Troubleshoot the Corrupted Ethernet Packet on Cisco Nexus 9000

Contents

Introduction

This document describes how to troubleshoot the corrupted ethernet packet on Cisco Nexus 9000 when a padding information is corrupted or malformed.

Background Information

The minimal size of an Ethernet frame is 64 bytes, no matter VLAN tag is present there or not.

The minimal Ethernet payload size is:

- 46 bytes if the VLAN tag is absent.
- 42 bytes if the VLAN tag is present.

You can verify this fact:

- On Wikipedia, section **Payload**: https://en.wikipedia.org/wiki/Ethernet_frame
- On the IEEE 802.3 standard (http://people.ee.duke.edu/~mbrooke/EE164.02/Spring_2004/group_2/index_files/8023.pdf), where the MAC frame format (without VLAN) is defined in section 3.1.1, page 39, and the elements of a tagged MAC frame is defined on page 43 section 3.5.

The minimal size of an ethernet packet is 64 bytes, no matter VLAN header is present there or not. The server is allowed to send a 64 bytes long packet that contains a VLAN, which you should accept and process correctly.

> **Note**: This behaviour is correctly handled by a catalyst 4500x not by Nexus 9k.

How is a Packet Processed by a Switch

Step 1. Receive a **VALID** 64 bytes Ethernet frame.

Step 2. Remove the Frame Check Sequence (FCS), so the packet becomes 60 bytes long.

Step 3. Remove the VLAN tag, so the packet becomes 56 bytes long.

Step 4. Add padding to make the packet 60 bytes long.

Step 5. It adds the FCS, making the packet 64 bytes long.

Padding should not get modified when a packet goes through cut-through switch.

Padding Modified with Tagged VLANs when Traffic traverses N9K

Instead of padding with zeroes, the packet is padded with garbage characters, in most of the cases it has no impact because checksums are not modified and so nobody uses these datas. However, if customers have a special usage and need to recompute the checksums, these garbage data leads to corruption of checksums in the end (other appliances, like NAT/load-balancers might see the issue too).

Device is a N9K 93120TX (was initially detected on a 9372TX though), version is latest NXOS 7.0(3)I2(2a).

Use Linux hosts with directly connected hardware to the N9K (no virtualization of any kind) here (1000base-T links).

Use this configuration:

```
interface Ethernet1/59

  switchport mode trunk

!

interface Ethernet1/60

  switchport mode trunk
```

```
linux configurations:

inet 10.2.1.1/24 brd 10.2.1.255 scope global eth1  <= native vlan

inet 10.1.1.1/24 brd 10.1.1.255 scope global eth1.100 <= taggued vlan 100
```

or

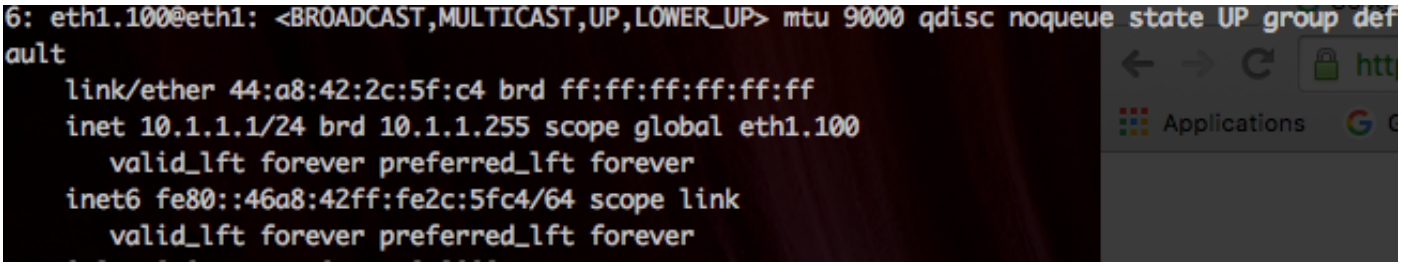
Just connect the windows host and send the tagged frames, they should trigger the problem. Moreover, ensure that the Network Interface Card (NIC) has the ability to tag the packet.

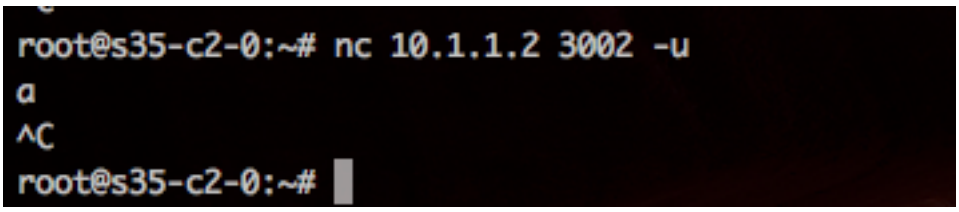
Switch adds the non-zero padding to the frames that passes through.

For Example: Host —— [Trunk] N9K [Trunk] —— Host

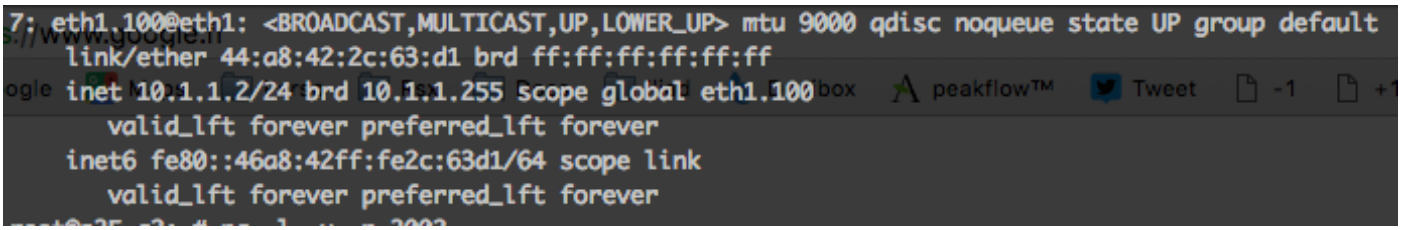
You can use netcat to send and receive the packets.

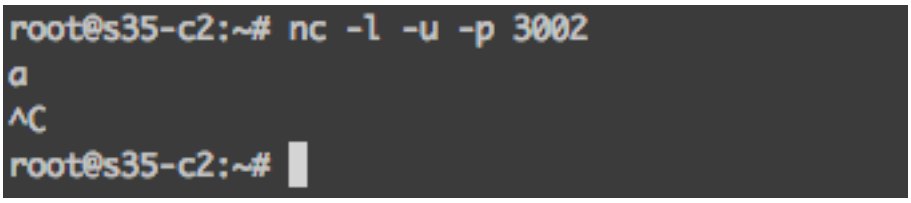
As shown in the image, it sends Side (VLAN 100 tagged), port e1/59 on the switch.



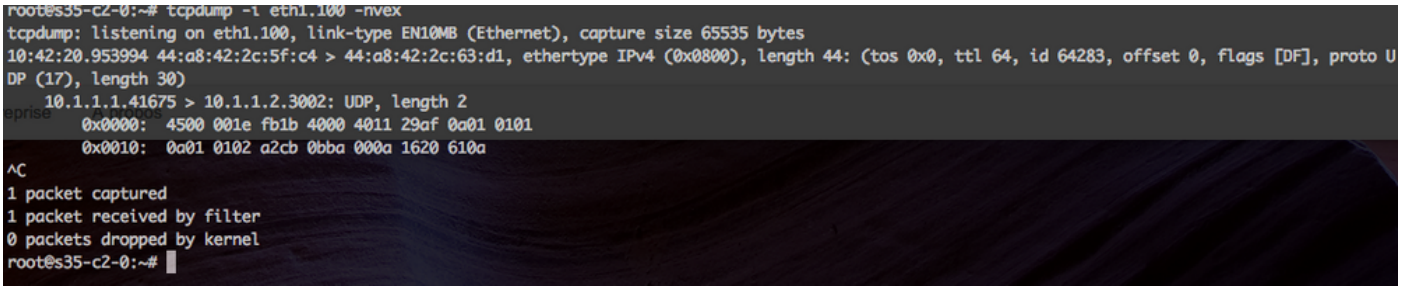


It receives Side (VLAN 100 tagged), port e1/60 on the switch, as shown in the image:

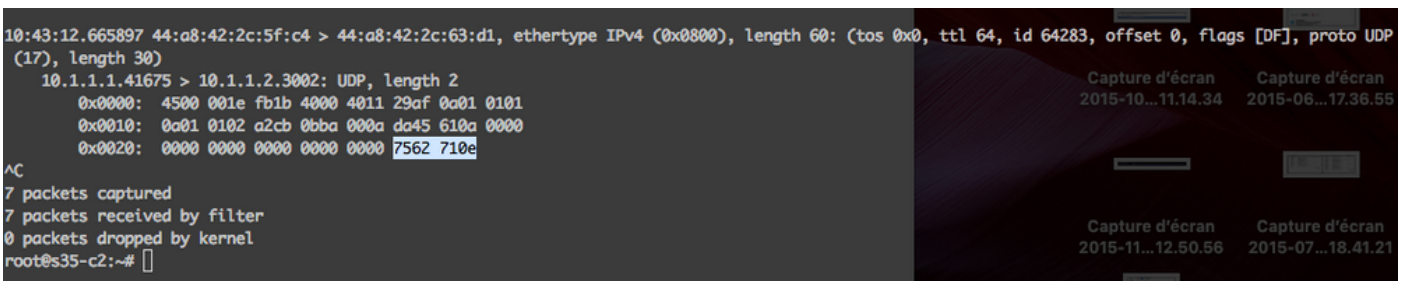




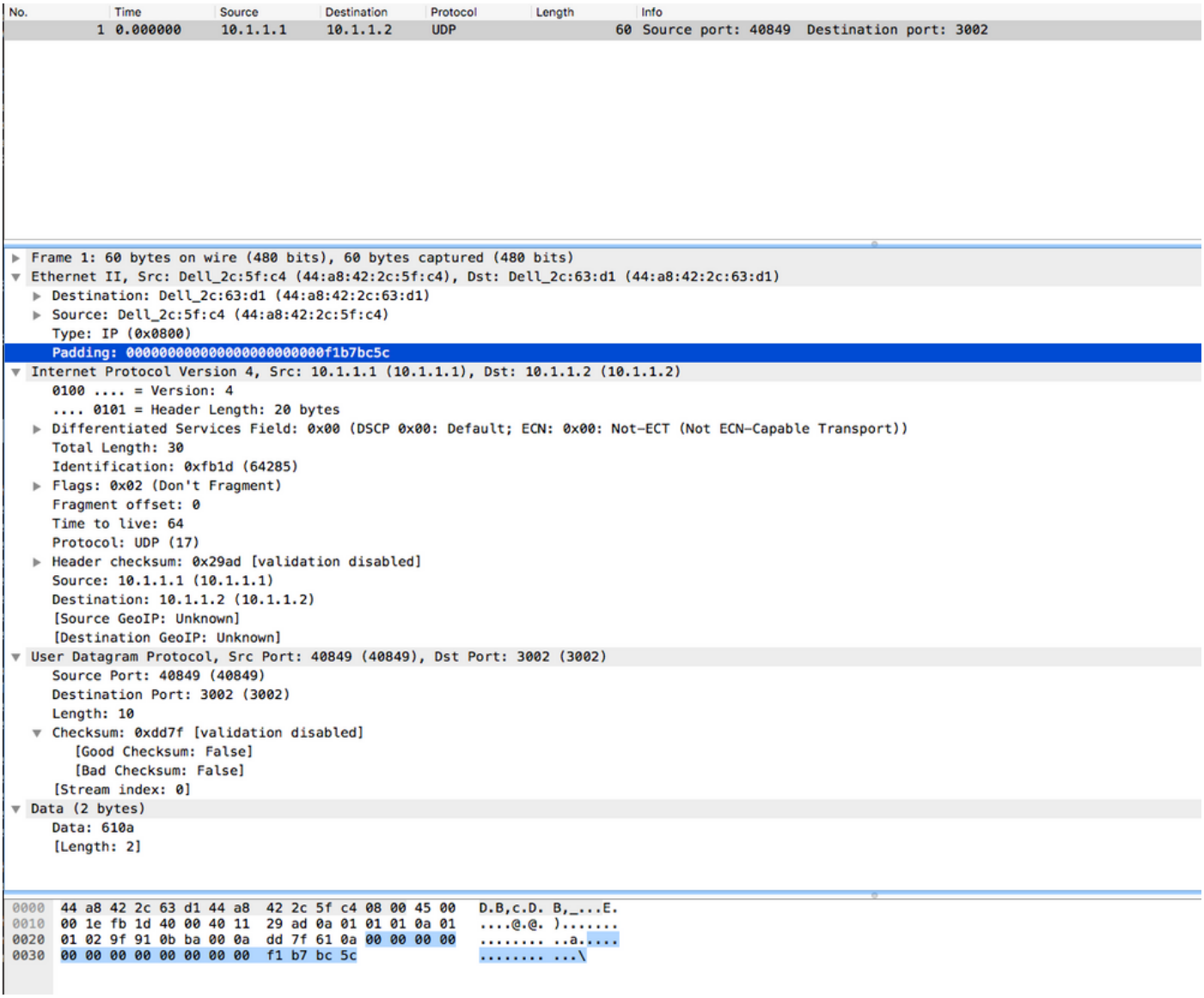
As shown in the image, the packet is transmitted.



The packet is received, as shown in the image:



As shown in the image, the wrong padding is highlighted.



This is also displayed with a packet analyzer (in another packet, the data is different from the previous screenshots but test and bug is identical),

Solution

The workaround is to disable buffer-boost on interface where we have this server connected.

```
C9396PX-1(config)# int et 1/7
C9396PX-1(config-if)# no buffer-boost
```

Related defect:

CSCva46849 60 Byte Frame with dot1q Header L2 switching on N9k