networksorcery.com/