**Suricata - Feature #1239**

**Best effort TCP stack**

07/16/2014 06:41 AM - JmpCallPoo

<table>
<thead>
<tr>
<th>Status:</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>OISF Dev</td>
</tr>
<tr>
<td>Category:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td>TBD</td>
</tr>
<tr>
<td>Effort:</td>
<td></td>
</tr>
<tr>
<td>Difficulty:</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

Hello,

Is it possible to have a best effort TCP stack in suricata?

The problem is when a single packet is lost, all the stream is broken and in the context of a keep-alived HTTP stream, this is not so cool :(

Thank you.

**History**

**#1 - 07/16/2014 06:58 AM - Victor Julien**

- **Subject changed from Best effort TCP stack. to Best effort TCP stack**

This is not really a TCP stream engine issue, as we do continue inspection of 'raw' stream data in case of packet loss. However, the app layer parsers like HTTP are not able to handle missing data. It will be very hard to adapt those to handle missing data. In some cases it may be possible, for example in case of missing HTTP Body data, where we know the length of the body because of the content length. But in other cases missing data would mean we'd have to figure out what the missing data would mean. If we miss 16k of HTTP data to the server, did we miss a single large request? Or multiple smaller requests? I think this likely brings more problems that solutions.

**#2 - 07/16/2014 07:10 AM - JmpCallPoo**

I agree that this will be a little bit tricky with HTTP pipeline!

Maybe, it is possible to set the stream as "incertain/insecure" and continue to pass the stream to the HTTP app layer. But in most of the cases, user will use single request, one by one, and then the stream could be "resynchronized".

In my opinion, that will be better than nothing.

**#3 - 07/16/2014 09:28 AM - Anoop Saldanha**

The resynchronization idea is possible, yeah. We had planned to use synchronization against a PDU record boundary for an app layer protocol in another area of the engine(picking up streams midstream). The same logic can be extended here.

Of course, this will require more than some support from the parsers. For example, we will have to inform the parser, that there is a gap, and we want to send a fresh PDU, and it will have to reset the state, etc. with whatever pending state it has from a previously parsed but broken PDU.

**#4 - 01/01/2016 06:03 PM - Andreas Herz**

- **Assignee set to OISF Dev**
- **Target version set to TBD**