Configure AppID Early Packet Detection in Secure Firewall Threat Defense 7.4

Contents

Introduction

This document describes how to configure AppID Early Packet Detection in Cisco Secure Firewall 7.4.

Background – Problem (Customer Requirements)

- Application detection through Deep Packet Inspection can take more than one packet to identify traffic.
- Sometimes, where IP and/or port for an application server is known, you can avoid inspecting additional packets.

What’s New

- A new Snort-based Lua AppID API has been created which allows us to map an IP address, port, and protocol to the respective:
    - Application protocol (service appid),
    - Client application (client appid) and
    - Web application (payload appid).
- Custom Application Detectors can be created on FMC using this API for application detection.
- Once this detector is activated, this new API would allow us to identify applications on the very first packet in a session.

Feature Overview

- The API is identified as:
    - **addHostFirstPktApp** (protocol_appId, client_appId, payload_appId, IP address, port, protocol, reinspect)
- A cache entry is created for every mapping created in the custom app detector.
- The first packet of all incoming sessions is inspected to see if a match is found in the cache.
- Once a match is found, we assign the corresponding appids for the session and the app discovery process stops.
- Users have the option to reinspect traffic even after a match was found by the API.
- The reinspect argument is a boolean value which indicates if there is a need to reinspect the applications found on the first packet or not.
- When reinspection is true, app discovery continues even if the API finds a match.
- In this case, the appids assigned on the first packet can change.

Prerequisites, Supported Platforms, Licensing

Minimum Software and Hardware Platforms

Application and Minimum Version	**Supported Managed Platform(s) and Version**	**Manager(s)**	**Notes**
Secure Firewall 7.4 Using Snort3	All platforms that support FTD 7.4	FMC On-Prem + FTD	This is a device-side feature; FTD must be on 7.4

Warning: Snort 2 does not support this API.

Snort 3, Multi-Instance, and HA/Clustering Support

Note: Requires that Snort 3 be the detection engine.

FTD	
Multi-instances supported?	Yes
Supported with HA’d devices	Yes
Supported with clustered devices?	Yes

Components Used

The information in this document is based on these software and hardware versions:
• Cisco Firepower Threat Defense running 7.4 or higher.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Feature Details

Functional Feature Description

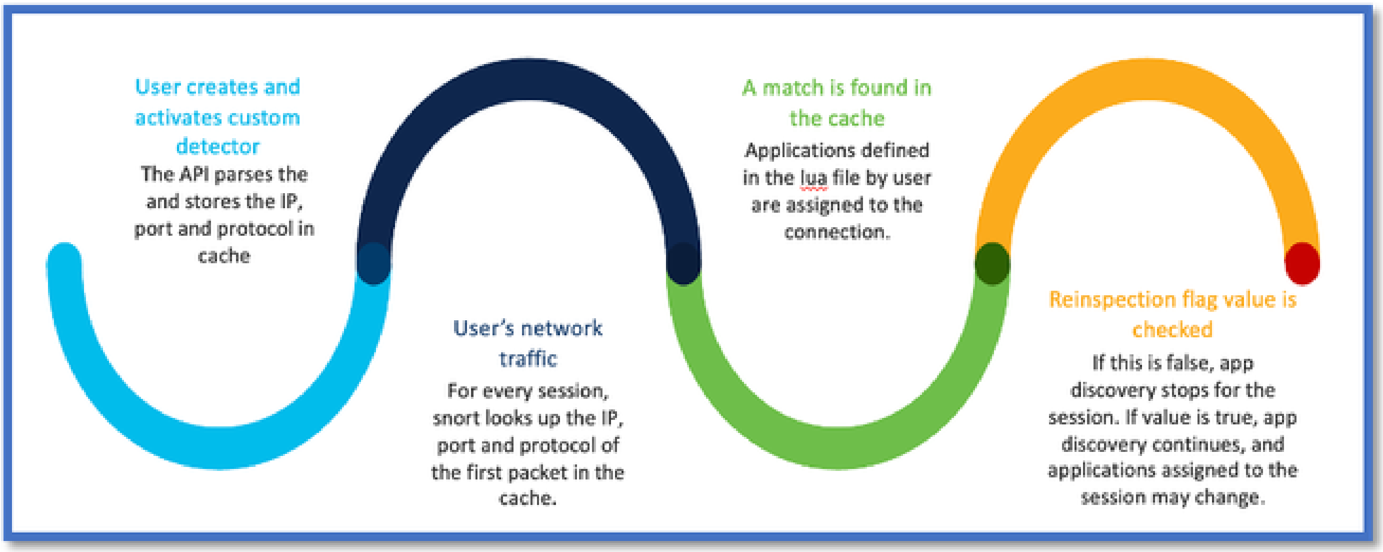
Contrasting Previous to This Release

In Secure Firewall 7.3 and Lower	New to Secure Firewall 7.4
• Application Detection for a known IP/Port/Protocol combination was only available as a fallback option after exhaustion of all other app detection mechanisms. • Essentially, detection on the first packet in a session was not supported.	• The new lua detector API is evaluated before any other app detection mechanism, • Thus in 7.4, we support detection on the very first packet in a session.

How it Works

1. Create a lua file: Ensure the file is in the lua template (no syntax errors). Also verify the arguments given to the API in the file are correct.
2. Create a new custom detector: Create a new custom detector on FMC and upload your lua file in it. Activate the detector.
3. Run traffic: Send traffic that matches the IP/port/protocol combination defined in the custom app detector to the device.
4. Check connection events: On FMC, check the connection events filtered by the IP and port. User-defined applications would be identified.

AppID Early Packet Detection API Workflow


User creates and activates custom detector
The API parses the and stores the IP, port and protocol in cache
User's network traffic
For every session, snort looks up the IP, port and protocol of the first packet in the cache.
A match is found in the cache
Applications defined in the lua file by user are assigned to the connection.
Reinspection flag value is checked
If this is false, app discovery stops for the session. If value is true, app discovery continues, and applications assigned to the session may change.

API Fields Description from Custom Detector Example

gDetector:addHostFirstPktApp

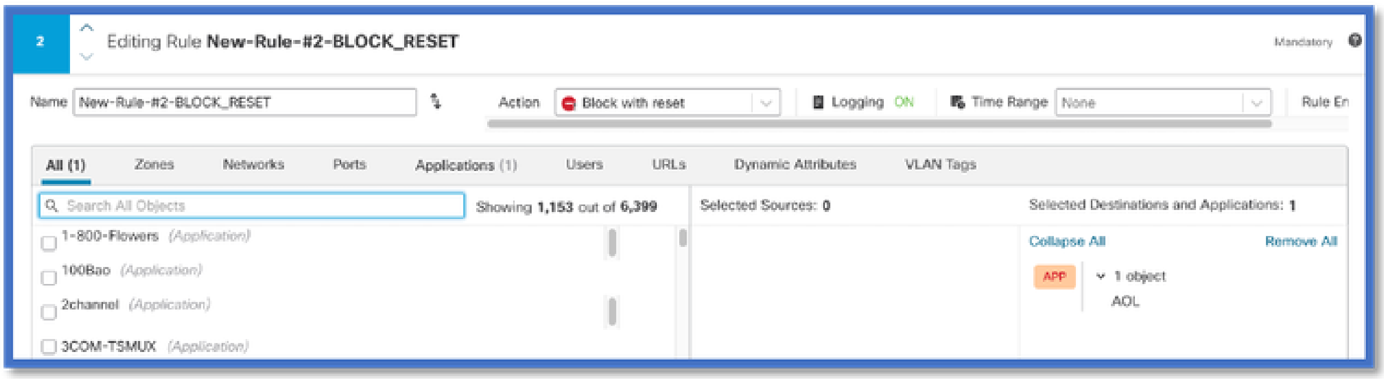
(gAppIdProto, gAppIdClient, gAppId, 0, "192.0.2.1", 443, DC.ipproto.tcp);

- The highlighted arguments are the user-defined values for the reinspect flag, IP address, port and protocol.
- 0 indicates a wildcard.

Arguments	**Explanation**	**Expected Values**
Reinspect flag	If a user prefers to inspect the traffic instead of taking firewall action based on IP/Port/Protocol, they can enable the reinspect flag value to 1.	0 = reinspect disabled or 1 = reinspect enabled
IP Address	Target IP (single or range of IPs in a subnet) of the server. Destination IP of the 1st packet in a session.	192.168.4.198 OR 192.168.4.198/24 OR 2a03:2880:f103:83:face:b00c:0:25de OR 2a03:2880:f103:83:face:b00c:0:25de/32
Port	Destination Port of the 1st packet in a session.	0 to 65535
Protocol	Network Protocol	TCP/UDP/ICMP

Use Case: How to Block Traffic Faster

- Policy View: Block Rule for the application “AOL”.



Name New-Rule-#2-BLOCK_RESET

Action Block with reset

Logging ON

Time Range None

Showing 1,153 out of 6,399

1-800-Flowers (Application)

100Bao (Application)

2channel (Application)

3COM-TSMUX (Application)

Selected Sources: 0

Selected Destinations and Applications: 1

APP 1 object

AOL

- Testing Traffic using curl with: curl https://www.example.com v/s curl https://192.0.2.1/ (one of TEST’s IP addresses)

<#root>

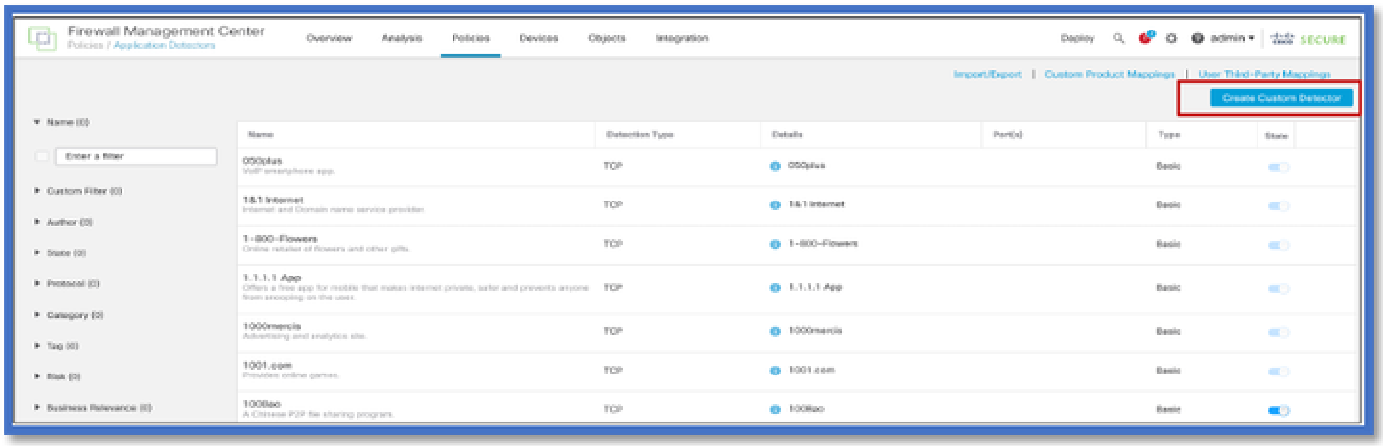
```
> curl https://www.example.com/

curl: (35) OpenSSL SSL_connect: SSL_ERROR_SYSCALL in connection to www.example.com:443

> curl https://192.0.2.1/

curl: (7) Failed to connect to 192.0.2.1 port 443: Connection refused
```

Firewall Management Center Walkthrough

Steps to Create a Custom Detector Using the API

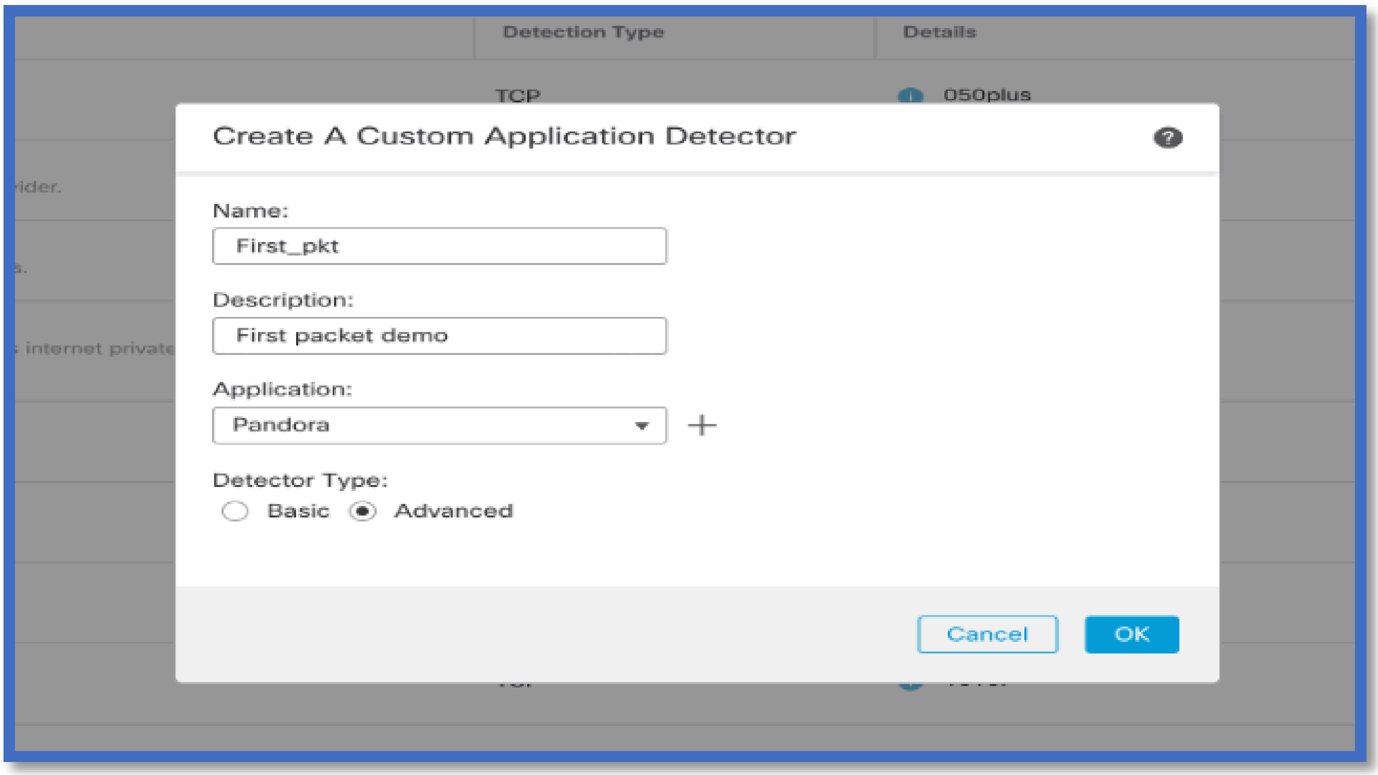
Create a new custom detector on the FMC from:

- Policies > Application Detectors > Create Custom Detector .



Name	Detection Type	Details	Port(s)	Type	State
050plus VoIP smartphone app.	TCP	050plus		Basic	
1&1 Internet Internet and Domain name service provider.	TCP	1&1 Internet		Basic	
1-800-Flowers Online retailer of flowers and other gifts.	TCP	1-800-Flowers		Basic	
1.1.1.1 App Offers a free app for mobile that makes internet private, safer and prevents anyone from snooping on the user.	TCP	1.1.1.1 App		Basic	
1000mercis Advertising and analytics site.	TCP	1000mercis		Basic	
1001.com Provides online games.	TCP	1001.com		Basic	
100Bao A Chinese P2P file sharing program.	TCP	100Bao		Basic	

- Define name and description.
  - Choose the application from the dropdown menu.
  - Select Advanced Detector Type.



Create A Custom Application Detector

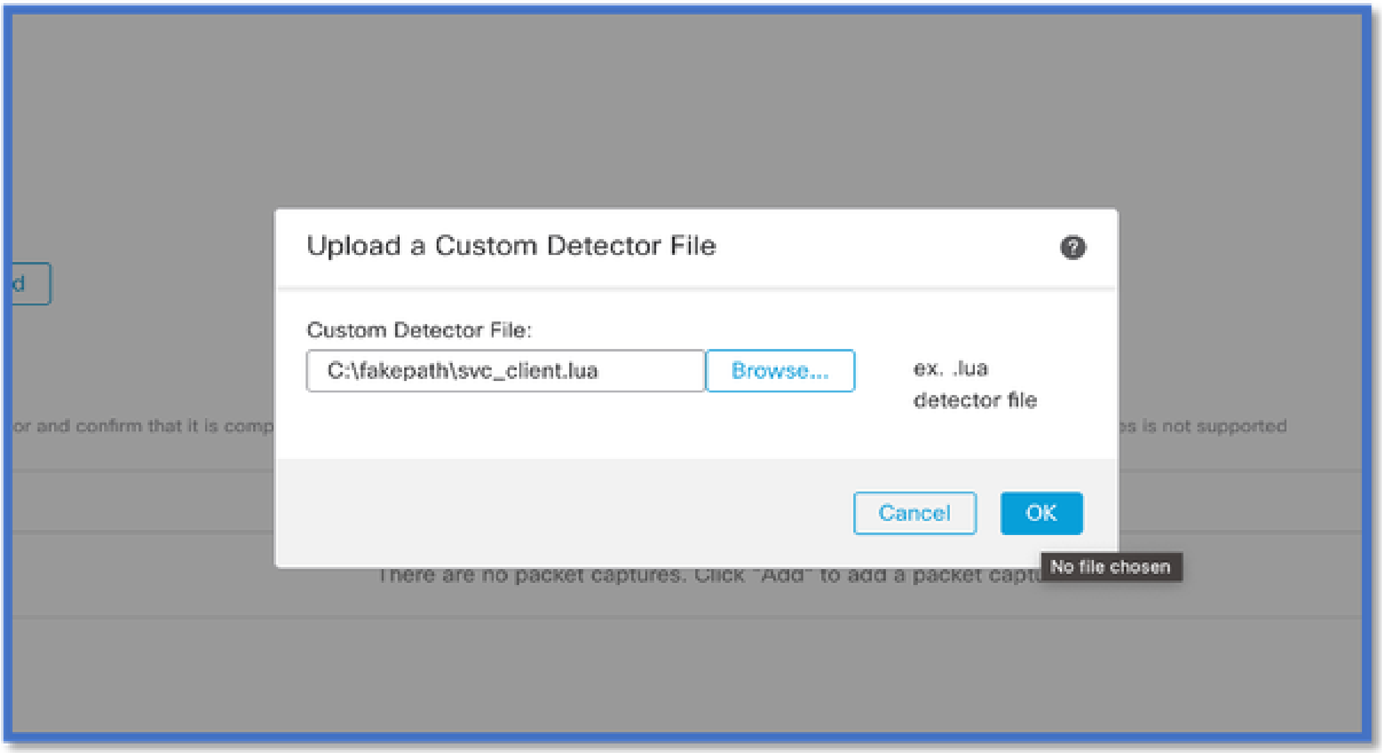
Name: First_pkt

Description: First packet demo

Application: Pandora

Detector Type: ◯ Basic [✓ Advanced]

- Upload the Lua file under Detection Criteria. Save and activate the detector.



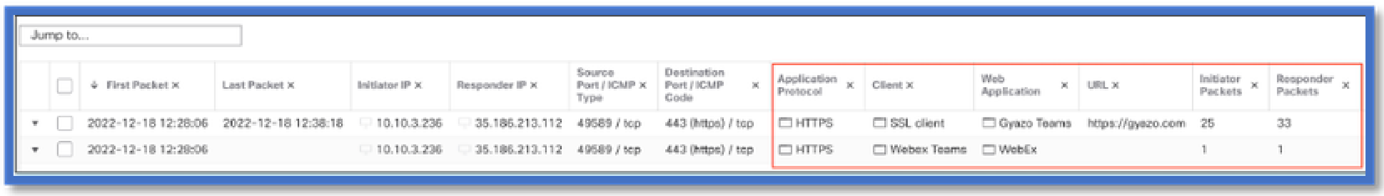
Upload a Custom Detector File

Custom Detector File: C:\fakepath\svc_client.lua

ex. .lua detector file

Browse...
Cancel
OK
No file chosen
There are no packet captures. Click "Add" to add a packet captu

Reinspect Enabled v/s Disabled



Jump to...

		First Packet	Last Packet	Initiator IP	Responder IP	Source Port / ICMP Type	Destination Port / ICMP Code	Application Protocol	Client	Web Application	URL	Initiator Packets	Responder Packets
▾	☐	2022-12-18 12:28:06	2022-12-18 12:38:18	10.10.3.236	35.186.213.112	49589 / tcp	443 (https) / tcp	HTTPS	SSL client	Gyazo Teams	https://gyazo.com	25	33
▾	☐	2022-12-18 12:28:06		10.10.3.236	35.186.213.112	49589 / tcp	443 (https) / tcp	HTTPS	Webex Teams	WebEx		1	1

- The two events show the beginning of the connection v/s the end of the connection when reinspection is enabled.

Note: Things to note:

1. ‘HTTPS, Webex and Webex Teams’ are identified by the API at the beginning of the connection. Since reinspection is true, app discovery continues and appIds are updated to ‘HTTPS, SSL Client and Gyazo Teams’.

2. Notice the number of initiator and responder packets. Regular app detection methods require a lot more packets than the API.

Troubleshooting/Diagnostics

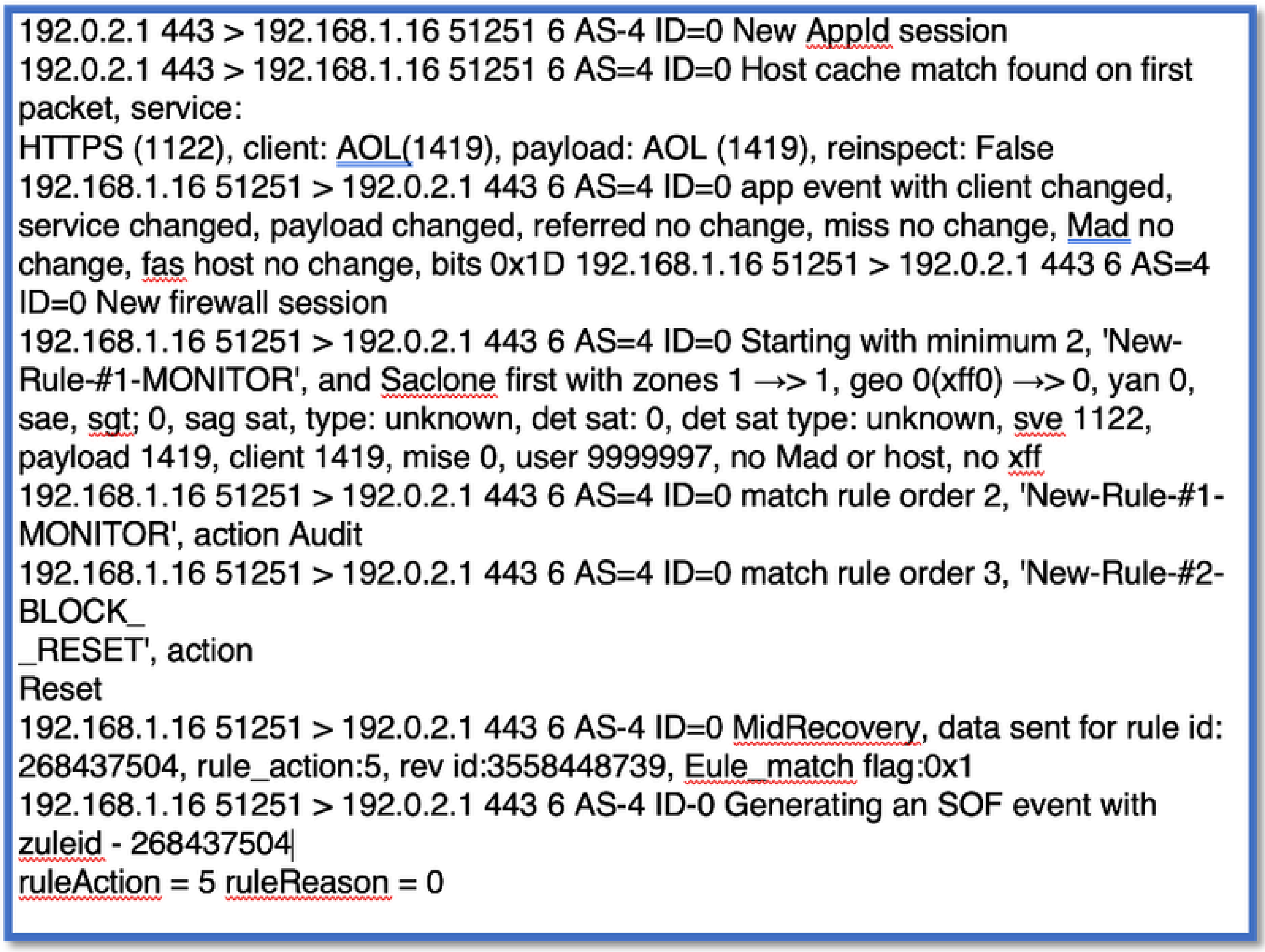
Overview of Diagnostics

- New logs are added in system support application identification debug to indicate if any applications are found by the 1st packet detection API.
- The logs also show if the user chose reinspection of traffic.
- Contents of the lua detector file uploaded by the user can be found on the FTD under /var/sf/appid/custom/lua/<UUID> .
- Any errors in the lua file are dumped on the FTD in the /var/log/messages file at the time of activating the detector.

CLI: system support application-identification-debug

<#root>

```
192.0.2.1 443 -> 192.168.1.16 51251 6 AS=4 ID=0 New AppId session

192.0.2.1 443 -> 192.168.1.16 51251 6 AS=4 ID=0 Host cache match found on first packet, service: HTTPS(1


192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 app event with client changed, service changed, payload
192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 New firewall session
192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 Starting with minimum 2, 'New-Rule-#1-MONITOR', and Src
192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 match rule order 2, 'New-Rule-#1-MONITOR', action Audit

192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 match rule order 3, 'New-Rule-#2-BLOCK_RESET', action Re


192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 MidRecovery data sent for rule id: 268437504, rule_acti
192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 Generating an SOF event with rule_id = 268437504 ruleAc

192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 reset action
```



```
192.0.2.1 443 > 192.168.1.16 51251 6 AS-4 ID=0 New AppId session
192.0.2.1 443 > 192.168.1.16 51251 6 AS=4 ID=0 Host cache match found on first
packet, service:
HTTPS (1122), client: AOL(1419), payload: AOL (1419), reinspect: False
192.168.1.16 51251 > 192.0.2.1 443 6 AS=4 ID=0 app event with client changed,
service changed, payload changed, referred no change, miss no change, Mad no
change, fas host no change, bits 0x1D 192.168.1.16 51251 > 192.0.2.1 443 6 AS=4
ID=0 New firewall session
192.168.1.16 51251 > 192.0.2.1 443 6 AS=4 ID=0 Starting with minimum 2, 'New-
Rule-#1-MONITOR', and Saclone first with zones 1 →> 1, geo 0(xff0) →> 0, yan 0,
sae, sgt; 0, sag sat, type: unknown, det sat: 0, det sat type: unknown, sve 1122,
payload 1419, client 1419, mise 0, user 9999997, no Mad or host, no xff
192.168.1.16 51251 > 192.0.2.1 443 6 AS=4 ID=0 match rule order 2, 'New-Rule-#1-
MONITOR', action Audit
192.168.1.16 51251 > 192.0.2.1 443 6 AS=4 ID=0 match rule order 3, 'New-Rule-#2-
BLOCK_
_RESET', action
Reset
192.168.1.16 51251 > 192.0.2.1 443 6 AS-4 ID=0 MidRecovery, data sent for rule id:
268437504, rule_action:5, rev id:3558448739, Eule_match flag:0x1
192.168.1.16 51251 > 192.0.2.1 443 6 AS-4 ID-0 Generating an SOF event with
zuleid - 268437504
ruleAction = 5 ruleReason = 0
```

Location of AppID Lua Detectors Content

To confirm if the Lua Detector with this new API exists on the Device/FTD you can look to see if the addHostFirstPktApp API is being used in the 2 application detector folders:

1. VDB AppID detectors -/var/sf/appid/odp/lua

2. Custom Detectors -/var/sf/appid/custom/lua

For example:grep addHostFirstPktApp * in each folder.

Sample Issues:

1. Issue: Custom Lua detector not activated on FMC.

   Location to check: /var/sf/appid/custom/lua/

   Expected result: One file for every custom app detector activated on the FMC must exist here. Verify contents match the uploaded lua file.

2. Issue: The uploaded lua detector file has errors.

   File to check: /var/log/messages on FTD

   Error log:

<#root>

```
Dec 18 14:17:49 intel-x86-64 SF-IMS[15741]:

Error - appid: can not set env of Lua detector /ngfw/var/sf/appid/custom/lua/6698fbd6-7ede-11ed-972c-d12
```

Troubleshooting Steps

Problem: Applications not correctly identified for traffic going to the user-defined IP address and port.

Steps to troubleshoot:

- Verify the lua detector is correctly defined and activated on the FTD.
    - Verify the contents of the lua file on the FTD and check no errors are seen on activating.
- Check the destination IP, port and protocol of 1st packet in the traffic session.
    - It can match the values defined in the lua detector.
- Check the system-support-application-identification-debug.
    - Look for the line `Host cache match found on first packet.` If that is missing, it indicates no match was found by the API.

Limitations Details, Common Problems, and Workarounds

In 7.4, there is no UI to use the API. UI support would be added in future releases.

Revision History

Revision	Publish Date	Comments
1.0	18-Jul-2024	Initial Release