Suricata - Support #3122

action missing in packet delivered in Bypass Callback

08/22/2019 07:29 PM - Phil Young

Status: Feedback
Priority: Normal
Assignee: Community Ticket
Category: 
Affected Versions: 
Label: 

Description
When the bypass callback is made to the capture method as the result of a rule, the packet structure delivered in the callback does not include the action. For example I have the following rule defined:

```
drop tcp 10.0.2.15 any -> any any (msg: "Bypass Test Rule"; classtype:protocol-command-decode; bypass; sid:1000013; rev:2;)
```

I get a call to my bypass-callback in the capture method when a packet matches this rule. However, the action field in the packet is always zero.

The use-case I'm trying to implement is to either block or shunt traffic - in an inline configuration - based on a particular signature. E.g. if the rule is drop/bypass the hardware will drop packets matching the sig. Similarly if the rule is pass/bypass the hardware will shut traffic to the output port without sending them to the host. I need to know the signature action in order to implement this.

History

#1 - 08/29/2019 08:10 PM - Andreas Herz
- Tracker changed from Bug to Support
- Status changed from New to Feedback
- Assignee set to Community Ticket
- Target version set to Support

Do you have a pcap for that so someone can try to reproduce it?

#2 - 09/12/2019 04:19 PM - Phil Young
- File test.pcap added

I've attached a file that I use. However, this can be observed using any rule/traffic combination that triggers the bypass callback. I've tried find a fix for this myself. But, it's just not clear to me which part of the the code should/could provide this info.

I have found a workaround. I currently tag the packet in the bypass callback, and later, program the hardware with the correct bypass action (drop/forward) when the packet is returned to the packet processing thread of the capture method. This has performance implications however, and could be done more efficiently in the callback if only the action information were available.

#3 - 09/13/2019 05:54 AM - Victor Julien

I think the issue is that the bypass is called from the post-match callback, but before the rule actions are applied to the packet.

Maybe we can call the post-match after applying the rule actions, but that would need careful review. I fear side effects there, esp since bypass is not the only users of the post-match callback.

Something that would require larger changes might ultimately be cleaner: introduce a new kind of callback after the actions are applied.

Another approach could be to update the Bypass API to allow for simply passing the action to it.

Eric Leblond any thoughts?

#4 - 09/13/2019 09:11 AM - Eric Leblond

Victor Julien wrote:

I think the issue is that the bypass is called from the post-match callback, but before the rule actions are applied to the packet.
Maybe we can call the post-match after applying the rule actions, but that would need careful review. I fear side effects there, esp since bypass is not the only users of the post-match callback.

Something that would require larger changes might ultimately be cleaner: introduce a new kind of callback after the actions are applied.

Another approach could be to update the Bypass API to allow for simply passing the action to it.

Eric Leblond any thoughts?

We could update the bypass keyword to add drop action. I see two ways to do that 'bypass:drop' syntax or at bypass parsing time check if signature has a drop and set the action in postmatch. To avoid side effect, we can also simply store drop action in bypass keyword and send it to the callback method.

---

**Files**

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>test.pcap</td>
<td>2.66 KB</td>
<td>09/12/2019</td>
<td>Phil Young</td>
</tr>
</tbody>
</table>