DOTS
INTERNET-DRAFT
Intended Status: Informational
Expires: September 14, 2017

T. Fu
C. Zhou
H. Zheng
Huawei

March 13, 2017

IP Flow Information Export (IPFIX) Information Elements Extension for TCP Connection Tracking

draft-fu-dots-ipfix-tcp-tracking-00

Abstract

This document proposes several new TCP connection related Information Elements (IEs) for the IP Flow Information Export (IPFIX) protocol. The new Information Elements can be used to export certain characteristics regarding a TCP connection. Through massive gathering of such characteristics, it can help build an image of the TCP traffics passing through a network. The image will facilitate the detection of anomaly TCP traffic, especially attacks targeting at TCP.

Status of this Memo

Copyright and License Notice

Table of Contents

1. Introduction

Due to its complex stateful operations, TCP [RFC0793] is especially vulnerable to attacks. The SYN Flood attack is an example, it involves with massive malicious clients attempting to set up connections with a server, but never completing the three-way handshake process, leaving the server-side of the connections in waiting states, eventually exhausting the server resources and no new connection can be created.

Attack aiming at TCP can be low and slow in traffic pattern. Sometimes it may not take down the server, but just impair the provided service. Even though a victim server is still operating, its performance can be significantly degraded. Without the insight of what is going on with the TCP traffics, this kind of situation can be very hard to detect and analyze.

For a network device, such as a router, to detect anomaly TCP traffics, it has to understand the semantics of TCP operations, more specifically, it has to be able to track TCP connection states. If a router has implemented such ability, it can export characteristics information regarding the TCP connections. By this way, offline analysis can be performed over the gathered information, which will facilitate the detection of anomaly TCP traffics, such as attacks.

The IP Flow Information Export (IPFIX) protocol [RFC7011], already defines a generic mechanism for flow information export. This document introduces several new Information Elements of IPFIX, that can be used to export TCP connection characteristics. The proposed Information Elements are listed in Figure 1 below.

Field Name	IANA IPFIX ID
tcpHandshakeSyn2SynAckTime	TBD
tcpHandshakeSynAck2AckTime	TBD
tcpHandshakeSyn2AckRttTime	TBD
tcpConnectionTrackingBits	TBD
tcpPacketIntervalAverage	TBD
tcpPacketIntervalVariance	TBD
tcpOutOfOrderDeltaCount	TBD

Figure 1: Information Element Table

The Information Elements defined in Figure 1 are supposed to be incorporated into the IANA IPFIX Information Elements registry [IPFIX-IANA]. Their definitions can be found at Section 6.

2. 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 [RFC2119]

2.1. Terminology

IPFIX-specific terminology (Information Element, Template, Template Record, Options Template Record, Template Set, Collector, Exporter, Data Record, etc.) used in this document is defined in Section 2 of [RFC7011]. As in [RFC7011], these IPFIX-specific terms have the first letter of a word capitalized.

This document also makes use of the same terminology and definitions as Section 2 of [RFC5470].

o Victim

The target that suffers from DDoS attack.

3. Connection Sampling and new IEs

3.1. Use Cases for New IEs

In this section, several use cases are discussed to identify the requirements where new IEs are desirable for the network attacks detection.

3.1.1. Response Time Calculation

For other DDoS attacks such as Http slowloris, there will be too many connections that should be kept in the victim (server), which lead to excessive resource consumption. As a result, the response time between client and server will increase greatly. Challenge Collapasar(CC) attack can also exhaust the resources of the server and generate the similar results. Thus, the following IEs are proposed as a symptom of these kinds of attacks:

tcpHandshakeSyn2SynAckTime: it denotes the time difference between the time point that the Metering Process detects the SYN packet from client to server and the time point that the observer views the SYN-ACK packet from server to client.

tcpHandshakeSynAck2AckTime: it denotes the time difference between the time point that the Metering Process detects the SYN-ACK packet from server to client and the time point that the observer views the ACK packet from client to server.

tcpHandshakeSyn2AckRttTime: it denotes The sum of tcpHandshakeSyn2SynAckTime and tcpHandshakeSynAck2AckTime. It is the Round Trip Time (RTT) between client and server.

3.1.2. Symptoms of Exceptions

In http slowloris attack the client may send packets to victim periodically which can cause the performance lost on the server. The characteristic of the attack is that there are too many connections on the victim. However, the traffic volume for these connections is small. In order to detect this attack, the first step is to get the packets that are belonging to the same connection. The second step is to find the periodicity. Thus the two indices tcpPacketIntervalAverage and tcpPacketIntervalVariance are needed. The index tcpPacketIntervalAverage denotes the average time difference between two successive packets and the index tcpPacketIntervalVariance denotes the variance of multiple time difference. Large tcpPacketIntervalAverage and small tcpPacketIntervalVariance can be a symptom of slow packet attack, since the attacker sends packets in large intervals just as to keep the connection open, and the intervals tend to differ very little in time.

To degrade the performance of the victim, the malicious clients may send too many out-of-order packets, which will consume too much memory on the server. Although out-of-order packets are permit in the TCP protocol, it is possible to be leveraged to cause DDoS attack. So the index tcpOutOfOrderDeltaCount is helpful to detect this kind of exception. For observer, it maintains one counter for each TCP connection. The initial sequence number of the client is saved in the counter. The counter increases by the sequence number of the packets it sees from client to server. If the observer sees a packet with lower sequence number than the current counter value, then the packet will be considered as an out-of-order packet.

In IPFIX, the index tcpControlBits is used to record the corresponding status bits in TCP header of the packets[IPFIX-IANA]. In order to detect the application attacks which can cause the protocol exception such as the wrong use of the TCP status bits before and after the TCP connection establishment, another index called tcpConnectionTrackingBits is needed. For example, when the observer sees the SYN packet from client to server, it sets 15th bit of tcpConnectionTrackingBits to 1; when it sees the SYN-ACK packet from server to client, it sets 14th bit to 1, and so on. If one endpoint sends the packet with wrong bits during the establishment of the connection, then the observer will identify the exception by the value of tcpConnectionTrackingBits.

4. Application of the New IEs for Attack Detection

This section presents a number of examples to help for the easy understanding of the application of these new IEs for attack detection.

4.1. Detect Slowloris Attack

The template for detecting resource exhausting application attack such as http slowloris attack should contain a subnet of IEs shown in Table 4.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Set ID = 2            |      Length = 48 octets       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Template ID TBD         |       Field Count = 10        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    sourceIPv4Address        |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| destinationIPv4Address      |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| protocolIdentifier          |       Field Length = 1        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| tcpHandshakeSyn2SynAckTime  |       Field Length = 2        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| tcpHandshakeSynAck2AckTime  |       Field Length = 2        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| tcpHandshakeSyn2SynAckTime  |       Field Length = 2        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| tcpPacketIntervalAverage    |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| tcpPacketIntervalVariance   |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|      flowStartSeconds       |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|      flowEndSeconds         |       Field Length = 4        | +-
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Template example for detecting slowloris attack

An example of the actual record is shown below in a readable form as below:

{sourceIPv4Address = 192.168.0.101, destinationIPv4Address = 192.168.0.201, protocolIdentifier = 6, tcpHandshakeSyn2SynAckTime = 200, tcpHandshakeSynAck2AckTime = 10, tcpHandshakeSyn2AckRttTime = 210, tcpPacketIntervalAverage = 500, tcpPacketIntervalVariance = 1000, flowStartSeconds = 100, flowEndSeconds = 200}

4.2. Detect Out-of-order Packets Attack

The template for detecting out-of-order packets attack should contain IEs shown in Table 5.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Set ID = 2            |      Length = 32 octets       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Template ID TBD         |       Field Count = 10        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    sourceIPv4Address        |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| destinationIPv4Address      |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|      protocolIdentifier     |       Field Length = 1        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|      packetDeltaCount       |       Field Length = 8        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   tcpOutOfOrderDeltaCount   |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|      flowStartSeconds       |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|      flowEndSeconds         |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Template example for detecting out-of-order attack

An example of the actual record is shown below in a readable form as below:

{sourceIPv4Address = 192.168.0.101, destinationIPv4Address = 192.168.0.201, protocolIdentifier = 6, packetDeltaCount =3000, tcpOutOfOrderDeltaCount = 2000, flowStartSeconds = 100, flowEndSeconds = 200}

5. Security Considerations

No additional security considerations are introduced in this document. The same security considerations as for the IPFIX protocol [RFC7011] apply.

6. IANA Considerations

The following information elements are requested from IANA IPFIX registry. Upon acceptation, the 'TBD' values of the ElementIds should be replaced by IANA for assigned numbers.

Name: tcpHandshakeSyn2SynAckTime
Description:
The time difference between a SYN and its corresponding SYN-ACK when the Metering Process observes a new TCP connection is going to be set up.
Abstract Data Type: dateTimeMicroseconds
ElementId: TBD
Status: current
Units: microseconds

Name: tcpHandshakeSynAck2AckTime
Description:
The time difference between a SYN-ACK and its corresponding ACK when the Metering Process observes a new TCP connection is going to be set up.
Abstract Data Type: dateTimeMicroseconds
ElementId: TBD
Status: current
Units: microseconds

Name: tcpHandshakeSyn2AckRttTime
Description:
The time difference between a SYN and its corresponding ACK sent from the same endpoint when the Metering Process observes a new TCP connection is going to be set up.

Conceptually tcpHandshakeSyn2AckRttTime can be thought as the sum of tcpHandshakeSyn2SynAckTime and tcpHandshakeSynAck2AckTime, but practically the values may differ.
Abstract Data Type: dateTimeMicroseconds
ElementId: TBD
Status: current
Units: microseconds

Name: tcpConnectionTrackingBits
Description:
These bits are used by the Metering Process to track a TCP connection. A bit is set to 1 if the corresponding condition is met. A value of 0 for a bit indicates the corresponding condition was net met.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|1|1|1|1|1|0|0|0|0|0|0|0|0|0|0|
|5|4|3|2|1|0|9|8|7|6|5|4|3|2|1|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|S|A|F|A|F|A|R|T|E|END|R|R|E|V|
|Y|/|C|I|C|/|C|S|M|N|REA|O|O|R|L|
|N|A|K|N|K|A|K|T|R|D|SON|P|D|R|D|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Bit 15 (SYN):
Set when there is no TCP connection between the endpoints and the Metering Process detects a SYN as it is used to setup a new TCP connection. The Metering Process starts to track the TCP connection.

Bit 14 (S/A):
Set when bit 15 has been set and the Metering Process detects a SYN-ACK in the flow, which effectively acknowledges the SYN causing bit 15 to be set.

Bit 13 (ACK):
Set when bit 15 and bit 14 have been set and the Metering Process detects an ACK which effectively acknowledges the SYN causing bit 14 to be set. Upon setting this bit, it means handshake of the TCP connection setup has completed.

Bit 12 (FIN):
Set when the Metering Process detects the first FIN for the established and tracked TCP connection. It means the TCP connection is going to be closed.

Bit 11 (ACK):
Set when bit 12 has been set and the Metering Process detects an ACK which effectively acknowledges the FIN causing bit 12 to be set.

Bit 10 (F/A):
Set when bit 12 has been set and the Metering Process detects a FIN that is from the opposite of the endpoint which sent the FIN causing bit 12 to be set.

Bit 09 (ACK):
Set when bit 10 has been set and the Metering Process detects an ACK that is from the same endpoint which sent the FIN causing bit 10 to be set.

Bit 08 (RST):
Set when the Metering Process detects any RST from either party of the tracked TCP connection.

Bit 07 (TMR):
Set when a flow record report is triggered by a periodic reporting timer. It means the TCP connection is still under tracking.

Bit 06 (END):
Set when the Metering Process has stopped tracking the TCP

connection, as the connection has been closed or aborted.
Bit 05 & Bit 04 (END REASON):
00: as default value or the tracked TCP connection is closed.
01: the tracked TCP connection is aborted.
10: the tracked TCP connection is inactive after a period of time.
11: reserved.
Bit 03 (ROP):
Set when the Metering Process detects any SYN or SYNACK, after the both endpoints have sent FIN or an RST has been detected.
Bit 02 (ROD):
Set when the Metering Process detects at least 50 TCP segments being exchanged, after both endpoints have sent FIN or an RST has been detected.
Bit 01 (ERR):
Set when the Metering Process detects any of the following abnormal signaling sequences for the TCP connection: SYN/FIN, SYN/FIN/PSH, SYN/FIN/RST, SYN/FIN/RST/PSH.
Bit 00 (VLD):
Set when the tracked TCP connection is closed normally.

Abstract Data Type: unsigned16
Data Type Semantics: flags
ElementId: TBD
Status: current

Name: tcpPacketIntervalAverage
Description:
The average time interval calculated by the Metering Process between two successive packets in the data flow of a TCP connection.
Abstract Data Type: unsigned32
ElementId: TBD
Status: current

Name: tcpPacketIntervalVariance
Description:
The variance of the time intervals calculated by the Metering Process between two successive packets in the data flow of a TCP connection.
Abstract Data Type: unsigned64
ElementId: TBD
Status: current

Name: tcpOutOfOrderDeltaCount
Description:
The number of out of order packets in the data flow of a TCP connection detected at the Observation Point since the previous report.
Abstract Data Type: unsigned64
Data Type Semantics: deltaCounter
ElementId: TBD
Status: current

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7011] Claise, B., Trammell, B., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, September 2013.

[RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.

[RFC0793] J. Postel, "Transmission Control Protocol", STD 7, RFC 793, September 1981.

7.2. Informative References

[IPFIX-IANA]
IANA, "IPFIX Information Elements registry", <http://www.iana.org/assignments/ipfix>.

[RFC5474] Duffield, N., Ed., Chiou, D., Claise, B., Greenberg, A., Grossglauser, M., and J. Rexford, "A Framework for Packet Selection and Reporting", RFC 5474, March 2009.

[RFC5475] Zseby, T., Molina, M., Duffield, N., Niccolini, S., and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection", RFC 5475, March 2009.

[RFC5476] Claise, B., Ed., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 2009.

[RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, March 2009,

8. Acknowledgments

The authors would like to acknowledge the following people, for their contributions on this text: DaCheng Zhang, Bo Zhang (Alex), Min Li.

Authors' Addresses

Tianfu Fu
Huawei
Q11, Huanbao Yuan, 156 Beiqing Road, Haidian District
Beijing 100095
China

Email: futianfu@huawei.com

Chong Zhou
Huawei

156 Beiqing Road, M06 Shichuang Technology Demonstration Park
Haidian, Beijing 100094
China

Email: mr.zhouchong@huawei.com

Hui Zheng (Marvin)
Huawei

101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: marvin.zhenghui@huawei.com