Troubleshoot Datapath Handling by UTD and URL-Filtering

Contents

Introduction

This document describes how to troubleshoot Unified Threat Defense (UTD) also known as Snort and Uniform Resource Locator (URL) Filtering on IOS® XE WAN Edges routers.

Background Information

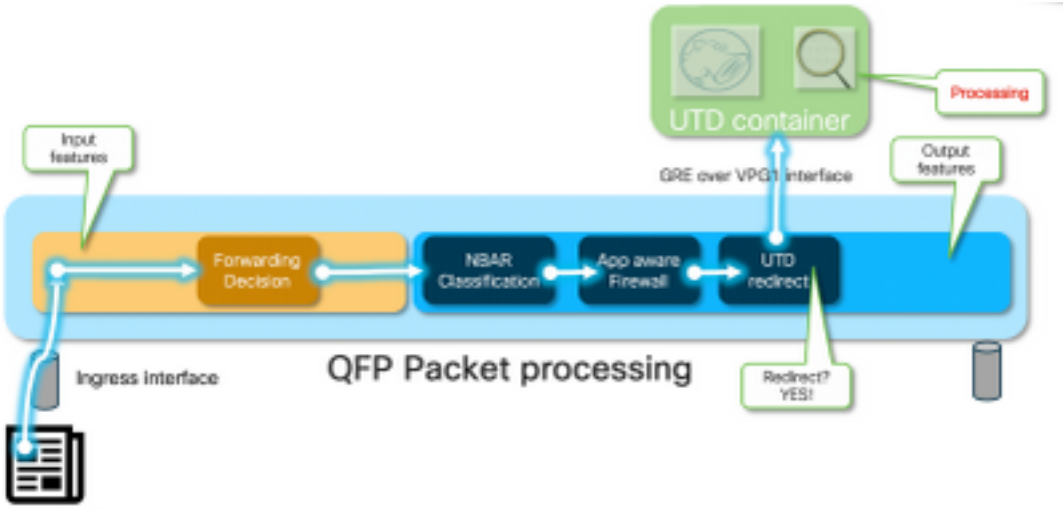
Snort is the most widely deployed Intrusion Prevention System (IPS) in the world. Since 2013, The company which created a commercial version of the Snort software, Sourcefire is acquired by Cisco. Starting from 16.10.1 IOS® XE SD-WAN software, UTD/URF-Filtering containers have been added to the Cisco SD-WAN solution.

The container registers to the IOS®XE router by using the app-nav framework. The explanation of that process is beyond the scope of this document.

Datapath High-Level View

At a high level, the datapath looks like this:

From the LAN/WAN to the Container



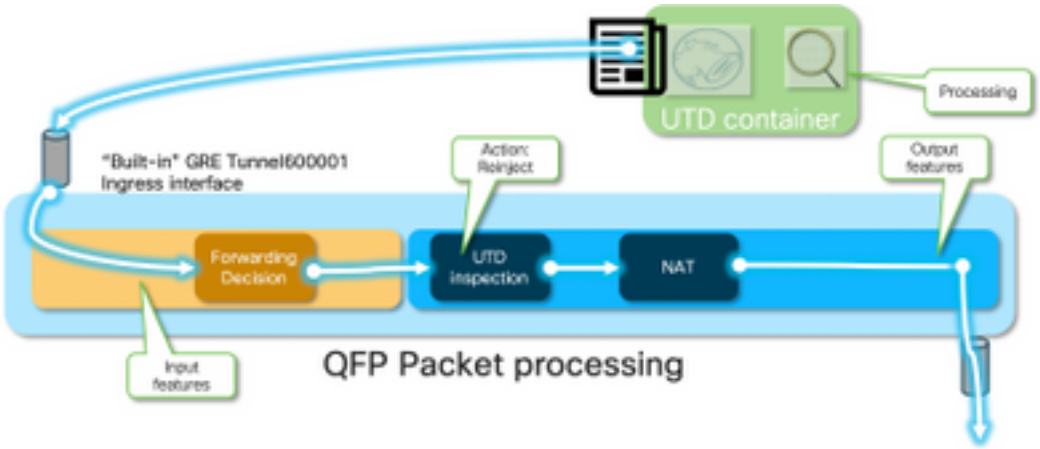
Traffic comes from the LAN side. Since IOS® XE knows the container is in a healthy state, it diverts the traffic to the UTD container. The diversion uses the VirtualPortGroup1 interface as the egress interface, which encapsulates the packet inside of a Generic Routing Encapsulation (GRE) tunnel.

The router performs "PUNT" action using cause :64 (Service Engine packet)" and sends the traffic towards the Route Processor (RP). A punt header is added and the packet is sent to the container using an internal egress interface towards the container "[internal0/0/svc_eng:0]"

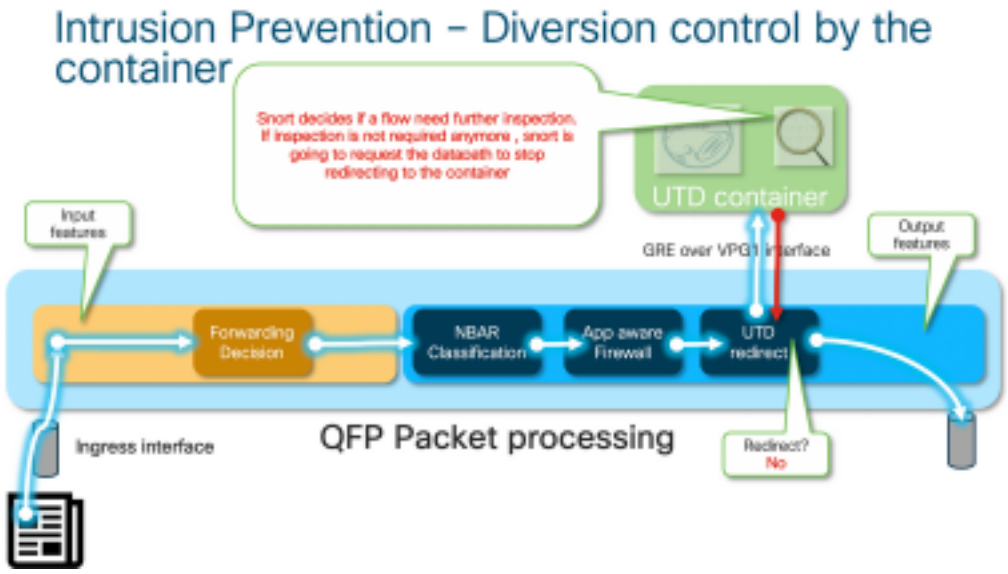
At this stage, Snort leverages its preprocessors and rule-sets. The packet can be dropped or forwarded based on the processing results.

From the Container to LAN/WAN

Assuming the traffic is not supposed to be dropped, the packet is forwarded back to the router after UTD proccessing. It appears on the Quantum Flow Processor (QFP) as coming from Tunnel6000001. Then it's processed by the router and must be (hopefully) routed towards the WAN interface.



Container controls the diversion result in the UTD inspection in the IOS® XE datapath.



For instance, with HTTPS flow, the preprocessors are interested to see the server Hello / Client Hello packets with TLS negotiation. Afterwards the flow is not redirected since there is little value in inspecting TLS encrypted traffic.

Datapath Deep Dive

From a packet-tracer standpoint, those set of action is going to be seen (192.168.16.254 is a web client):

```
debug platform condition ipv4 192.168.16.254/32 both
debug platform condition start
debug platform packet-trace packet 256 fia-trace data-size 3000
```

Ingress Packet from LAN or WAN Side towards the Container

In this particular scenario, the traced packet comes from the LAN. From a redirection standpoint, there are relevant differences if the flow comes from LAN or WAN.

The client tries to access www.cisco.com on HTTPS

```
cedge6#show platform packet-trace packet 14
Packet: 14          CBUG ID: 3849209
Summary
  Input     : GigabitEthernet2
  Output    : internal0/0/svc_eng:0
  State     : PUNT 64  (Service Engine packet)
  Timestamp
    Start   : 1196238208743284 ns (05/08/2019 10:50:36.836575 UTC)
    Stop    : 1196238208842625 ns (05/08/2019 10:50:36.836675 UTC)
Path Trace
  Feature: IPV4(Input)
    Input       : GigabitEthernet2
    Output      : <unknown>
    Source      : 192.168.16.254
    Destination : 203.0.113.67
    Protocol    : 6 (TCP)
      SrcPort   : 35568
      DstPort   : 443
  Feature: DEBUG_COND_INPUT_PKT
    Entry       : Input - 0x8177c67c
    Input       : GigabitEthernet2
    Output      : <unknown>
    Lapsed time : 2933 ns
<snip>
```

Traffic matching the condition is traced ingress on interface GigabitEthernet2.

```
  Feature: UTD Policy (First FIA)
    Action         : Divert
    Input interface : GigabitEthernet2
    Egress interface: GigabitEthernet3
  Feature: OUTPUT_UTD_FIRST_INSPECT
    Entry       : Output - 0x817cc5b8
    Input       : GigabitEthernet2
    Output      : GigabitEthernet3
    Lapsed time : 136260 ns
  Feature: UTD Inspection
    Action         : Divert             <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
    Input interface : GigabitEthernet2
    Egress interface: GigabitEthernet3
  Feature: OUTPUT_UTD_FINAL_INSPECT
    Entry       : Output - 0x817cc5e8
    Input       : GigabitEthernet2
    Output      : GigabitEthernet3
    Lapsed time : 43546 ns
<snip>
```

On the egress Feature Invocation Array (FIA) of the egress interface, UTD FIA decided to divert this packet to the container.

```
  Feature: IPV4_OUTPUT_LOOKUP_PROCESS_EXT
    Entry       : Output - 0x81781bb4
    Input       : GigabitEthernet2
    Output      : Tunnel6000001
<removed>
  Feature: IPV4_OUTPUT_LOOKUP_PROCESS_EXT
    Entry       : Output - 0x81781bb4
    Input       : GigabitEthernet2
    Output      : Tunnel6000001
<removed>
  Feature: IPV4_INPUT_LOOKUP_PROCESS_EXT
    Entry       : Output - 0x8177c698
```

```
    Input       : Tunnel6000001
    Output      : VirtualPortGroup1
    Lapsed time : 880 ns
<snip>
```

The packet is placed on the default tunnel Tunnel600001 and gets routed across the VPG1 interface. At this stage, the original packet is GRE encapsulated.

```
  Feature: OUTPUT_SERVICE_ENGINE
    Entry       : Output - 0x817c6b10
    Input       : Tunnel6000001
    Output      : internal0/0/svc_eng:0
    Lapsed time : 15086 ns
<removed>
  Feature: INTERNAL_TRANSMIT_PKT_EXT
    Entry       : Output - 0x8177c718
    Input       : Tunnel6000001
    Output      : internal0/0/svc_eng:0
    Lapsed time : 43986 ns
```

The packet is transmitted internally to the container.

> **Note**: Further information in this section about container internals is provided for information purposes only. The UTD container is not accessible via the normal CLI interface.

Going deeper in the router itself, the traffic arrives in an internal VRF on the route processor interface eth2:

```
[cedge6:/]$ chvrf utd  ifconfig
eth0      Link encap:Ethernet  HWaddr 54:0e:00:0b:0c:02
          inet6 addr: fe80::560e:ff:fe0b:c02/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1375101 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1366614 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:96520127 (92.0 MiB)  TX bytes:96510792 (92.0 MiB)

eth1      Link encap:Ethernet  HWaddr 00:1e:e6:61:6d:ba
          inet addr:192.168.1.2  Bcast:192.168.1.3  Mask:255.255.255.252
          inet6 addr: fe80::21e:e6ff:fe61:6dba/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:2000  Metric:1
          RX packets:1069 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2001 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:235093 (229.5 KiB)  TX bytes:193413 (188.8 KiB)

eth2      Link encap:Ethernet  HWaddr 00:1e:e6:61:6d:b9
          inet addr:192.0.2.2  Bcast:192.0.2.3  Mask:255.255.255.252
          inet6 addr: fe80::21e:e6ff:fe61:6db9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:2000  Metric:1
          RX packets:2564233 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2564203 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:210051658 (200.3 MiB)  TX bytes:301467970 (287.5 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
```

```
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Eth0 is a Transport Inter Process Communication (TIPC) interface connected to the IOSd process. The OneP channel runs over it for passing configurations and notifications back and forth between the IOSd and UTD container.

From what you are concerned, "eth2 [container interface]" is bridged to "VPG1 [192.0.2.1/192.168.2.2]" are the addresses pushed by vManage to the IOS-XE and container.

If you run **tcpdump**, you can see the GRE encapsulated traffic going to the container. The GRE encapsulation includes a VPATH header.

```
[cedge6:/]$  chvrf utd  tcpdump -nNvvvXi eth2    not udp
tcpdump: listening on eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
06:46:56.350725 IP (tos 0x0, ttl 255, id 35903, offset 0, flags [none], proto GRE (47), length
121)
    192.0.2.1 > 192.0.2.2: GREv0, Flags [none], length 101
gre-proto-0x8921
0x0000:  4500 0079 8c3f 0000 ff2f ab12 c000 0201  E..y.?.../......
0x0010:  c000 0202 0000 8921 4089 2102 0000 0000  .......!@.!.....
0x0020:  0000 0000 0300 0001 0000 0000 0000 0000  ................
0x0030:  0004 0800 e103 0004 0008 0000 0001 0000  ................
0x0040:  4500 0039 2542 4000 4011 ce40 c0a8 10fe  E..9%B@.@..@....
0x0050:  ad26 c864 8781 0035 0025 fe81 cfa8 0100  .&.d...5.%......
0x0060:  0001 0000 0000 0000 0377 7777 0363 6e6e  .........www.cnn
0x0070:  0363 6f6d 0000 0100 01                   .com.....
```

Ingress Packet from Container towards LAN or WAN Side

After Snort processing (assuming traffic is not to be dropped), it is reinjected back into to the QFP forwarding path.

```
cedge6#show platform packet-trace packet 15
Packet: 15          CBUG ID: 3849210
Summary
  Input     : Tunnel6000001
  Output    : GigabitEthernet3
  State     : FWD
```

Tunnel600001 is the egress interface from the container.

```
  Feature: OUTPUT_UTD_FIRST_INSPECT_EXT
    Entry       : Output - 0x817cc5b8
    Input       : GigabitEthernet2
    Output      : GigabitEthernet3
    Lapsed time : 2680 ns
  Feature: UTD Inspection
    Action         : Reinject
    Input interface : GigabitEthernet2
    Egress interface: GigabitEthernet3
  Feature: OUTPUT_UTD_FINAL_INSPECT_EXT
    Entry       : Output - 0x817cc5e8
    Input       : GigabitEthernet2
```

```
    Output      : GigabitEthernet3
    Lapsed time : 12933 ns
```

Since the traffic has already been inspected, the router knows this is a re-injection.

```
  Feature: NAT
    Direction   : IN to OUT
    Action      : Translate Source
    Steps       :
    Match id    : 1
    Old Address : 192.168.16.254 35568
    New Address : 172.16.16.254  05062
```

Traffic gets NATed and goes out towards the Internet.

```
  Feature: MARMOT_SPA_D_TRANSMIT_PKT
    Entry       : Output - 0x8177c838
    Input       : GigabitEthernet2
    Output      : GigabitEthernet3
    Lapsed time : 91733 ns
```

UTD flow logging integration with Packet-trace

IOS-XE 17.5.1 added UTD flow logging integration with packet-trace, where the path-trace output will include a UTD verdict. Verdict can be one of the following, for example:

- the packet that UTD decides to block/alert for Snort
- allow/drop for URLF
- block/allow for AMP

For packets that do not have the UTD verdict information, no flow logging information is logged. Also note there is no logging of IPS/IDS pass/allow verdict due to potential negative performance impact.

To enable flow-logging integration, use the CLI Add-On template with:

```
utd engine standard multi-tenancy
utd global
  flow-logging all
```

Example output for different verdicts:

URL Lookup Timeout:

```
show platform packet-trace  pack all | sec Packet: | Feature: UTD Inspection
Packet: 31          CBUG ID: 12640
Feature: UTD Inspection
   Action                   : Reinject
   Input interface          : GigabitEthernet2
   Egress interface         : GigabitEthernet3
   Flow-Logging Information :
    URLF Policy ID          : 1
    URLF Action             : Allow(1)
    URLF Reason             : URL Lookup Timeout(8)
```

URLF reputation and verdict Allow:

```
Packet: 21          CBUG ID: 13859
  Feature: UTD Inspection
    Action                   : Reinject
    Input interface          : GigabitEthernet3
    Egress interface         : GigabitEthernet2
    Flow-Logging Information :
     URLF Policy ID          : 1
     URLF Action             : Allow(1)
     URLF Reason             : No Policy Match(4)
     URLF Category           : News and Media(63)
     URLF Reputation         : 81
```

URLF reputation and verdict Block:

```
Packet: 26          CBUG ID: 15107
  Feature: UTD Inspection
    Action                   : Reinject
    Input interface          : GigabitEthernet3
    Egress interface         : GigabitEthernet2
    Flow-Logging Information :
     URLF Policy ID          : 1
     URLF Action             : Block(2)
     URLF Reason             : Category/Reputation(3)
     URLF Category           : Social Network(14)
     URLF Reputation         : 81
```

Prerequesite:

Checking if UTD version is compatible with IOS XE

```
cedge7#sh utd eng sta ver
UTD Virtual-service Name: utd
IOS-XE Recommended UTD Version: 1.10.33_SV2.9.16.1_XEmain
IOS-XE Supported UTD Regex: ^1\.10\.([0-9]+)_SV(.*)_XEmain$
UTD Installed Version: 1.0.2_SV2.9.16.1_XE17.5 (UNSUPPORTED)
```

If "UNSUPPORTED" is displayed, the container upgrade is required as first step prior starting troubleshooting.

Check for valid nameserver configuration in the container

Some of the security services such as AMP and URLF will require the UTD container to be able to resolve names for the cloud service providers, so the UTD container must have valid nameserver configurations. This can be verified by checking the resolv.conf file for the container under the system shell:

```
cedge:/harddisk/virtual-instance/utd/rootfs/etc]$ more resolv.conf
nameserver 208.67.222.222
nameserver 208.67.220.220
nameserver 8.8.8.8
```

Problem 1

Per design, Unified Thread Defense must be configured altogether with the Direct Internet Access use case (DIA). The container will try to resolve **api.bcti.brightcloud.com** in order to query URL

reputations and categories. In this example, none of the inspected URLs are blocked even if the proper config gets applied

Troubleshoot

Always look at the container log file.

```
cedge6#app-hosting move appid utd log to bootflash:
Successfully moved tracelog to bootflash:
iox_utd_R0-0_R0-0.18629_0.20190501005829.bin.gz
```

That copies the log file on the flash itself.

Displaying the log can be achieved with the command:

```
cedge6#  more /compressed iox_utd_R0-0_R0-0.18629_0.20190501005829.bin.gz
```

Displaying the log reveals:

```
2019-04-29 16:12:12 ERROR: Cannot resolve host api.bcti.brightcloud.com: Temporary failure in
name resolution
2019-04-29 16:17:52 ERROR: Cannot resolve host api.bcti.brightcloud.com: Temporary failure in
name resolution
2019-04-29 16:23:32 ERROR: Cannot resolve host api.bcti.brightcloud.com: Temporary failure in
name resolution
2019-04-29 16:29:12 ERROR: Cannot resolve host api.bcti.brightcloud.com: Temporary failure in
name resolution
2019-04-29 16:34:52 ERROR: Cannot resolve host api.bcti.brightcloud.com: Temporary failure in
name resolution
2019-04-29 16:40:27 ERROR: Cannot resolve host api.bcti.brightcloud.com: Temporary failure in
name resolution
```

By default vManage provisions a container that uses OpenDNS server [208.67.222.222 and 208.67.220.220]

Root Cause

Domain Name System (DNS) traffic to resolve **api.bcti.brightcloud.com** is dropped somewhere in the path between the container and the umbrella DNS servers. Always ensure both DNS are reachable.

Problem 2

In a scenario where Computer and Internet Info category websites are supposed to be blocked, http request to www.cisco.com is properly dropped while HTTPS requests are not.

Troubleshoot

As explained before, the traffic is punted to the container. When this flow is encapsulated in the GRE header, software appends as well a VPATH header. Leveraging this header, the system allows to pass a debug condition to the container itself. This means UTD containers are well serviceable.

In this scenario, the client IP address is 192.168,16.254. Let's troubleshoot the snort handling by the container itself for the traffic that comes from my client.

```
debug platform condition ipv4 192.168.16.254/32 both
debug platform condition feature utd controlplane submode serviceplane-web-filtering level
verbose
debug platform condition start
```

This set of commands instructs IOS-XE to mark traffic from or to 192.168.16.254. That enables debug-me flag to be passed to the container via the VPATH header

LSMPI punt header	Outer IP header (e.g. 192.0.2.x)	GRE header	vPath header (conditional debug flag is here)	Inner (original) IP packet

Snort debugs only that specific flow while others are processed normally.

At this stage, you can ask the user to trigger the traffic from the client towards www.cisco.com.

The next step would be to retrieve the logs:

```
app-hosting move appid utd log to bootflash:
```

In the case of HTTP traffic, the snort HTTP preprocessor discovers the URL In the get request.

```
2019-04-26 13:04:27.773:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 39540, p->dst_port
= 80
2019-04-26 13:04:27.793:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 80, p->dst_port =
39540
2019-04-26 13:04:27.794:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 39540, p->dst_port
= 80
2019-04-26 13:04:27.794:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 39540, p->dst_port
= 80
2019-04-26 13:04:27.794:(#1):SPP-URL-FILTERING got utmdata_p
2019-04-26 13:04:27.794:(#1):SPP-URL-FILTERING HTTP Callback, direction = 00000080

2019-04-26 13:04:27.795:(#1):SPP-URL-FILTERING White list regex match not enabled
2019-04-26 13:04:27.795:(#1):SPP-URL-FILTERING Black list regex match not enabled
2019-04-26 13:04:27.795:(#1):SPP-URL-FILTERING URL database Request: url_len = 12, msg overhead
12 url: www.cisco.com/ <<<<<<<
2019-04-26 13:04:27.795:(#1):SPP-URL-FILTERING Send to URL database: req_id=0x10480047
2019-04-26 13:04:27.795:(#1):SPP-URL-FILTERING Sent to URL database 24 bytes
2019-04-26 13:04:27.795:(#1):SPP-URL-FILTERING Send to URL database done, idx: 71, URL:
www.cisco.com/
2019-04-26 13:04:27.795:(#1):SPP-URL-FILTERING Received from URL database 24 bytes
2019-04-26 13:04:27.816:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 80, p->dst_port =
39540
2019-04-26 13:04:27.816:(#1):SPP-URL-FILTERING Found UTMData at 0x007f8d9ee80878, action =
0000000a
2019-04-26 13:04:27.816:(#1):SPP-URL-FILTERING Utm_verdictProcess: vrf_id 1, category 0x63,
score 81 <<<<<<<<<<<<<<<<<<<<<<<<<
2019-04-26 13:04:27.816:(#1):SPP-URL-FILTERING Category 0x3f
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
2019-04-26 13:04:27.816:(#1):SPP-URL-FILTERING index = 63, action = 1
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
2019-04-26 13:04:27.816:(#1):SPP-URL-FILTERING Blocking category = 0x3f
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

In case of https traffic, the destination DNS has been extracted from the server hello by the

HTTPS pre-processor

```
2019-05-01 00:56:18.870:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 35322, p->dst_port
= 443
2019-05-01 00:56:18.886:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 443, p->dst_port =
35322
2019-05-01 00:56:18.887:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 35322, p->dst_port
= 443
2019-05-01 00:56:18.887:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 35322, p->dst_port
= 443
2019-05-01 00:56:18.903:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 443, p->dst_port =
35322
2019-05-01 00:56:18.906:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 443, p->dst_port =
35322
2019-05-01 00:56:18.906:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 35322, p->dst_port
= 443
2019-05-01 00:56:18.907:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 443, p->dst_port =
35322
2019-05-01 00:56:18.907:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 443, p->dst_port =
35322
2019-05-01 00:56:18.907:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 443, p->dst_port =
35322

2019-05-01 00:56:18.908:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 443, p->dst_port =
35322
2019-05-01 00:56:18.908:(#1):SPP-URL-FILTERING utm_sslLookupCallback
2019-05-01 00:56:18.908:(#1):SPP-URL-FILTERING got utmdata_p
2019-05-01 00:56:18.909:(#1):SPP-URL-FILTERING White list regex match not enabled
2019-05-01 00:56:18.909:(#1):SPP-URL-FILTERING Black list regex match not enabled
2019-05-01 00:56:18.909:(#1):SPP-URL-FILTERING URL database Request: url_len = 11, msg overhead
12 url: www.cisco.com  <<<<<<<<
2019-05-01 00:56:18.909:(#1):SPP-URL-FILTERING Send to URL database: req_id=0x10130012
2019-05-01 00:56:18.909:(#1):SPP-URL-FILTERING Sent to URL database 23 bytes
2019-05-01 00:56:18.909:(#1):SPP-URL-FILTERING Send to URL database done, idx: 18, URL:
www.cisco.com
2019-05-01 00:56:18.909:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 443, p->dst_port =
35322
2019-05-01 00:56:18.910:(#1):SPP-URL-FILTERING Found UTMData at 0x007f1d9c479640, action =
00000008
2019-05-01 00:56:18.910:(#1):SPP-URL-FILTERING Verdict very late, in queryig state 2, idx=18
2019-05-01 00:56:18.910:(#1):SPP-URL-FILTERING UTM preprocessor p->src_port = 443, p->dst_port =
35322

2019-05-01 00:56:18.910:(#1):SPP-URL-FILTERING Found UTMData at 0x007f1d9c479640, action =
00000009
2019-05-01 00:56:18.910:(#1):SPP-URL-FILTERING Verdict very late, in queryig state 2, idx=18
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
2019-05-01 00:56:18.909:(#1):SPP-URL-FILTERING Received from URL database 24 bytes
```

Here you don't see the blocking page being triggered since the software does not report the results from the webroot query.

Root Cause

CSCvo77664 "UTD URL filtering for category lookup is failing with webroot lookup failure" is about the traffic is being leaked when software has no response to our URL verdict request yet.

Problem 3

In this scenario, intermittently, web browsing sessions that should be allowed by URL-Filtering [because of their classification] are dropped. As example, accessing www.google.com is randomly not possible even if the category "web search engine" is allowed.

Troubleshoot

Step1: Gathering general statistics

> **Note** This command output is reset every 5 minutes

```
cedge7#show utd engine standard statistics internal
*************Engine #1*************
<removed> ================================================================================ HTTP
Inspect - encodings (Note: stream-reassembled packets included): <<<<<<<<<< generic layer7 HTTP
statistics POST methods: 0 GET methods: 7 HTTP Request Headers extracted: 7 HTTP Request Cookies
extracted: 0 Post parameters extracted: 0 HTTP response Headers extracted: 6 HTTP Response
Cookies extracted: 0 Unicode: 0 Double unicode: 0 Non-ASCII representable: 0 Directory
traversals: 0 Extra slashes ("//"): 0 Self-referencing paths ("./"): 0 HTTP Response Gzip
packets extracted: 0 Gzip Compressed Data Processed: n/a Gzip Decompressed Data Processed: n/a
Http/2 Rebuilt Packets: 0 Total packets processed: 13 <removed>
================================================================================ SSL
Preprocessor: <<<<<<<<<< generic layer7 SSL statistics SSL packets decoded: 38 Client Hello: 8
Server Hello: 8 Certificate: 2 Server Done: 6 Client Key Exchange: 2 Server Key Exchange: 2
Change Cipher: 10 Finished: 0 Client Application: 2 Server Application: 11 Alert: 0 Unrecognized
records: 11 Completed handshakes: 0 Bad handshakes: 0 Sessions ignored: 4 Detection disabled: 1

<removed> UTM Preprocessor Statistics < URL filtering statistics including ----------------------
------ URL Filter Requests Sent: 11 URL Filter Response Received: 5 Blacklist Hit Count: 0
Whitelist Hit Count: 0 Reputation Lookup Count: 5 Reputation Action Block: 0 Reputation Action
Pass: 5 Reputation Action Default Pass: 0 Reputation Action Default Block: 0 Reputation Score
None: 0 Reputation Score Out of Range: 0 Category Lookup Count: 5 Category Action Block: 0
Category Action Pass: 5 Category Action Default Pass: 0 Category None: 0 UTM Preprocessor
Internal Statistics ---------------------------------- Total Packets Received: 193 SSL Packet
Count: 4 Action Drop Flow: 0 Action Reset Session: 0 Action Block: 0 Action Pass: 85 Action
Offload Session: 0 Invalid Action: 0 No UTM Tenant Persona: 0 No UTM Tenant Config: 0 URL Lookup
Response Late: 4 <<<<< Explanation below URL Lookup Response Very Late: 64 <<<<< Explanation
below URL Lookup Response Extremely Late: 2 <<<<< Explanation below Response Does Not Match
Session: 2 <<<<< Explanation below No Response When Freeing Session: 1 First Packet Not From
Initiator: 0 Fail Open Count: 0 Fail Close Count : 0 UTM Preprocessor Internal Global Statistics
---------------------------------------- Domain Filter Whitelist Count: 0 utmdata Used Count:
11 utmdata Free Count: 11 utmdata Unavailable: 0 URL Filter Response Error: 0 No UTM Tenant Map:
0 No URL Filter Configuration : 0 Packet NULL Error : 0 URL Database Internal Statistics -------
------------------------ URL Database Not Ready: 0 Query Successful: 11 Query Successful from
Cloud: 6 <<< 11 queries were succesful but 6 only are queried via brightcloud. 5 (11-6) queries
are cached Query Returned No Data: 0 <<<<<< errors Query Bad Argument: 0 <<<<<< errors Query
Network Error: 0 <<<<<< errors URL Database UTM disconnected: 0 URL Database request failed: 0
URL Database reconnect failed: 0 URL Database request blocked: 0 URL Database control msg
response: 0 URL Database Error Response: 0
================================================================================ Files processed:
none ================================================================================
```

- "late request" - represents the HTTP GET or the HTTPS client/server certificate [where SNI / DN can be extracted for lookup. Late request are forwarded.
- "very late requests" - means that some sort of session drop counter where further packets in the flow are dropped until the router receives an URL verdict from Brightcloud. In other words, anything after the initial HTTP GET or the remaining of the SSL flow will be dropped until a verdict is received.

- "extremely late requests" - when the session query to Brightcloud has been reset without providing a verdict. The session will timeout after 60 seconds for version < 17.2.1. From 17.2.1 onwards, the querying session to Brightcloud will timeout after 2 seconds. [via CSCvr98723 UTD: Timeout URL requests after two seconds]

In this scenario, we see global counters that highlight an unhealthy situation.

Step2: Looking at the application log file

The Unified Thread Detection software is going to record events in the application log file.

```
cedge6#app-hosting move appid utd log to bootflash:
Successfully moved tracelog to bootflash:
iox_utd_R0-0_R0-0.18629_0.20190501005829.bin.gz
```

That extracts the container application log file and save it on the flash itself.

Displaying the log can be achieved with the command:

```
cedge6#  more /compressed iox_utd_R0-0_R0-0.18629_0.20190501005829.bin.gz
```

> **Note:** In IOS-XE software version 20.6.1 and later, it's no longer required to manually move the UTD application log. These logs can now be viewed using the standard command **show logging process vman module utd**

Displaying the log reveals:

```
.....
2020-04-14 17:47:57.504:(#1):SPP-URL-FILTERING txn_id miss match verdict txn_id 245 , utmdata
txn_id 0 2020-04-14 17:47:57.504:(#1):SPP-URL-FILTERING txn_id miss match verdict txn_id 248 ,
utmdata txn_id 0 2020-04-14 17:47:57.504:(#1):SPP-URL-FILTERING txn_id miss match verdict txn_id
249 , utmdata txn_id 0 2020-04-14 17:47:57.504:(#1):SPP-URL-FILTERING txn_id miss match verdict
txn_id 250 , utmdata txn_id 0 2020-04-14 17:47:57.660:(#1):SPP-URL-FILTERING txn_id miss match
verdict txn_id 251 , utmdata txn_id 0 2020-04-14 17:47:57.660:(#1):SPP-URL-FILTERING txn_id miss
match verdict txn_id 254 , utmdata txn_id 0 2020-04-14 17:47:57.660:(#1):SPP-URL-FILTERING
txn_id miss match verdict txn_id 255 , utmdata txn_id 0 2020-04-14 17:48:05.725:(#1):SPP-URL-
FILTERING txn_id miss match verdict txn_id 192 , utmdata txn_id 0 2020-04-14
17:48:37.629:(#1):SPP-URL-FILTERING txn_id miss match verdict txn_id 208 , utmdata txn_id 0
2020-04-14 17:49:55.421:(#1):SPP-URL-FILTERING txn_id miss match verdict txn_id 211 , utmdata
txn_id 0 2020-04-14 17:51:40 ERROR: Cannot send to host api.bcti.brightcloud.com: Connection
timed out 2020-04-14 17:52:21 ERROR: Cannot send to host api.bcti.brightcloud.com: Connection
timed out 2020-04-14 17:52:21 ERROR: Cannot send to host api.bcti.brightcloud.com: Connection
timed out 2020-04-14 17:53:56 ERROR: Cannot send to host api.bcti.brightcloud.com: Connection
timed out 2020-04-14 17:54:28 ERROR: Cannot send to host api.bcti.brightcloud.com: Connection
timed out 2020-04-14 17:54:29 ERROR: Cannot send to host api.bcti.brightcloud.com: Connection
timed out 2020-04-14 17:54:37 ERROR: Cannot send to host api.bcti.brightcloud.com: Connection
timed out
....
```

- "ERROR: Cannot send to host api.bcti.brightcloud.com" - means the querying session to Brightcloud has been timing out [60 seconds < 17.2.1 / 2 seconds >= 17.2.1]. This is the sign of a bad connectivity to Brightcloud.
  In order to demonstrate the issue, the use of EPC [Embedded Packet Capture] would allow to visualise the connectivity issue.
- "SPP-URL-FILTERING txn_id miss match verdict" - This error condition requires a bit more

explanation. Brightcloud query is performed via a POST where a query ID is generated by the router

Problem 4

In this scenario, IPS is the only security feature enabled in UTD, and customer is experiencing problems with printer communication which is a TCP application.

Troubleshoot

To troubleshoot this datapath issue, first take the packet capture from the TCP host having the issue. The capture shows a successful TCP 3-way handshake, but subsequent data packets with TCP data seem to have been dropped by the cEdge router. Next enable packet-trace, which showed the following:

```
edge#show platform packet-trace summ
Pkt   Input                     Output                    State  Reason
0     Gi0/0/1                   internal0/0/svc_eng:0     PUNT   64  (Service Engine packet)
1     Tu2000000001              Gi0/0/2                   FWD
2     Gi0/0/2                   internal0/0/svc_eng:0     PUNT   64  (Service Engine packet)
3     Tu2000000001              Gi0/0/1                   FWD
4     Gi0/0/1                   internal0/0/svc_eng:0     PUNT   64  (Service Engine packet)
5     Tu2000000001              Gi0/0/2                   FWD
6     Gi0/0/1                   internal0/0/svc_eng:0     PUNT   64  (Service Engine packet)
7     Tu2000000001              Gi0/0/2                   FWD
8     Gi0/0/2                   internal0/0/svc_eng:0     PUNT   64  (Service Engine packet)
9     Gi0/0/2                   internal0/0/svc_eng:0     PUNT   64  (Service Engine packet)
```

The above output indicated packet number 8 and 9 have been diverted to the UTD engine but they did not get re-injected back into the forwarding path. Checking the UTD engine logging events also does not reveal any Snort signature drops. Next check the UTD internal statistics, which does reveal some packet drops due to the TCP normalizer:

```
edge#show utd engine standard statistics internal
<snip>
Normalizer drops:
           OUTSIDE_PAWS: 0
             AHEAD_PAWS: 0
           NO_TIMESTAMP: 4
                BAD_RST: 0
             REPEAT_SYN: 0
            WIN_TOO_BIG: 0
               WIN_SHUT: 0
                BAD_ACK: 0
             DATA_CLOSE: 0
          DATA_NO_FLAGS: 0
             FIN_BEYOND: 0
```

Root Cause

The root cause of the problem is due to misbehaving TCP stack on the printers. When the Timestamp option is negotiated during the TCP 3-way handshake, RFC7323 states a TCP MUST send the TSopt option in eevery non-<RST> packet. A closer examination of the packet capture will show the TCP data packets getting dropped do not have these options enabled. With the IOS-XE UTD implementation, Snort TCP normalizer with the block option is enabled irrespective of IPS

or IDS.

References

- Security Configuration Guide: Unified Threat Defense