Understand Snort 3: Stateful Signature Evaluation Byte_Jump

Contents

Introduction

This document describes the new techniques added in Snort 3 from 7.4 onward.

Background Information

- The Snort 3 detection module works in block mode. While that approach gives a performance advantage and implementation simplicity (relatively), it has some limitations on detecting signatures that span multiple data blocks.
- To ease user experience some improvements are already implemented in Snort, namely:
  1. Flowbits allow the rule writer to mark the network flow with a user-defined property; that property could be set, cleared and tested on any packet from the flow (it presents a way to conclude about some bigger signature over packets).
- A stream module accumulates wire packets into a rebuilt packet, which is a bigger and more meaningful block than a raw packet; evaluating IPS rules against the rebuilt packet gives more chances to see the whole picture and match a bigger pattern (signature).
- In some cases the rebuilt packet presents not only new data, but includes some portion of previous data already processed by detection; again that block of accumulated data allows signatures that span backwards on the flow (to some degree) to be detected.
- A stream splitter cuts the flow into blocks, but the cut point is potentially a weak point the attacker could use to avoid pattern detection; thus Snort has a jittering mechanism implemented to make

splitting more unpredictable. This further complicates the analysis for the attacker.

What’s New

Stateful signature evaluation is a new technique which can be added to the list. It extends detection abilities by enabling IPS rule evaluation over multiple blocks. Thus, a rule does not mismatch immediately if the current block lacks data, but, instead, waits for more data to arrive.

Supported Platforms

Minimum Software and Hardware Platforms

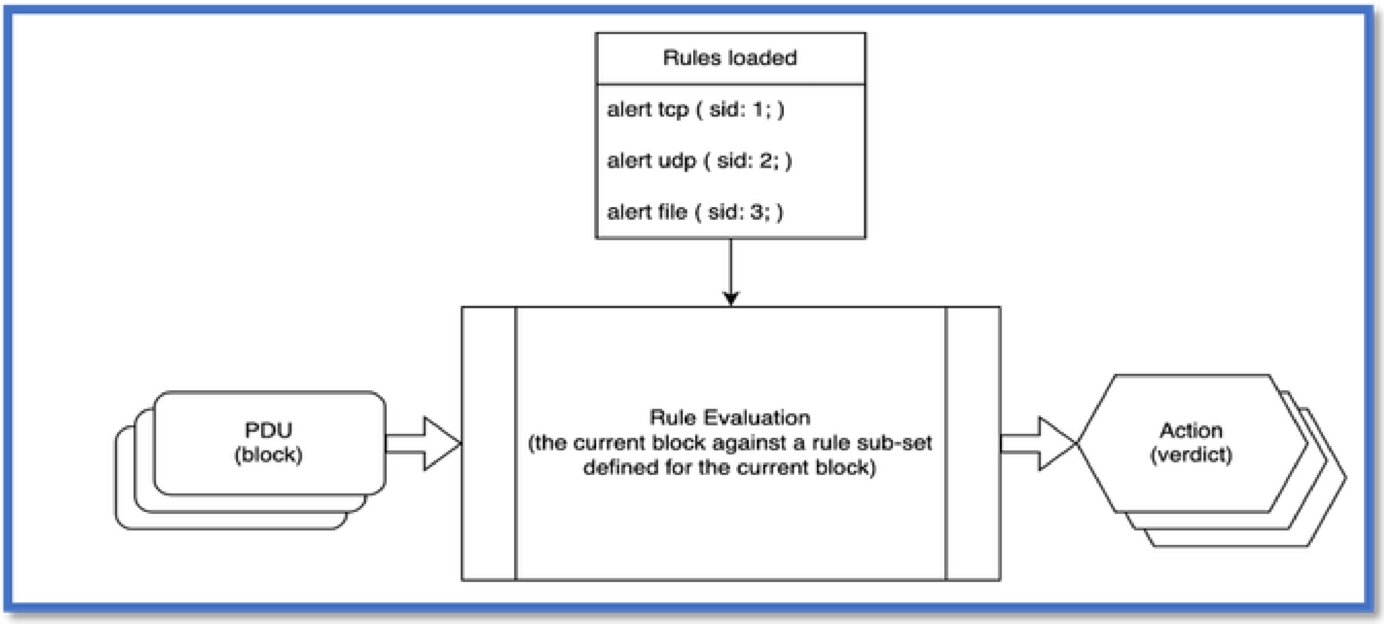
Min Supported Manager Version	Managed Devices	Min Supported Managed Device Version Required	Notes
Management Center 7.4.0	FTD	7.4.0	Snort 3 only
Device Manager 7.4.0	Any FTD which supports FDM management	7.4.0	Snort 3 only

Feature Details

Functional Feature Description

How Does It Work?

The detection module workflow is depicted in the diagram. At the traffic processing stage, the module already has all rules loaded, and it accepts data blocks in a one-by-one manner, evaluates rules, and defines the actions to be taken for the process stateful signature evaluation block.



Notes about the scheme:

1. Once a rule sub-set is defined for the current data block, each rule from it is evaluated independently from other rules.
2. Each data block is evaluated independently from other blocks.
3. The data block is an abstraction for a set of IPS buffers which are evaluated for the current packet.
4. Action is a list of actions evaluated for the current packet; the final verdict is determined later.

To understand how stateful signature evaluation works, take a look at how a common IPS rule is evaluated and how data blocks can form a stream.

Common Rule Evaluation

An IPS rule can be presented in this form:

```
action protocol source → destination ( option_1: parameters; option_2: parameters;
option_3: parameters; gid: 1; sid: 1; meta_option_1; meta_option_2; meta_option_3; )
```

Where:

action - IPS Action on the packet if the rule fires

protocol - protocol to match

source, destination - IP address and port

option_1, option_2, option_3 - IPS options which are the part of rule evaluation

gid, sid - a unique pair which identifies the rule (they are like metadata options)

meta_option_1, meta_option_2, meta_option3 - rule metadata like a message, a class type, or a reference, these options do not participate in rule evaluation.

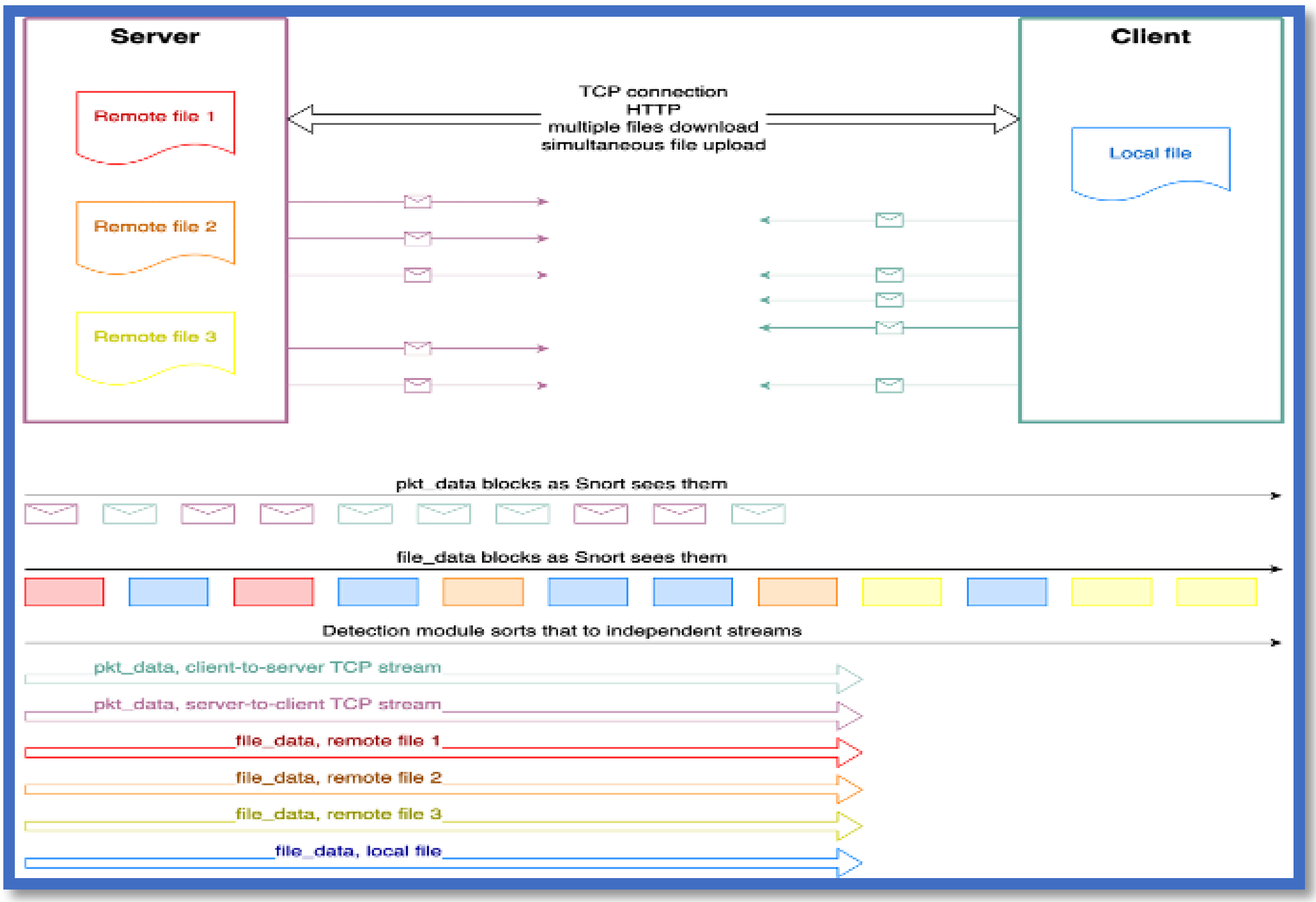
- Protocol, source, and destination form a rule header. It acts like a filter for a network flow (which flow to accept for evaluation). Everything in parentheses is a rule body. IPS options (except rule metadata) from the rule body are the ones which are evaluated for the data block. They comply with these statements:
- options are evaluated strictly in left-to-right order.
    1. can be one of two major types.
    2. buffer setter, the option selects the IPS buffer for the current packet.
- others (pattern search, math operation, cursor manipulation, flowbit operation)
- a cursor is used to track position in the selected IPS buffer.

- an option can be either:
    1. 'absolute', meaning it does not depend on the cursor position
    2. 'relative', meaning it starts its evaluation from the cursor position
- if an option attempts to set the cursor out of the selected IPS buffer, then it fails and the whole rule mismatches (due to lack of data)
- The last point is a limitation of the detection module. If Snort could have unlimited resources, it would cache all data seen to evaluate rules again and again when data becomes available (more wire packets arrive).

Data Stream and IPS Buffers

- The data stream is a stream of bytes in a contiguous form from the same source. It is a new concept presented to support stateful evaluation. Rule evaluation between blocks has to be done within the same logical data (whether it is a file, pure TCP stream, or JavaScript text).

- In general, a data block received by the detection module could:
    - Be from a different IPS buffer (for example, pkt_data and file_data are not the same)
    - Belong to another stream
    - Not form a stream (buffers generated from a raw packet)
    - Not form a contiguous stream (ICMP, UDP)
    - Be not in order (HTTP Partial Response)
    - Contain repeated data (an accumulated block, like in http_inspect.script_detection or HTTP Chunked Response)
- The detection module can sort things out to concatenate blocks from the same stream only, otherwise, the evaluation process would see unwanted interference from interleaving blocks.


Server
Client
TCP connection
HTTP
multiple files download
simultaneous file upload
Remote file 1
Remote file 2
Remote file 3
Local file
pkt_data blocks as Snort sees them
file_data blocks as Snort sees them
Detection module sorts that to independent streams
pkt_data, client-to-server TCP stream
pkt_data, server-to-client TCP stream
file_data, remote file 1
file_data, remote file 2
file_data, remote file 3
file_data, local file

Note: The example here presents a case where an HTTP client uploads and downloads multiple files simultaneously.

- Currently, only two IPS buffers can represent a stream: pkt_data and file_data, where:
    1. pkt_data form two streams for TCP protocol (client-to-server and server-to-client directions)
    2. file_data must form streams for files, MIME attachments, and other protocol data (like HTTP HTML page and/or other Content-Type)
- Stateful evaluation is done strictly within the data stream.

Rule Continuation

- The section earlier ends with a statement that the IPS option mismatches if it sets the cursor out of the current IPS buffer. But when the IPS buffer forms a data stream, the stateful signature evaluation feature steps in and saves the rule evaluation context in the Snort flow object. The saved evaluation context (state) is called rule continuation. Stateful signature evaluation postpones the rule's final verdict till more data becomes available.
- Rule continuation has three major parts: IPS buffer name, buffer source and targeted cursor position (buffer source is a unique identifier for the data stream).
- When a data block is processed by the detection module, the subsequent actions take place:-

    - Stateful signature evaluation creates a rule continuation and attaches it to the flow if:
        - IPS option (byte_jump, content, pcre or anything else which updates cursor position) sets the cursor after the current IPS buffer
        - The current IPS buffer supports the data stream.
        - The current IPS buffer forms a data stream at the moment.
- Stateful signature evaluation withdraws just-created rule continuation and removes it from the flow if:
    - The IPS rule has fired on the current data block (the rule matches on other places of the block)
- Stateful signature evaluation rejects pending rule continuations and removes them from the flow if:
    - IPS buffer does not form a contiguous stream (for example, the blocks have repeated data in them, or there is a gap (some part of data was missed or the block is not in order).
- Stateful signature evaluation updates targeted cursor position with new data available when:
    - The buffer source from the rule continuation is the same as the selected buffer source
    - IPS buffer forms a contiguous stream
- Stateful signature evaluation sends the rule continuation back to the IPS rule engine when:
    1. Targeted cursor position points inside the selected IPS buffer (which means, it finally received all data needed to complete rule evaluation).

User Configurations

- Since rule continuations take memory, Snort cannot store an unlimited number of them. There is a configuration option to control the limit:
    1. Detection.max_continuations_per_flow = 1024: maximum number of continuations stored simultaneously on the flow { 0:65535 }
- When stateful signature evaluation reaches the limit, it replaces the oldest rule continuation with a new one.
- The oldest rule continuation residing on the flow is there for too long, meaning it still does not meet a condition to resume rule evaluation.
- Additionally, there are plenty of peg counts available to fine-tune IPS rules (which must be the main focus) and the limit (if needed):
    1. detection.cont_creations: total number of continuations created (sum)
    2. detection.cont_recalls: total number of continuations recalled (sum)
    3. detection.cont_flows: total number of flows using continuation (sum)
    4. detection.cont_evals: total number of condition-met continuations (sum)
    5. detection.cont_matches: total number of continuations matched (sum)
    6. detection.cont_mismatches: total number of continuations mismatched (sum)
    7. detection.cont_max_num: peak number of simultaneous continuations per flow (max)
    8. detection.cont_match_distance: total number of bytes jumped over by matched continuations (sum)
    9. detection.cont_mismatch_distance: total number of bytes jumped over by mismatched continuations (sum)

Troubleshooting

The feature is an enhancement to the existing detection process, so cannot be troubleshoot explicitly. In case of any failures in detection, rules, configuration, or traffic must be examined.

Sample Problem

Problem: Description

- Let us say a signature has to check the beginning of the file and its tail at the same time.
- For example, in a targeted file of this structure (header, body, meta data) we need to see if any of its

metadata has a 0 value.

- File bytes: e1 f3 22 03 7f ff xx xx … xx 01 00 02 00 where
    - e1 f3 22 03 – 4 bytes for magic number, which identifies the file type
    - 7f ff – 2 bytes for body size
    - xx xx … xx – 32kb of some data
    - 01 00 02 00 – 4 bytes of meta data, in tag-value format (1 byte for each)
- IPS rule would look like: alert file (file_data; content:"|e1f32203|",fast_pattern; byte_jump:2,0,relative; content:"00",within:4, relative; sid: 1;)
    - Where
        - File protocol ensures that the rule accepts rebuilt packets only (raw packets do not participate in stateful signature evaluation)
        - The ‘file_data option selects a file data buffer, which can form a stream
        - 1st content option is a fast pattern and it checks for the magic number (if that is the intended file type)
        - byte_jump option reads the file body size and jumps over the file body
        - 2nd content option performs the final check for meta data values, within parameter limits search depth and makes the option relative.

Problem: Solution

The rule would be evaluated in this way:

Upon the 1st packet (of 8kB size), which carries a file header and a part of the body:

1. IPS buffer file_data is selected. The cursor points to the 0th byte e1.
2. The fast pattern option matches and sets the cursor position right after the magic number, pointing at byte 7f.
3. The byte_jump option reads two bytes of file body size. The cursor is updated by these two bytes. Then byte_jump calculates a jump for over 32768 bytes.
4. stateful signature evaluation creates a rule continuation, where it needs 24578 bytes more (32768 - (8kB - 4 bytes of header - 2 bytes of body size)).
5. The whole rule mismatches, since the byte_jump option fails to set the cursor position that far.

Upon the 2nd packet (of 16kB size), which carries the file body part:

1. stateful signature evaluation sees pending rule continuation.
2. It selects the buffer by its name and sees that file_data is available and the new data size is 16384.
3. The updated cursor shows that 8194 bytes are still needed (24578 - 16384)
4. The rule is not resumed.

Upon the 3rd packet (of 8198 sizes), which carries file body part and metadata:

1. stateful signature evaluation sees pending rule continuation.
2. It selects the buffer by its name and sees that file_data is available and the new data size is 8198.
3. The updated cursor shows that the buffer has enough data, cursor position is 8194.
4. stateful signature evaluation deletes the rule continuation.
5. stateful signature evaluation resumes rule evaluation from the 2nd content option with the cursor pointing at byte 01.
6. The content option finds a match on the 2nd byte searched.
7. The whole rule finally fires.

Limitations Details and Common Problems

Limitations And Other Considerations

- Due to stateful signature evaluation implementation, Snort drops all pending rule continuations when it reloads its configuration. Note that rule continuations despite being dropped still occupy Snort memory till the next data block is sent to the detection module.
- The rule latency feature for the IPS rule in stateful evaluation acts the same as if it were a common rule evaluation. Evaluation time for rule parts on different data blocks is summarized. If the time exceeds the limit, rule evaluation does a short-circuit, exiting earlier.
- Flowbits operations preserve their meaning, though they still perform like 'static' options.
  A flowbit set/clear/test operation is performed within a currently known context. So, if the flowbit option is evaluated in a rule continuation, it would take into account the current environment (flowbits set), not the one when the rule started its evaluation.

Also, a rule writer has to pay attention to the fast-pattern location.

Even if it can be in any part of the rule, the fast-pattern option is evaluated before the whole rule. It triggers rule evaluation. For stateful signature evaluation-based rule it means that rule continuation point must be after the fast-pattern option.
Additionally, the IPS rule can have multiple rule continuations in its evaluation (one after the other, not at the same time). Since any option from the rule body can have its continuation, it allows the rule writer to perform extra checks in different places of the data stream with the same IPS rule.