Firepower Data Path Troubleshooting Phase 7: Intrusion Policy

Contents

Introduction
Prerequisites
Troubleshooting the Intrusion Policy Phase
Using the "trace" Tool to Detect Intrusion Policy Drops (FTD Only)
Check for Suppressions in the Intrusion Policies
Create a Targeted Intrusion Policy
False Positive Troubleshooting
True Positive Example
Data to Provide to TAC
Next Steps

Introduction

This article is part of a series of articles which explain how to systematically troubleshoot the data path on Firepower systems to determine whether components of Firepower may be affecting traffic. Please refer to the Overview article for information about the architecture of Firepower platforms and links to the other Data Path Troubleshooting articles.

This article covers the seventh phase of the Firepower data path troubleshooting, the Intrusion Policy feature.

Prerequisites

- This article is applicable to all Firepower platforms running an Intrusion Policy The **trace** feature is only available in version 6.2 and above for the Firepower Threat Defense (FTD) platform only
- Knowledge of open source Snort is helpful, though not required For information on open source Snort, please visit https://www.snort.org/

Troubleshooting the Intrusion Policy Phase

Using the "trace" Tool to Detect Intrusion Policy Drops (FTD Only)

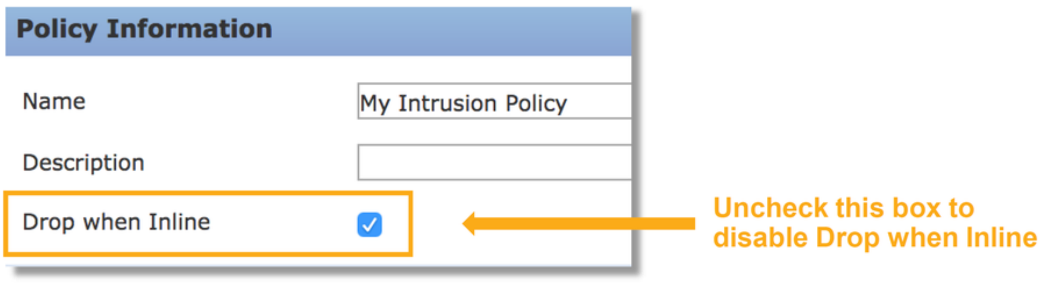
The system support trace tool can be run from the FTD Command Line Interface (CLI). This is similar to the **firewall-engine-debug** tool mentioned in the Access Control Policy phase article, except it digs deeper into the inner workings of Snort. This can be useful to see if any Intrusion Policy rules are triggering on the interesting traffic.

In the example below, traffic from the host with IP address 192.168.62.6 is being blocked by an Intrusion Policy rule (in this case, **1:23111**)

```
> system support trace


Please specify an IP protocol: tcp
Please specify a client IP address: 192.168.62.69
Please specify a client port:
Please specify a server IP address:
Please specify a server port:
Enable firewall-engine-debug too? [n]: y
Monitoring packet tracer debug messages

[… output omitted for brevity]

173.37.145.84-80 - 192.168.62.69-38488 6 Packet: TCP, ACK, seq 3594105349, ack 3856774965
173.37.145.84-80 - 192.168.62.69-38488 6 AppID: service HTTP (676), application Cisco (2655)
192.168.62.69-38488 > 173.37.145.84-80 6 AS 1 I 0 URL SI: ShmDBLookupURL("http://www.cisco.com/<?php") returned 0
…
192.168.62.69-38488 > 173.37.145.84-80 6 AS 1 I 0 match rule order 5, 'inspect it all', action Allow
192.168.62.69-38488 > 173.37.145.84-80 6 AS 1 I 0 allow action
192.168.62.69-38488 > 173.37.145.84-80 6 Firewall: allow rule, 'inspect it all', allow
192.168.62.69-38488 > 173.37.145.84-80 6 IPS Event: gid 1, sid 23111, drop
192.168.62.69-38488 > 173.37.145.84-80 6 Snort detect_drop: gid 1, sid 23111, drop
192.168.62.69-38488 > 173.37.145.84-80 6 AS 1 I 0 Deleting session
192.168.62.69-38488 > 173.37.145.84-80 6 NAP id 1, IPS id 0, Verdict BLACKLIST
192.168.62.69-38488 > 173.37.145.84-80 6 ===> Blocked by IPS
Verdict reason is sent to DAQ's PDTS
```

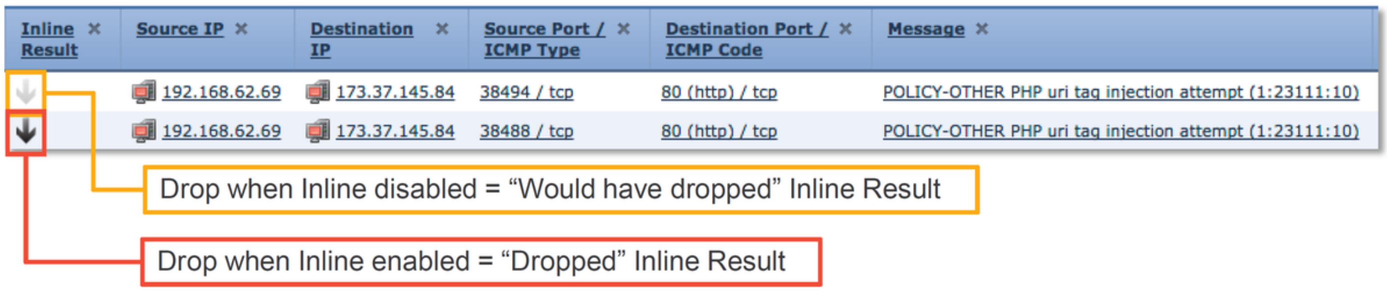
Notice that the action applied by snort was **drop**. When a drop is detected by snort, that particular session is then blacklisted so that any additional packets are dropped as well.

The reason why snort is able to perform the **drop** action is that the "Drop when Inline" option is enabled within the Intrusion Policy. This can be verified in the initial landing page within the Intrusion Policy. In the Firepower Management Center (FMC), navigate to **Policies > Access Control > Intrusion** and click the edit icon next to the policy in question.



Policy Information

Name	My Intrusion Policy
Description	
Drop when Inline	✓

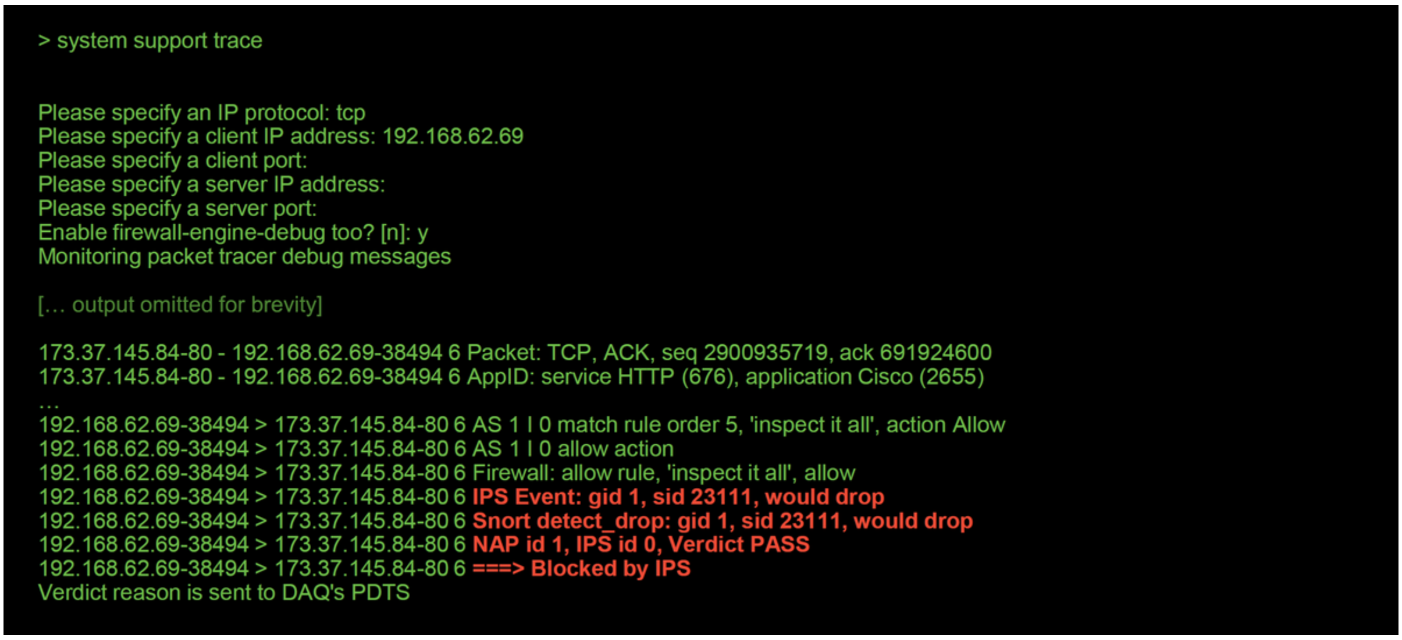


Inline Result	Source IP	Destination IP	Source Port / ICMP Type	Destination Port / ICMP Code	Message
⇣	192.168.62.69	173.37.145.84	38494 / tcp	80 (http) / tcp	POLICY-OTHER PHP uri tag injection attempt (1:23111:10)
↓	192.168.62.69	173.37.145.84	38488 / tcp	80 (http) / tcp	POLICY-OTHER PHP uri tag injection attempt (1:23111:10)

Drop when Inline disabled = "Would have dropped" Inline Result
Drop when Inline enabled = "Dropped" Inline Result

If "Drop When Inline" is disabled, snort no longer drops offending packets, but it still alerts with an **Inline Result** of "Would Have Dropped" in the Intrusion Events.

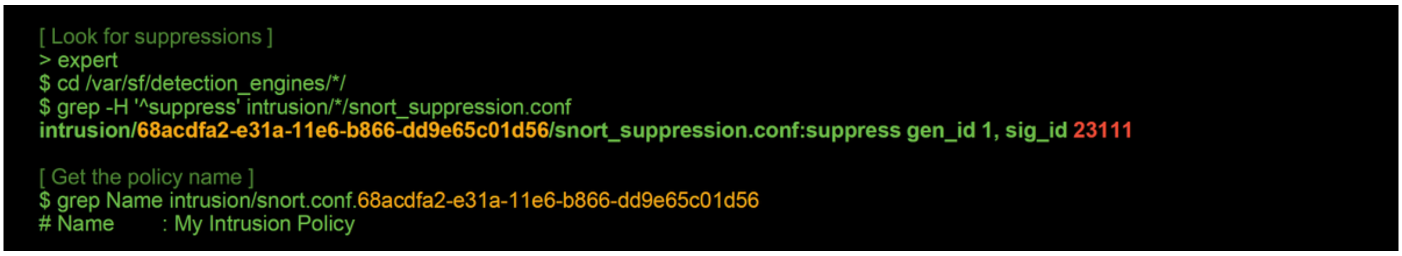
With "Drop When Inline" disabled, the trace output shows a **would drop** action for the traffic session in question.



```
> system support trace


Please specify an IP protocol: tcp
Please specify a client IP address: 192.168.62.69
Please specify a client port:
Please specify a server IP address:
Please specify a server port:
Enable firewall-engine-debug too? [n]: y
Monitoring packet tracer debug messages

[… output omitted for brevity]

173.37.145.84-80 - 192.168.62.69-38494 6 Packet: TCP, ACK, seq 2900935719, ack 691924600
173.37.145.84-80 - 192.168.62.69-38494 6 AppID: service HTTP (676), application Cisco (2655)
…
192.168.62.69-38494 > 173.37.145.84-80 6 AS 1 I 0 match rule order 5, 'inspect it all', action Allow
192.168.62.69-38494 > 173.37.145.84-80 6 AS 1 I 0 allow action
192.168.62.69-38494 > 173.37.145.84-80 6 Firewall: allow rule, 'inspect it all', allow
192.168.62.69-38494 > 173.37.145.84-80 6 IPS Event: gid 1, sid 23111, would drop
192.168.62.69-38494 > 173.37.145.84-80 6 Snort detect_drop: gid 1, sid 23111, would drop
192.168.62.69-38494 > 173.37.145.84-80 6 NAP id 1, IPS id 0, Verdict PASS
192.168.62.69-38494 > 173.37.145.84-80 6 ===> Blocked by IPS
Verdict reason is sent to DAQ's PDTS
```

Check for Suppressions in the Intrusion Policies

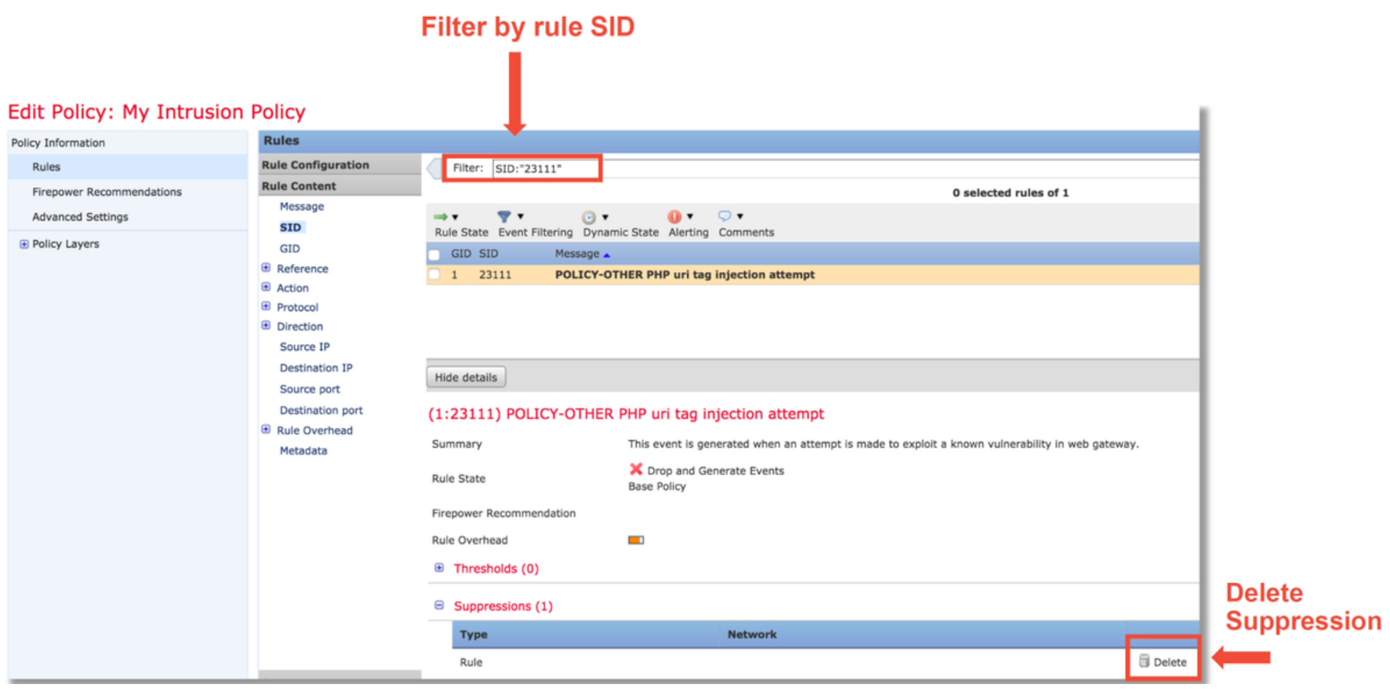
It is possible for snort to drop traffic without sending Intrusion Events to the FMC (silently drop). This is accomplished by configuring **Suppressions.** In order to verify whether any suppression has been configured in an Intrusion Policy, the expert shell can be checked on the backend, as illustrated below.



```
[ Look for suppressions ]
> expert
$ cd /var/sf/detection_engines/*/
$ grep -H '^suppress' intrusion/*/snort_suppression.conf
intrusion/68acdfa2-e31a-11e6-b866-dd9e65c01d56/snort_suppression.conf:suppress gen_id 1, sig_id 23111

[ Get the policy name ]
$ grep Name intrusion/snort.conf.68acdfa2-e31a-11e6-b866-dd9e65c01d56
# Name        : My Intrusion Policy
```

Notice that the Intrusion Policy called "My Intrusion Policy" contains a suppression for the 1:23111 rule. Therefore, traffic can be dropped due to this rule, without any events. This is another reason why the trace utility can be helpful, as it still shows the drops occurring.

In order to delete the suppression, the rule in question can be filtered within the Intrusion Policy **Rules** view. This brings up an option to delete the suppression, as shown below.



Edit Policy: My Intrusion Policy

Policy Information
Rules
Firepower Recommendations
Advanced Settings
Policy Layers

Rules
Rule Configuration
Rule Content
Message
SID
GID
Reference
Action
Protocol
Direction
Source IP
Destination IP
Source port
Destination port
Rule Overhead
Metadata

Filter: SID:"23111"

0 selected rules of 1

Rule State Event Filtering Dynamic State Alerting Comments

GID	SID	Message
1	23111	POLICY-OTHER PHP uri tag injection attempt

Hide details

(1:23111) POLICY-OTHER PHP uri tag injection attempt

Summary	This event is generated when an attempt is made to exploit a known vulnerability in web gateway.
Rule State	Drop and Generate Events Base Policy
Firepower Recommendation	
Rule Overhead	

Thresholds (0)

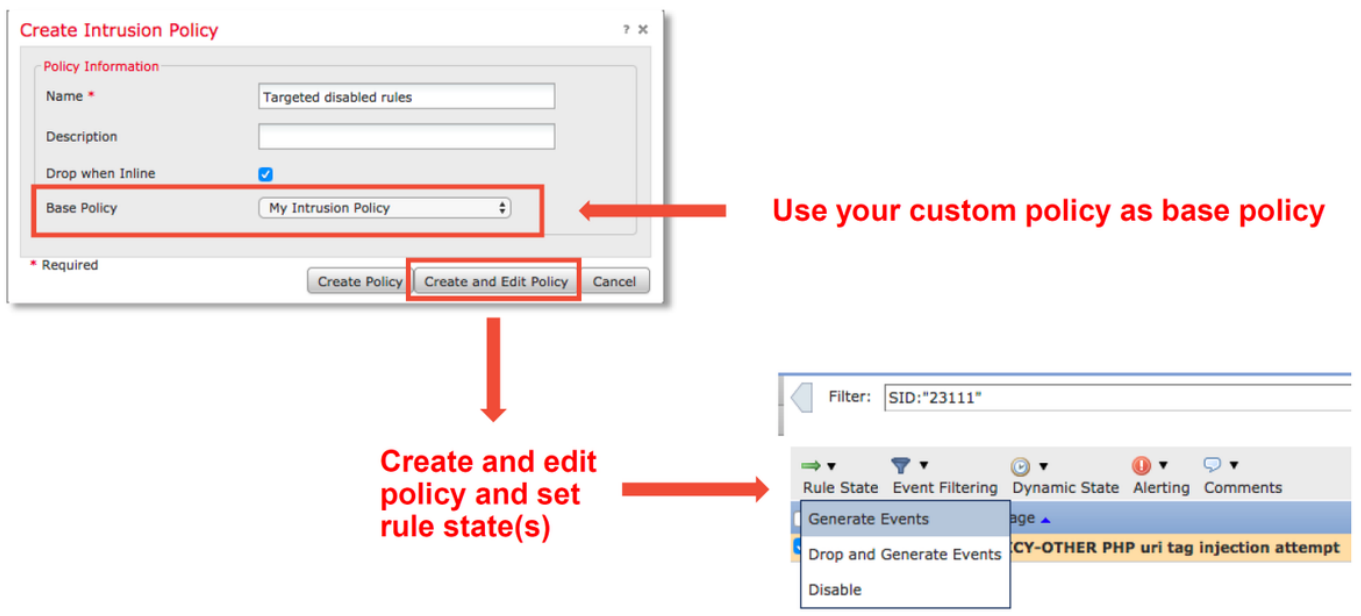
Suppressions (1)

Type	Network	
Rule		Delete

Create a Targeted Intrusion Policy

If traffic is being dropped by a particular Intrusion Policy rule, you may not want the traffic in question to be dropped but you may also not want to disable the rule. The solution is to create a new Intrusion Policy with the offending rule(s) disabled and then have it evaluate the traffic from the targeted hosts.

Here is an illustration on how to create the new Intrusion Policy (under **Policies > Access Control > Intrusion**).


Use your custom policy as base policy
Create and edit policy and set rule state(s)

Create Intrusion Policy

Policy Information

Name *	Targeted disabled rules
Description	
Drop when Inline	✓
Base Policy	My Intrusion Policy

* Required

Create Policy | Create and Edit Policy | Cancel

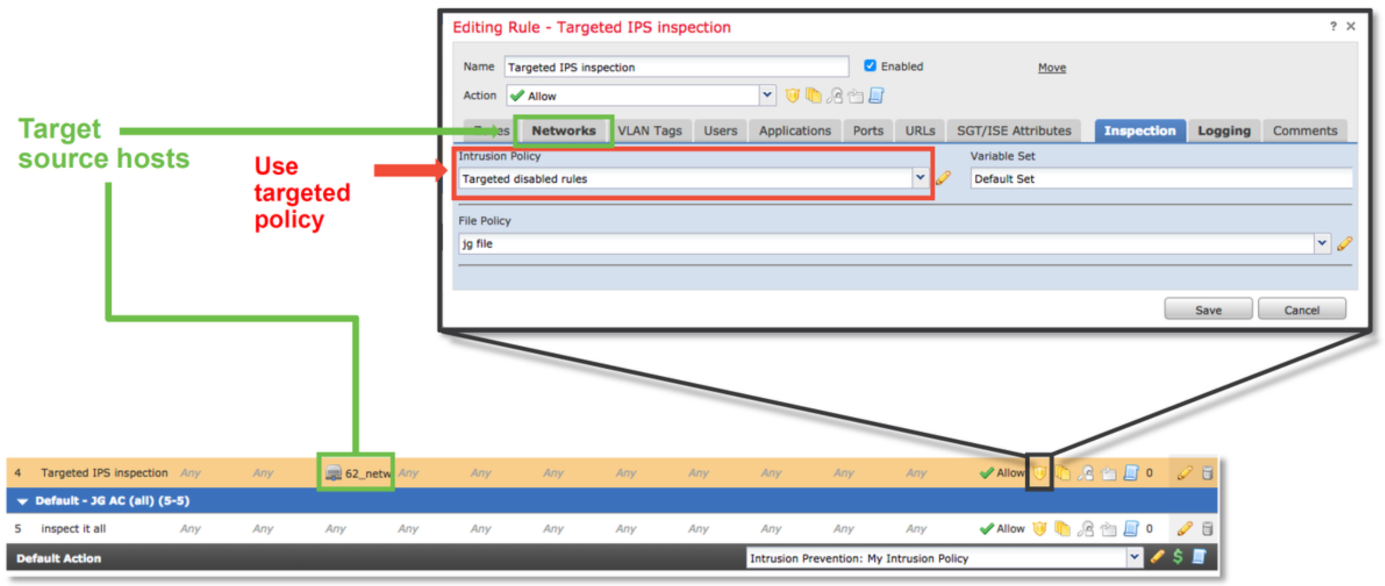
Filter: SID:"23111"

Rule State Event Filtering Dynamic State Alerting Comments

Generate Events
Drop and Generate Events
Disable

CY-OTHER PHP uri tag injection attempt

After creating the new Intrusion Policy, it can then be used within a new Access Control Policy rule, which targets the hosts in question, whose traffic was previously being dropped by the original Intrusion Policy.



Name: Targeted IPS inspection ☑ Enabled
Action: Allow
Intrusion Policy: Targeted disabled rules
Variable Set: Default Set
File Policy: jg file

4	Targeted IPS inspection	Any	Any	62_netw	Any	Any	Any	Any	Any	Any	Any	Any	Any	Allow
Default - JG AC (all) (5-5)														
5	inspect it all	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Allow

Default Action: Intrusion Prevention: My Intrusion Policy

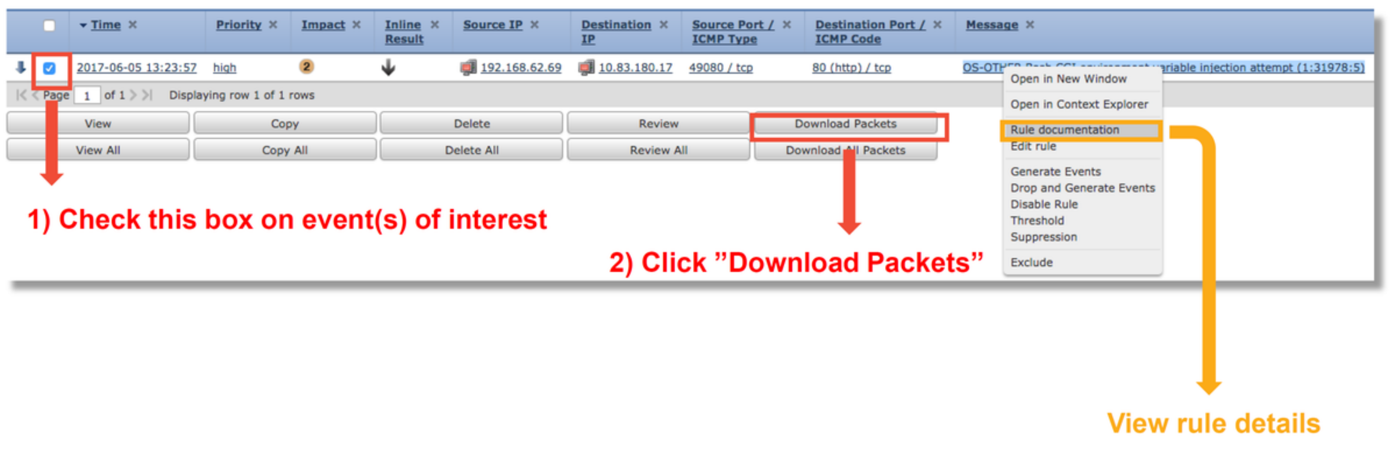
False Positive Troubleshooting

A common case scenario is false positive analysis for Intrusion Events. There are several things which can be checked before opening a false positive case.

1. From the **Table View of Intrusion Events** page, click the checkbox for the event in question

2. Click on **Download Packets** to get the packets captured by Snort when the Intrusion Event was triggered.

3. Right-click on the rule name in the **Message** column and then **Rule Documentation**, to see the rule syntax and other relevant information.



	Time	Priority	Impact	Inline Result	Source IP	Destination IP	Source Port / ICMP Type	Destination Port / ICMP Code	Message
☑	2017-06-05 13:23:57	high	2	↓	192.168.62.69	10.83.180.17	49080 / tcp	80 (http) / tcp	OS-OTHER Bash CGI environment variable injection attempt (1:31978:5)

Page 1 of 1 Displaying row 1 of 1 rows

Below is the rule syntax for the rule which triggered the event in the example above. The parts of the rule which can be verified against a packet capture (PCAP) file downloaded from the FMC for this rule are in bold.

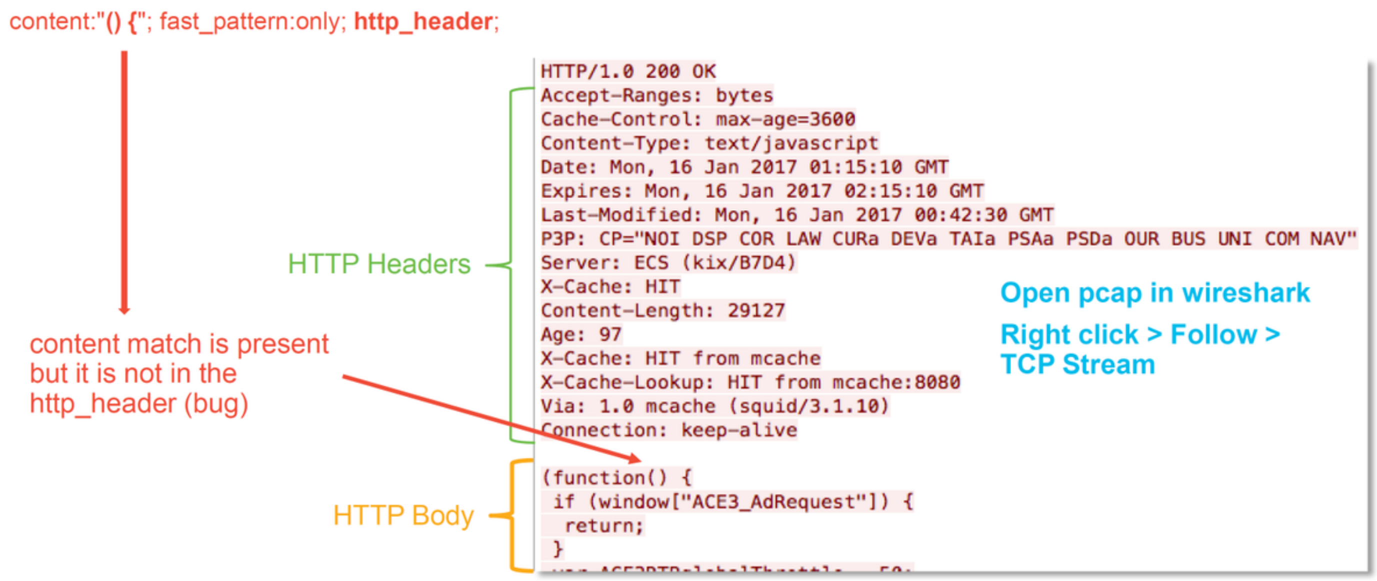
alert **tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS** \
(msg:"OS-OTHER Bash CGI environment variable injection attempt"; \
flow:to_server,established; \
content:"() {"; fast_pattern:only; http_header; \
metadata:policybalanced-ipsdrop, policy max-detect-ipsdrop, policy security-ipsdrop, ruleset

community, **service http**; \
reference:cve,2014-6271; reference:cve,2014-6277; reference:cve,2014-6278;
reference:cve,2014-7169; \
classtype:attempted-admin; \
sid:31978; rev:5;)

These initial steps can then be followed to perform the analysis process, to see if the traffic should have matched the rule which triggered.

1. Check the Access Control rule that the traffic matched. This information is found as part of the columns on the Intrusion Events tab.
2. Find the Variable Set used in said Access Control rule. The Variable Set can then be reviewed under **Objects > Object Management > Variable Sets**
3. Make sure the IP addresses in the PCAP file match variables (in this case, a host included in **$EXTERNAL_NET** variable connecting to a host included in the **$HOME_NET** variable configuration)
4. For **flow**, a full session/connection may need to be captured. Snort won't capture the complete flow due to performance reasons. However, in most cases, it is safe to assume that if a rule with flow:established triggered, the session was established at the time the rule triggered, so a full PCAP file isn't necessary to verify this option in a snort rule. But it may be useful to better understand the reason why it was triggered.
5. For **service http**, look at the PCAP file in Wireshark to see if it looks like HTTP traffic. If you have network discovery enabled for the host and it has seen the application "HTTP" before, it can cause service to match on a session.

With this information in mind, the packet(s) which are downloaded from the FMC can be further reviewed in Wireshark. The PCAP file can be evaluated to determine whether the event which is triggered is a false positive.


content:"() {"; fast_pattern:only; http_header;
HTTP Headers
HTTP Body
content match is present but it is not in the http_header (bug)
Open pcap in wireshark
Right click > Follow > TCP Stream

```
HTTP/1.0 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=3600
Content-Type: text/javascript
Date: Mon, 16 Jan 2017 01:15:10 GMT
Expires: Mon, 16 Jan 2017 02:15:10 GMT
Last-Modified: Mon, 16 Jan 2017 00:42:30 GMT
P3P: CP="NOI DSP COR LAW CURa DEVa TAIa PSAa PSDa OUR BUS UNI COM NAV"
Server: ECS (kix/B7D4)
X-Cache: HIT
Content-Length: 29127
Age: 97
X-Cache: HIT from mcache
X-Cache-Lookup: HIT from mcache:8080
Via: 1.0 mcache (squid/3.1.10)
Connection: keep-alive

(function() {
 if (window["ACE3_AdRequest"]) {
  return;
 }
```

In the illustration above, the content for which the rule detects was present in the PCAP file - **"() {"**

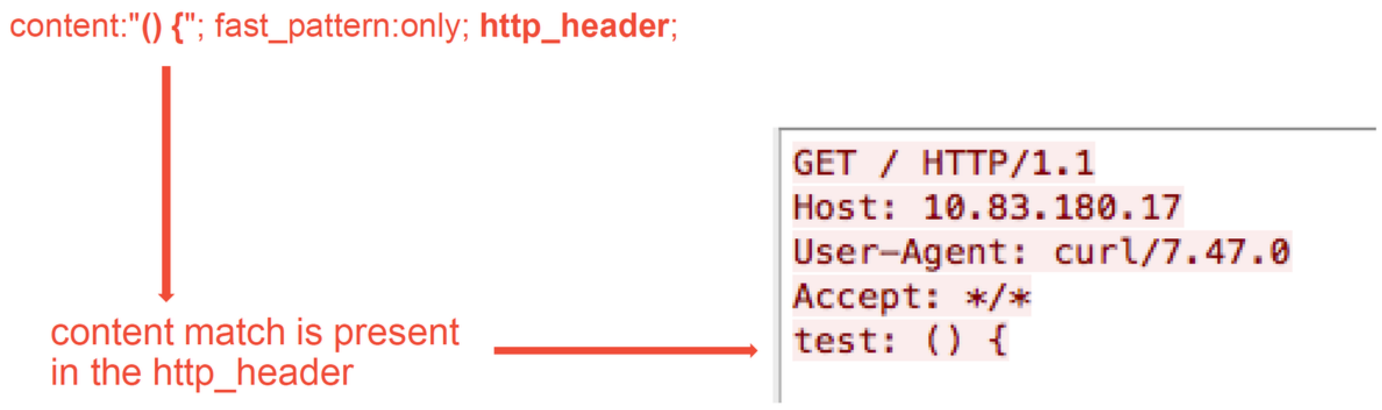
However, the rule specifies that the content should be detected in the HTTP header of the packet - **http_header**

In this case, the content was found in the HTTP body. Therefore this is a false positive. However, it is not a false positive in the sense that the rule is written incorrectly. The rule is correct and it can not be improved in this case. This example is likely encountering a Snort bug which is causing snort to have buffer confusion. This means that Snort has identified the http_headers incorrectly.

In this case, you can check for any existing bugs for snort/IPS engine in the version that your device is running, and if there are none, a case with Cisco Technical Assistance Center (TAC) can be opened. Full session captures are required to investigate such an issue as the Cisco team needs to review how Snort got into that state, which can't be done with a single packet.

True Positive Example

The illustration below shows packet analysis for the same Intrusion Event. This time, the event is a true positive because the content does appear in the HTTP header.


content:"() {"; fast_pattern:only; http_header;
content match is present in the http_header

```
GET / HTTP/1.1
Host: 10.83.180.17
User-Agent: curl/7.47.0
Accept: */*
test: () {
```

Data to Provide to TAC

Data	Instructions
Troubleshoot file from the Firepower device inspecting the traffic	http://www.cisco.com/c/en/us/support/docs/security/sourcefire-defense-center/117663-tech
Packet captures which were downloaded from the FMC	See this Article for instructions
Any relevant CLI output gathered, such as **trace** output	See this Article for instructions

Next Steps

If it has been determined that the Intrusion Policy component is not the cause of the issue, the next step would be to troubleshoot the Network Analysis Policy feature.

Click here to proceed to the last article.