Suricata - Feature #294
Limit inspection of a stream and/or rule...
06/30/2011 02:18 AM - Edward Fjellskål

Status: New
Priority: Normal
Assignee: Community Ticket
Category:
Target version: TBD
Effort: medium
Difficulty: low

Description
I would like to have different rule options to limit inspection of a stream.

Not sure if it would speed up or slow down performance, but I see it nice for limiting false positives as some rules are specifically
looking for something in the start of the session/stream.

Say we have a binary protocol, and we are just looking for something that identifies it in the start of the stream/session, it would be
waste of cycles to keep inspecting that stream with that rule (And maybe other rules?)
A bit like snorts "preprocessor ssl: noinspect_encrypted"... Just editable by a user...

Example_1 (stream_noinspect;):
alert tcp $EXTERNAL_NET 888 -> $HOME_NET any (msg:"Stop inspect custom encrypted/binary protocol SSLv8"; flow:established,from_server; content:"|02|Handshake done|03|"; flowbits:noalert; stream_noinspect; classtype:friendly_fire; sid:888; rev:1;)

Example_2 (stream_depth:5; and stream_noinspect;):
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Gh0st Trojan CnC"; flow:established,to_server; content:"Gh0st"; stream_depth:5; stream_noinspect; classtype:trojan-activity; sid:2010859; rev:100;)

There could be:
stream_noinspect or stream_noinspect_both (The same): Stops inspections of client and server traffic in the stream.
stream_noinspect_client: Stop inspection of traffic from client in the stream
stream_noinspect_server: Stop inspection of traffic from server in the stream
If the rule matches its conditions and one of the above options are presents, it would stop inspect respectively.

stream_depth (and stream_depth_raw): Only inspect a certain amount of bytes in a reassabled (or raw?) stream for the respective
content:"" match in a rule? The rule would no longer be applied to packets in the stream after it fires.

I have no idea if my stream_* thoughts are a good or bad idea... but here they are :)
client_packets:3; # The option is true if this is the 3rd packet seen from the client [1]
server_packets:10; # The option is true if this is the 10th packet seen from the server [1]
client_packets:<100; # The option is true if the client packet is less than 100 (use "<=" for less or equal)
server_packets:<500; # The option is true if the server packet is less than 500 (use "<=" for less or equal)
client_packets:>=200; # The option is true if the client packet is equal or over 200
server_packets:>=10; # The option is true if the server packet is equal or over 10

[1]
We count packets as they come in. We even could count retransmitted packets, we dont need to be 100% accurate?
We could look at packets between 0 and 10, if we know that what we are looking for is normally in packet 3, we add a slack to our search.

As stream_depth as proposed earlier, is looking at the reassembled stream, the *_bytes should work on raw data.
We dont care about retransmits or evasions. We can work with that not using fixed values [1].
If we know that what we are looking for, normally comes in the first 200 bytes of data from a server, we can broaden our "search" with client_bytes:100<>500;

client_bytes:<100; # The option is true if the raw client bytes are less than 100
server_bytes:<500; # The option is true if the raw server bytes are less than 500
client_bytes:100<>200; # The option is true if the raw client bytes are between 100 and 200 bytes
server_bytes:0<>22000; # The option is true if the raw server bytes are between 0 and 22000 bytes
client_bytes:101; # The option is true if the raw client bytes are exactly 101 bytes
server_bytes:555; # The option is true if the raw server bytes are exactly 555 bytes
client_bytes:<10; # The option is true if the raw client bytes are equal to or less than 10
server_bytes:>=10; # The option is true if the raw server bytes are equal to or more than 10

1. Just some rule example:
2. Poison Ivy
   alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET Possible PoisonIvy client Checkin"; flow:established,from_client; client_packets:3; dsize:256; flowbits:PI_CHECKIN; flowbits:PI_CHECKIN; classtype:trojan-activity; sid:1; rev:1;)
   alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET Possible PoisonIvy server Reply"; flow:established,from_server; server_packets:3; dsize:256; flowbits:SET_CHECKIN; classtype:trojan-activity; sid:2; rev:1;)
To take account retransmissions, we might even use: *_packets:<5; and add maybe add *_bytes:256;

1. tcp sessions without data
2. Looks for the closing of a connection and checks if there was no data in the session.
   alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET Possible TCP session without data (client)"; flow:established,from_client; client_bytes:0; classtype:prank; flags:F; sid:3; rev:1;)
   alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET Possible TCP session without data (server)"; flow:established,from_server; server_bytes:0; classtype:prank; flags:F; sid:4; rev:1;)
3. OR just simply
   alert tcp any any -> any any (msg:"ET Possible TCP session without data"; flow:established; flags:F; flow_bytes:0; classtype:prank; sid:5; rev:1;)

Also see the Flowint for similar usage, as this might be a temporary help to get similar functionality :)  
https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Flowint

#2 - 11/03/2011 02:10 PM - Peter Manev

Hi,

I personally like the idea - i see it useful, but i am not sure how (or how bad) it would affect the resource need? And how hard is it to implement?. May be an option in the yaml to switch that particular functionality "on/off", if it is resource intensive?

Any developers' oppinions?

Thanks
P.S.
Sorry for the double post, just forgot to keep it in the ticket itself.

#3 - 05/16/2012 05:53 AM - Victor Julien

- Assignee set to OISF Dev
- Priority changed from Low to Normal
- Target version set to 1.4

I think this is a good idea. Should improve performance and reduce accuracy.

The thing I wonder though is would rulesets implement this? Deciding to set a inspection limit based on a rule match may be controversial. What if the rule FPs? FN's can result...

Opinions are welcomed!
A crude version of this is the bypass keyword. A rule can use it to stop all further processing of that flow/stream. The bypass can even be offloaded to kernel or hardware.