Rules Authors Introduction to Writing Snort 3 Rules

Generated: 2020-09-03
Author: Yaser Mansour

This guide introduces some of the new changes to Snort 3 rules language. The goal of this guide is to facilitate the transition of rules writing skills from Snort 2 to Snort 3 syntax.

Rule Header

The rule header follows a specific format:

`Action Protocol Networks Ports Direction Operator Networks Ports`

Examples:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (RULE_OPTIONS)
alert udp $EXTERNAL_NET $FILE_DATA_PORTS -> $HOME_NET any (RULE_OPTIONS)
```

In Snort 3, the options Protocol, Networks, Ports, and Direction Operator are optional and can be omitted, effectively matching any; analogous to replacing Networks and Ports with the keyword `any`.

Examples:

```
alert tcp (RULES_OPTIONS)
```

This allows for faster and less redundant rules authoring. Omitting the options should be selective to avoid ambiguity when reading the rules. A typical scenario is when writing rules to detect content regardless to its direction (inbound/outbound) and protocol (tcp, udp, or icmp).

Alert “http” Service Keyword

In Snort 2, the protocol used when writing rules to detect content in the HTTP URI, Header, or Body is defined as `tcp`. In Snort 3, a new protocol keyword `http` is available for HTTP content detection. This provides the following benefits:

1. Snort can detect and alert on HTTP content regardless of ports (HTTP on non-standard ports). Thus, the rule writer need not to worry about the ports some malware is communicating on.
2. The service mapping defined in the metadata option in Snort 2 (e.g.: service http) is no longer required. Thus, removing the burden from the rule writer of having to define the service mapping in the rule metadata option. The metadata option is discussed further in this guide.
3. The ability to use new sticky and dynamic buffers available in Snort 3 allow for streamlined and potential performance improvements to Snort 3 rules as discussed in the following sections.

New Sticky Buffers and Dynamic Buffer Selectors

Sticky buffers such as `file_data` and `sip_header` allow rule writers to define the detection cursor at specific buffers that hold content such as the HTTP response body or SIP header. Sticky buffers must precede the content being detected and remain in effect until changed. Snort 3 introduces new sticky buffers and selectors, specifically for HTTP content detection, such as `http_uri` and `http_header`. In addition, Snort 3 adds dynamic buffer selectors as subcategories under certain sticky buffers, such as the `field` selector under the `http_header` sticky buffer. Using the `field` selector, the rule author can restrict the content match against a specific HTTP header, where header names are case insensitive.

Example – “http_uri” Sticky Buffer:

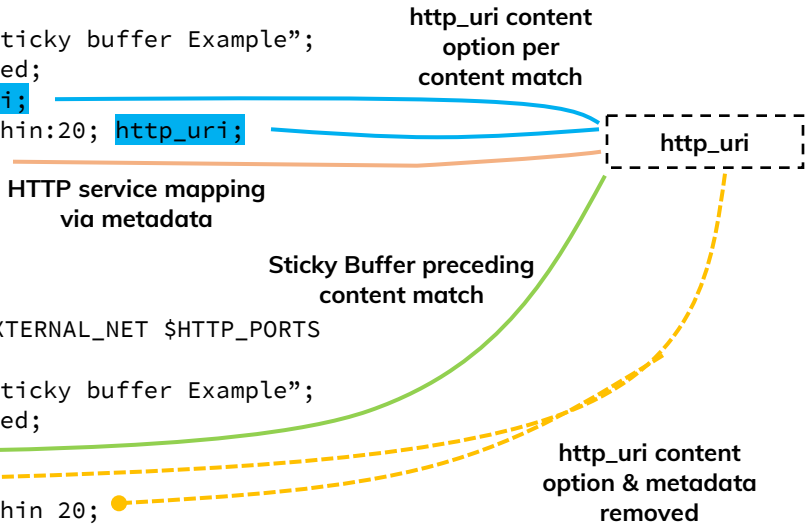
In this example, the use of the `http_uri` sticky buffer in Snort 3 removes the redundant need for using the `http_uri` content option after each content match.

Snort 2:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(
        msg: “Snort 3 http_uri sticky buffer Example”;
        flow:to_server,established;
        content:”var=1”; http_uri;
        content:”malicious”; within:20; http_uri;
        metadata: service http;
        sid:1;
)
```

Snort 3:

```
alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(
        msg: “Snort 3 http_uri sticky buffer Example”;
        flow:to_server,established;
        http_uri;
        content:”var=1”;
        content:”malicious”, within 20;
        sid:1;
)
```



Example – “http_header” Sticky Buffer and “field” Dynamic Buffer Selector:

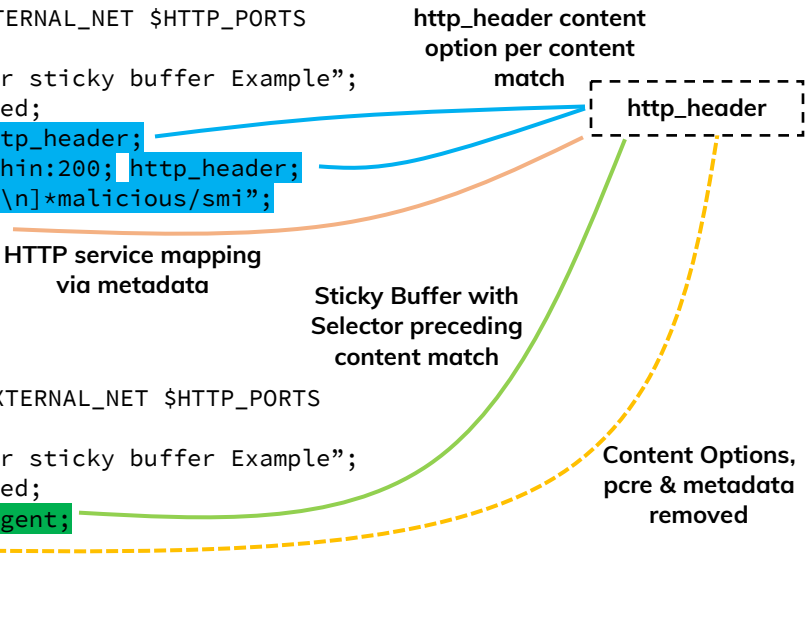
In this example, the use of the `http_header` along with the `field` selector allows the rule writer to target the content match at a specific field within the HTTP request header (case insensitive).

Snort 2:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(
        msg: “Snort 3 http_header sticky buffer Example”;
        flow:to_server,established;
        content:”User-Agent”; http_header;
        content:”malicious”; within:200; http_header;
        pcre:”/^User-Agent\s*:[^\n]*malicious/smi”;
        metadata: service http;
        sid:2;
)
```

Snort 3:

```
alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(
        msg: “Snort 3 http_header sticky buffer Example”;
        flow:to_server,established;
        http_header:field user-agent;
        content:”malicious”;
        sid:2;
)
```

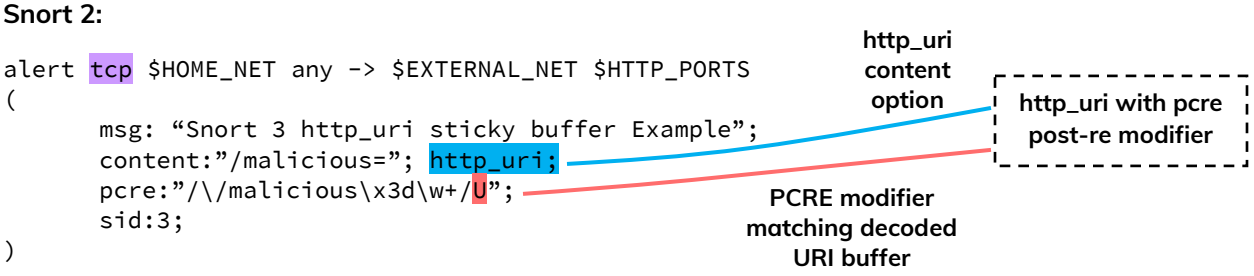


Example – Sticky Buffers and Deleted PCRE Options

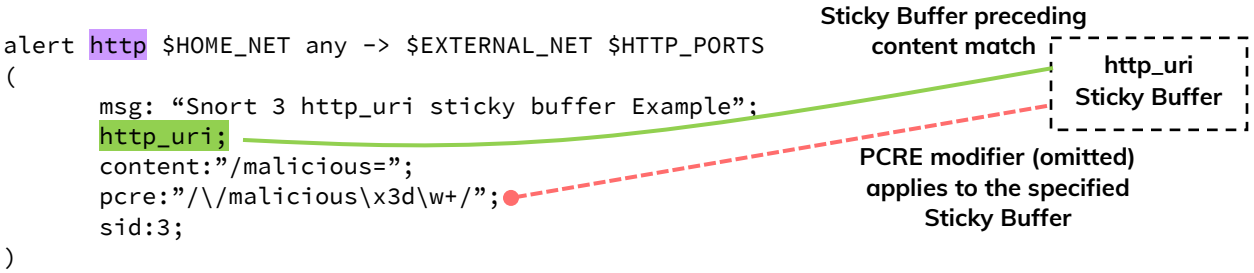
In Snort 2, the post-re modifiers (`B`, `U`, `P`, `H`, `M`, `C`, `I`, `D`, `K`, `S`, `Y`) set compile time flags for the regular expression. For example, the Snort specific modifier for pcre `U` is used to match the decoded URI buffers.

In Snort 3, some of post-re modifiers (`B`, `U`, `P`, `H`, `M`, `C`, `I`, `D`, `K`, `S`, `Y`) have been deleted in favor of sticky buffers.

Snort 2:



Snort 3:

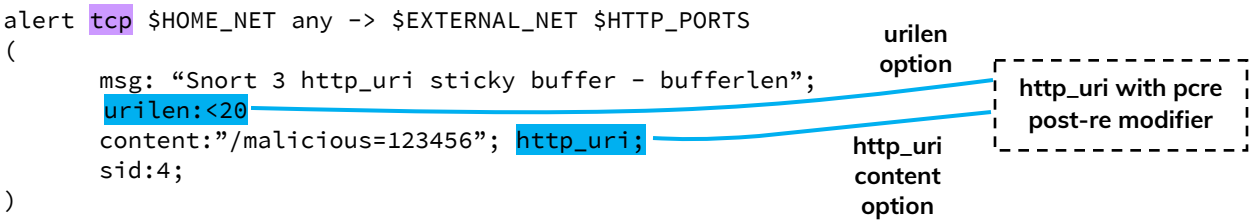


Example – Sticky Buffers and URL Length with “bufferlen” Option

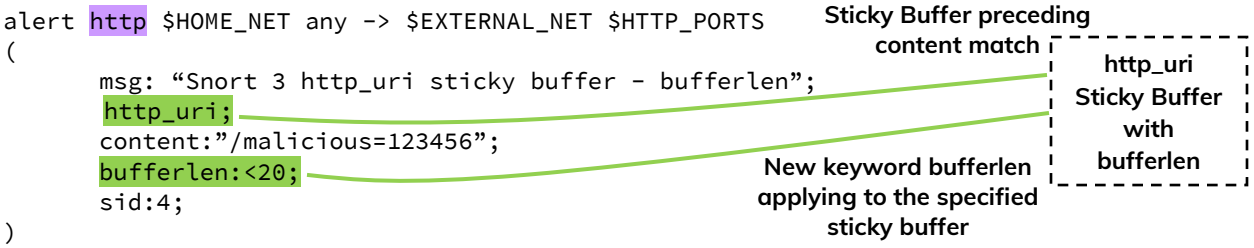
In Snort 2, inspecting the URI length is achieved via the `urilen` option.

In Snort 3, the option `urilen` is removed and is replaced with the generic buffer `bufferlen`, which applies to the specified sticky buffer.

Snort 2:



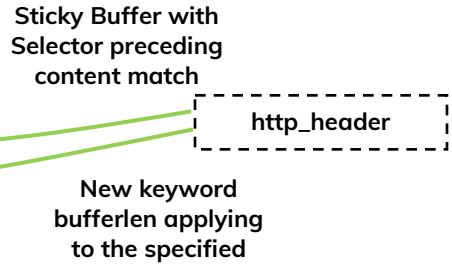
Snort 3:



Example – Sticky Buffers and User-Agent Length with “bufferlen” Option

The introduction of the generic buffer `bufferlen` in Snort 3 creates new detective capabilities. For example, detecting the length of the User-Agent in the HTTP request header.

```
alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(
       msg: “Snort 3 http_uri sticky buffer Example”;
       http_header:field user-agent;
       content:”malicious”;
       bufferlen:=10;
       sid:5;
)
```



Alert “file” Keyword

The `file` keyword after the rule action applies to anywhere a file is seen regardless of protocol or encoding. This removes the rule writer from the burden of:

1. Maintaining multiple rules to detect the same file or content over different protocols.
2. Maintaining multiple rules to detect the same or content traversing at different directions.
3. Creating or modifying rules when new protocols are added.
4. No having to worry about and potentially replace `flowbits` options in rules.

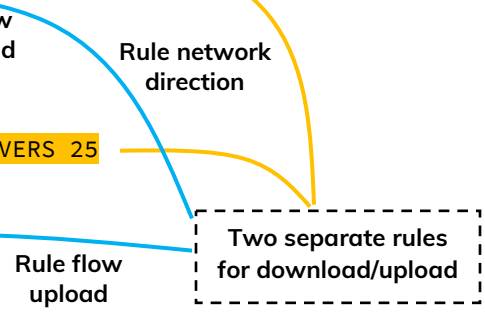
Example – Detect Specific Content Regardless of Protocol or Direction

This example attempts at detecting malicious content traversing the network over HTTP (download) and SMTP (upload).

Snort 2:

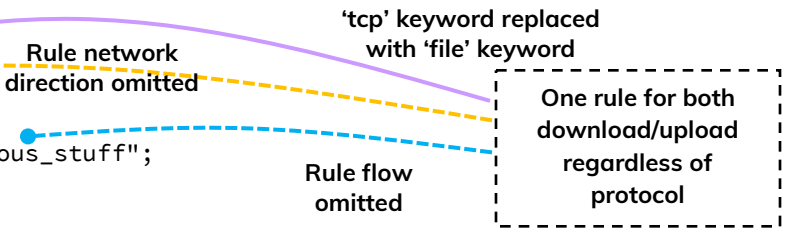
```
alert tcp $EXTERNAL_NET $FILE_DATA_PORTS -> $HOME_NET any
(
       msg: “Alert File example – Download”;
       flow:to_client,established;
       file_data; content:"malicious_stuff";
       sid:6;
)

alert tcp $EXTERNAL_NET $FILE_DATA_PORTS -> $SMTP_SERVERS 25
(
       msg: “Alert File example – Upload”;
       flow:to_server,established;
       file_data; content:"malicious_stuff";
       sid:7;
)
```



Snort 3:

```
alert file
(
       msg: “Alert File example”;
       file_data; content:"malicious_stuff";
       sid:6;
)
```



The use of the `file` keyword in Snort rules should also improve performance. The below examples demonstrating rules syntax differences between Snort 2 and Snort 3 although they are subtle.

Example – Detect Single File Type

Snort 2:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “PDF File Detected”;
       file_type: PDF;
       sid:8;
)
```

Snort 3:

```
alert file $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “PDF File Detected”;
       file_type: “PDF”;
       sid:8;
)
```

Sample Snort 3 Output (alert_talos)

```
##### test.pcap #####
       [1:8:0] PDF File Detected (alerts: 881)
#####
```

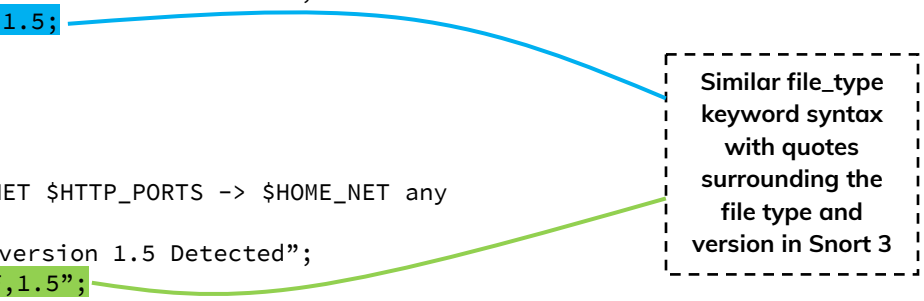
Example – Detect Single File Type with Version

Snort 2:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “PDF File version 1.5 Detected”;
       file_type: PDF,1.5;
       sid:9;
)
```

Snort 3:

```
alert file $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “PDF File version 1.5 Detected”;
       file_type: “PDF,1.5”;
       sid:9;
)
```



Sample Snort 3 Output (alert_talos)

```
##### test.pcap #####
       [1:9:0] PDF File version 1.5 Detected (alerts: 881)
#####
```

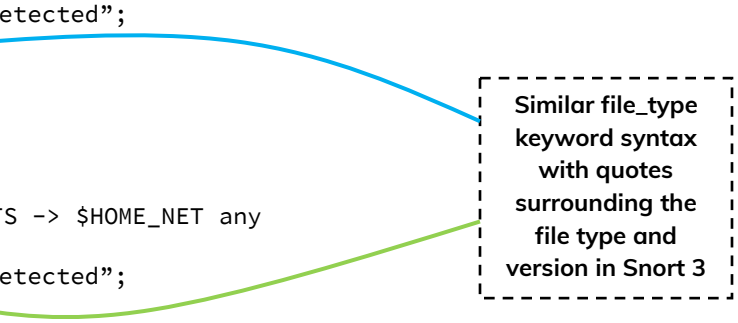
Example – Detect Single File Type with Multiple Versions

Snort 2:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “PDF File version 1.5 Detected”;
       file_type: PDF,1.5,1.7;
       sid:10;
)
```

Snort 3:

```
alert file $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “PDF File version 1.5 Detected”;
       file_type: “PDF,1.5,1.7”;
       sid:10;
)
```



Sample Snort 3 Output (alert_talos)

```
##### test.pcap #####
       [1:10:0] PDF File version 1.5 Detected (alerts: 881)
#####
```

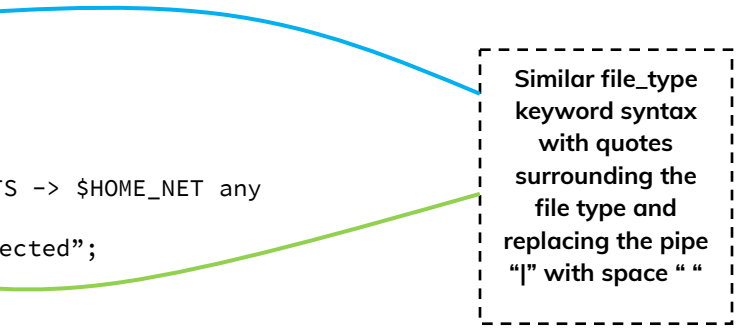
Example – Detect Multiple File Types

Snort 2:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “PDF or MSEXE Files Detected”;
       file_type: PDF|MSEXE;
       sid:11;
)
```

Snort 3:

```
alert file $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “PDF or MSEXE Files Detected”;
       file_type: "PDF MSEXE";
       sid:11;
)
```



Sample Snort 3 Output (-A cmg)

```
10/13-13:55:36.104000 [**] [1:11:0] "PDF or MSEXE Files Detected" [**] [Priority: 0] [AppID:
HTTP] {TCP} 173.37.145.84:80 -> 192.168.0.1:9208
http_inspect.stream_tcp[16389]:
- - - - - - - - - - - -  - - - - - - - - - - - -  - - - - - - - - -
4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ...... ........

0/13-13:55:44.130000 [**] [1:8:0] "PDF or MSEXE Files Detected" [**] [Priority: 0] [AppID: HTTP]
{TCP} 173.37.145.84:80 -> 192.168.0.1:14685
http_inspect.stream_tcp[16391]:
- - - - - - - - - - - -  - - - - - - - - - - - -  - - - - - - - - -
67 E1 39 D8 07 CF C3 0B  BF 28 E7 C9 21 0D 4F BD  g.9..... .(..!.O.
```

Example – Detect Multiple File Types and Versions

Snort 2:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “PDF File Detected”;
       file_type: PDF,1.6,1.7|RAR,1.1;
       sid:12;
)
```

Snort 3:

```
alert file $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “PDF File version 1.5 Detected”;
       file_type: "PDF,1.5,1.7 RAR,1.1";
       sid:12;
)
```

Similar file_type keyword syntax with quotes surrounding the file types and versions in Snort 3 and replacing the pipe “|” with space “ “

Application Detection with OpenAppID (ODP)

Snort 2:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “Firefox Detected”;
       appid: Firefox;
       sid:13;
)
```

Snort 3:

```
alert file $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(
       msg: “Firefox Detected”;
       appids: "Firefox";
       sid:13;
)
```

Replace “appid” keyword from Snort 2 with the keyword “appids” while also surrounding the app name with quotes

Rule Metadata Option

In Snort 2, certain keywords such as `engine`, `soid`, and `service` keys in the `metadata` option can affect Snort detection behavior, such as using `service` key for Target-Based Service Identifier when a Host Attribute Table is provided.

In Snort 3, `metadata` is now truly metadata with no impact on detection. Snort does not care about metadata internal structure/syntax.

Example – Replacing Services in Metadata with 'Service' Key in Metadata

Snort 2:

```
alert tcp any any -> any any
(
      msg: "Service Key Example";
      ...;
      metadata:service http, service smtp;
      sid:14;
)
```

'service' key is repeated for each service within the 'metadata' keyword

'service' key

Snort 3:

```
alert tcp any any -> any any
(
      msg: "Service Key Example";
      ...;
      service:http, smtp;
      sid:14;
)
```

'service' is a standalone keyword without repetition

Rule Remarks Option

The remarks `rem` option is a new option allows including arbitrary comments in the rule body.

Example – 'rem' Option

```
alert tcp any any -> any any
(
      msg: "rem option Example";
      ...;
      rem: "tlp white";
      sid:15;
)
```

'rem' option

References

1. https://www.youtube.com/watch?v=3gS7MKO-cFE
2. https://github.com/snort3/snort3
3. https://www.snort.org/documents/snort-rule-infographic
4. https://www.snort.org/documents/snort-users-manual-html
5. https://www.snort.org/downloads/snortplus/snort_manual.html
6. README.file_ips
7. Snort3/doc/differences.txt