IRISA

INSTITUT DE RECHERCHE
EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042 RENNES CEDEX FRANCE
Tél.: 02 99 84 71 00 - Télex : UNIRISA 950 473 F
Télécopie : 02 99 84 71 71

Test Specification

SIIT/NAT-PT Conformance Test Suite

Technical Document V 2.2

IRISA (Europe)
TAHI Project (Japan)
TTA/IT Testing Laboratory (Korea)

Table Of Contents

1 Introduction

SIIT is a protocol translation mechanism at the edge of the network that allows communication between IPv6-only and IPv4-only nodes via protocol independent translation of IPv4 and IPv6 datagrams, requiring no state information for the session. The SIIT proposal assumes that V6 nodes are assigned a V4 address for communicating with V4 nodes.

NAT-PT provides transparent routing to end-nodes in V6 realm trying to communicate with end-nodes in V4 realm and vice versa. This is achieved using a combination of Network Address Translation and Protocol Translation. Moreover, NAT-PT uses a pool of V4 addresses for assignment to V6 nodes on a dynamic basis as sessions are initiated across V4-V6 boundaries. The NAT-PT mechanism uses the SIIT translation rules.

The SIIT Mechanism is described in **[RFC2765]**, Stateless IP/ICMP Translation Algorithm (SIIT), February 2000. The NAT-PT Mechanism is described in **[RFC2766]**, Network Address Translation - Protocol Translation (NAT-PT), February 2000

The relation of each NAT-PT technology is the following :

- **Traditional NAT-PT**

  Basic NAT-PT: Traditional-NAT-PT allows hosts within a V6 network to access hosts in the V4 network. In a traditional-NAT-PT, sessions are unidirectional, outbound from the V6 network.

  NAPT-PT: NAPT-PT is a variant of NAT-PT that translates transport identifiers such as TCP/UDP port numbers and ICMP identifiers in addition to IP header. As a consequence, NAPT-PT allows multiple IPv6 nodes to communicate with IPv4 nodes using a single public IPv4 address.

- ***Bi-Directional NAT-PT****:* bi-directional NAT-PT handles sessions in outbound and inbound directions.

A NAT-PT Box MUST take into account Traditional NAT-PT to be conform to the standards. It means that either Basic NAT-PT either NAPT-PT has to be implemented. Bi-directionnal NAT-PT is optional since the simplest form of NAT-PT is considered as only one way connectivity. Moreover because a NAT-PT Box acts as a single router, we also have to run conformance test suite for core protocol on this router.

This documents gives NAT-PT Conformance Tests developed by IRISA with technical inputs from:

- TAHI Project (Japan)
- UNH InterOperability Lab (USA)

Even though this test suite was verified and tested, there may still be a few mistakes. All suggestions are welcome and can be send to:

- Frédéric Roudaut (frederic.roudaut@irisa.fr)
- César Viho (Cesar.Viho@irisa.fr)
- Annie Floch (annie.floch@irisa.fr)

Test with (*) and (!*) are only described in the extended version of the NAT-PT Conformance Test suite. Their run is only for informational purpose, if needed.

- ***(*): Concerns DNS-ALG not really recommanded for NAT-PT***
- ***(!*): Concerns really tricky tests for NAT-PT***

2 Test Case Description

Each test case of this SIIT/NAT-PT Conformance Test suite is described using the following sections:

Purpose:	The goal of this section is to describe briefly in a general way the main aim of the test case.
References:	This section is used to give references in the standards concerning the test purpose.
Resource Requirement:	It describes test equipments needed to perform the test case. It can be hardware or software need.
Test Requirement:	It is used to describe specific configurations for the node to test or for the tester in order to perform the test case. This section is in fact the initialization part of the test case.
Discussion:	This section gives more specificaly, the aim of the test according our current test architecture. It can also describe what are the awaited problems.
Packets:	All the different requiered packets for the test case will be described in this part. In the different packets, the following key-words will be deeply used: XXXXX : The value cannot be predicted. This key-word is only used in packets sent by the NUT. Indeed it is always predictable in packets sent by the tester. This value can also depend from other possible predictable fields and from unpredictable fields. This is the case of IPv4 checksums in packets sent by the NUT while we cannot predict TTL values. Nevertheless, UDP/TCP Checksum may be predicted if we know every field defined in the TCP/UDP pseudo-header and header **[RFC2460] [RFC 793]**. To calculate : The value can be calculated. It depends from other predictable fields. ( ) : The value is not strictely fixed. In packets sent by the tester it is juste an example of a possible value. Inn packets sent by the NUT, it means that this value is a consequence of a previous possible value chosen by the tester. All the others values are fixed.
Observable Results:	This section emphasizes the Observable Results to check by the tester to verify that the NUT is operating as specified in the standards.
Test Sequence:	The test sequence described packet exchanges at the IP Layer level between entities available in the test case. Packet exchange labeled by a number references the corresponding test intruction of the “Procedure” section. When the Packet Exchange is not labeled, it means that it is an observable result.
Postambule:	This section gives the current state of the Node Under Test after the test execution. This part is used to have the Node Under Test in its initial state after a run of the test case.
Possible Problem:	This section gives a description of possible problems that an execution of the test case could encounter. It discusses know issues which could lead to a “FAIL” verdict.

A few tests of this test suite need to have the capability to modify the different Link MTUs. Sometimes, the NUT cannot modify this value. Because NAT-PT is not required to handle PATH MTU Discovery, such tests MUST be skipped. These tests are described using the symbol **.

3 Terminology

Acronyms:

The following acronyms will be used in this document:

TR: Test Router.

TN: Test Node.

NUT: Node Under Test.

RUT: Router Under Test.

Terminology:

The terminology defined in **[RFC2765]** and **[RFC2766]** applies also to this document:

IPv4 capable node: A node which has an IPv4 protocol stack. In order for the stack to be usable the node must be assigned one or more IPv4 addresses.

IPv4 enabled node: A node which has an IPv4 protocol stack and is assigned one or more IPv4 addresses. Both IPv4-only and IPv6/IPv4 nodes are IPv4 enabled.

IPv6 capable node: A node which has an IPv6 protocol stack. In order for the stack to be usable the node must be assigned one or more IPv6 addresses.

IPv6 enabled node: A node which has an IPv6 protocol stack and is assigned one or more IPv6 addresses. Both IPv6-only and IPv6/IPv4 nodes are IPv6 enabled.

IPv4-mapped: An address of the form 0::ffff:a.b.c.d which refers to a node that is not IPv6-capable. In addition to its use in the API this protocol uses IPv4-mapped addresses in IPv6 packets to refer to an IPv4 node.

IPv4-compatible: An address of the form 0::0:a.b.c.d which refers to an IPv6/IPv4 node that supports automatic tunneling. Such addresses are not used in this protocol.

IPv4-translated: An address of the form 0::ffff:0:a.b.c.d which refers to an IPv6-enabled node. Note that the prefix 0::ffff:0:0:0/96 is chosen to checksum to zero to avoid any changes to the transport protocol's pseudo header checksum.

Application Level Gateway (ALG): Application Level Gateway (ALG) is an application specific agent that allows a V6 node to communicate with a V4 node and vice versa. Some applications carry network addresses in payloads. NAT-PT is application unaware and does not snoop the payload. ALG could work in conjunction with NAT-PT to provide support for many such applications.

PREFIX: a V6 node that needs to communicate with a V4 node needs to use a specific prefix PREFIX::/96) in front of the IPv4 address of the V4 node. Addresses using this PREFIX will be routed to the NAT-PT gateway (PREFIX::/96)

4 Topology and Test Configuration

Test Architecture:

The NUT node and the tester are connected with 2 links in which no other communications are allowed. The tester uses a packet generator to send IPv6 packets on the links, and a packet analyzer to see the packets sent by the NUT. The logical environment represented by this architecture is shown on Figure 1.

Each Link is an R45 crossed cable. One of these links represents the IPv4 Link whereas the other is for the IPv6 Link.

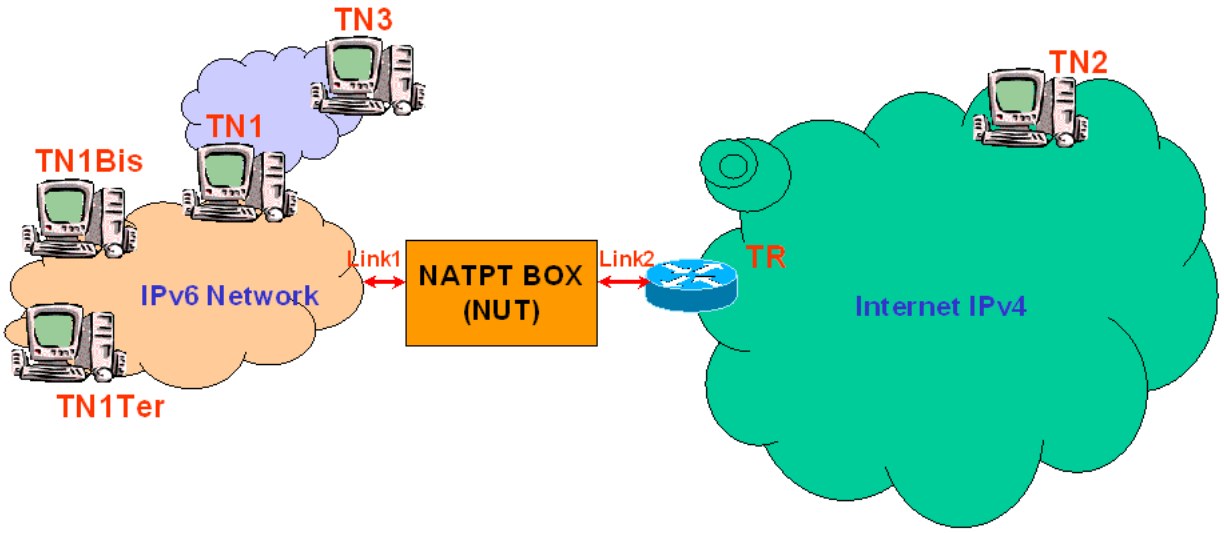


Figure 1: Logical Test Environment for NAT-PT Testing

Addressing:

The following defined values are just an exemple of needed values for this test suite:

- **TN1:**
  - **MAC address:** 00:00:00:00:a1:a1
  - **IPv4 Address:** 131.254.199.90 (Given by the NUT)
  - **IPv6 Global address**: 3ffe:501:ffff:100:200:ff:fe00:a1a1
  - **IPv6 Link local address:** fe80::200:ff:fe00:a1a1
- **TN1Bis:**
  - **MAC address:** 00:00:00:00:a3:a3
  - **IPv4 Address:** 131.254.199.91 (Given by the NUT)
  - **IPv6 Global address**: 3ffe:501:ffff:100:200:ff:fe00:a3a3
  - **IPv6 Link local address:** fe80::200:ff:fe00:a3a3
- **TN1Ter:**
  - **MAC address:** 00:00:00:00:a4:a4
  - **IPv4 Address:** 131.254.199.91 (Given By THE NUT)
  - **IPv6 Global address**: 3ffe:501:ffff:100:200:ff:fe00:a4a4
  - **IPv6 Link local address:** fe80::200:ff:fe00:a4a4
- **TN2:**
  - **IPv4 address:** 131.254.201.1
  - **IPv6 Global Address:** 3ffe:501:41c:c1ad::83fe:c901

(NAT-PT Address Corresponding To IPv4 Address)

- **TR:**

  MAC address: 00:00:00:00:a0:a0

  IPv4 address: 131.254.200.2

  IPv6 Global Address: 3ffe:501:41c:c1ad::83fe:c802

  (NAT-PT Address Corresponding To IPv4 Address)

- **TN3:**

  IPv4 Address: 131.254.199.92 (Given by the NUT)

  IPv6 Global address: 3ffe:501:ffff:200:200:ff:fe00:a0a0

- **NAT-PT Box:**

  NAT-PT Prefix: 3ffe:501:41c:c1ad::/64

  Link1 :

  Link1 Prefix: 3ffe:501:ffff:100::/64

  Global IPv6 address Link1: 3ffe:501:ffff:100::EUI-64 MAC Link1

  Link-Local Address Link1: fe80::EUI-64 MAC Link1

  Link2 :

  IPv4 address Link2: 131.254.200.1

Moreover the NUT MUST always assign the previous defined IPv4 address to TN1, TN1Bis and TN1Ter.

The NAT-PT rules will be the following:

Host	IPv6	IPv4
TN1	3ffe:501:ffff:100:200:ff:fe00:a1a1	131.254.199.90
TN1Bis	3ffe:501:ffff:100:200:ff:fe00:a3a3	131.254.199.91
TN1Ter	3ffe:501:ffff:100:200:ff:fe00:a4a4	131.254.199.91
TN3	3ffe:501:ffff:200:200:ff:fe00:a0a0	131.254.199.92

TN1Bis and TN1Ter will use the same IPv4 address for outbound packets.

To complete, this test topology, we need a default IPv4 route for the NUT. It means that on the NUT we should have:

- **default IPv4 Route:** 131.254.200.2 (TR)

Moreover all traffic to 3ffe:501:ffff:200::/64 will be forwarded to TN1. It involves that all packets to TN3 will be forwarded to TN1.

On NUT the IPv4 MTU will be set to 1500 when not explicitly precised. We will consider that medias of the IPv4 clouds are Ethernet Links. As defined in **[RFC2464]**, The default MTU size for IPv6 packets on an Ethernet is 1500 octets.

Moreover for best tests analyze, it could be required to desactivate Router Advertisement from the NUT.

Execution of test cases:

Each test case described here should begin when the NUT is at its initial state, e.g. with empty binding caches. We tried to limit the test link effect so that the test cases can be executed in sequence. However, some tests will have more reliable verdicts in a lone run. As no procedure ensures perfect clean-up when a NUT is found to be non conformant, a reinitialization of the NUT should follow every test case that leads to a verdict « FAIL ».

Link-local addresses resolution and ARP:

As part of the Neighbor Discovery protocol **[RFC2461]**, the NUT should send Neighbor Solicitation packets to discover or probe the link-local address of a node at any time. In most cases, the tester node should respond by a Neighbor Advertisement packet.

Moreover the Node should too send ARP Packet to discover MAC address From an IPv4 Address. In most cases, the tester node should also respond by an ARP Reply packet.

With, our test topology, it means that the NUT will send Neighbor Solicitation Packet to probe the link-local address of TN1 or to discover its MAC address and ARP Request Message to obtain the MAC address of TR.

These default packets are described hereafter. The values set in these packets should be taken as long as no explicit values are given.

- ARP Request (From NUT to get MAC Address of TR) (length: 28 bytes)

ARP Header (length: 28)
Hardware: 1 Protocol: 2048 HLEN: 6 PLEN: 4 Operation: 1 SenderHAddr: **NUT MAC Address Link2** SenderPAdd: **NUT IPv4 address Link2** TargetHAddr: XXXXX TargetPAddr: **TR IPv4 address Link2**

- ARP Reply (From TR, to give its MAC Address to the NUT) (length: 28 bytes)

ARP Header (length: 28)
Hardware: 1 Protocol: 2048 HLEN: 6 PLEN: 4 Operation: 2 SenderHAddr: **TR MAC Address Link2** SenderPAdd: **TR IPv4 address Link2** TargetHAddr: **NUT MAC Address Link2** TargetPAddr: **NUT IPv4 address Link2**

- Neighbor Solicitation (From NUT to get MAC Address of TN1, TN1Bis, TN1Ter) (length: 72 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: 0 FlowLabel: 0 PayloadLength: 32 NextHeader: 58 HopLimit : 255 SourceAddress: **NUT IPv6 link-local/Global address** DestinationAddress: **TN1/TN1Bis/TN1Ter IPv6 solicited node multicast address**
Neighbor Solicitation (length: 32)
Type: 135 Code: 0 Checksum: To Calculate Reserved: 0 TargetAddress: **TN1/TN1Bis/TN1Ter Global IPv6 Address or TN1/TN1Bis/TN1Ter Link-local Address**

Option ICMPv6 Source Link Layer (length:8) Type: 1 Length: 1 LinkLayerAddress: **NUT MAC Address Link1**

- Neighbor Solicitation (From NUT to check reachability of TN1/TN1Bis/TN1Ter) (length: 72 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: 0 FlowLabel: 0 PayloadLength: 32 NextHeader: 58 HopLimit : 255 SourceAddress: **NUT IPv6 link-local/Global address** DestinationAddress: **TN1/TN1Bis/TN1Ter Global IPv6 Address or TN1/TN1Bis/TN1Ter Link-local Address**
Neighbor Solicitation (length: 32) Type: 135 Code: 0 Checksum: To Calculate Reserved: 0 TargetAddress: **TN1/TN1Bis/TN1Ter Global IPv6 Address or TN1/TN1Bis/TN1Ter Link-local Address** **Option ICMPv6 Source Link Layer (length:8)** Type: 1 Length: 1 LinkLayerAddress: **NUT MAC Address Link1**

- Neighbor Advertisement (From TN1/TN1Bis/TN1Ter, to give its MAC Address to the NUT) (length: 72 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: 0 FlowLabel: 0 PayloadLength: 32 NextHeader: 58 HopLimit : 255 SourceAddress: **TN1/TN1Bis/TN1Ter Global IPv6 Address or TN1/TN1Bis/TN1Ter Link-local Address** DestinationAddress: **NUT IPv6 link-local/Global address (Same as the source field in the corresponding Neighbor Solicitation)**
Neighbor Advertisement (length: 32) Type: 136 Code: 0 Checksum: To calculate Rflag: 0 Sflag: 1 Oflag: 1 Reserved: 0 TargetAddress: **TN1/TN1Bis/TN1Ter Global IPv6 address or TN1/TN1Bis/TN1Ter Link-local Address (Same as the Target Address field in the corresponding**

Neighbor Solicitation)

Option ICMPv6 Target Link Layer (length:8)

Type: 2
Length: 1
LinkLayerAddress: **TN1/TN1Bis/TN1Ter MAC Address**

5 NAT-PT Conformance Test Suite

5.1 Basic NAT-PT

Traditional-NAT-PT allows hosts within a V6 network to access hosts in the V4 network. In a traditional-NAT-PT, sessions are unidirectional, outbound from the V6 network.

5.1.1 Basic Translation

5.1.1.1 Translation from IPv4 Network without state in the NAT-PT Box

Purpose:

Check Basic Translation from v4 to v6 when state is not present in the NAT-PT Translator.

References:

- **[RFC2765]** ***section 3.3***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- No State MUST be present in the NUT before the run of the test.

Discussion:

NAT-PT uses a pool of V4 addresses for assignment to V6 nodes on a dynamic basis as sessions are initiated across V4-V6 boundaries. It means that if no state is present in the NUT, an IPv6 host located in the NAT-PT network does not have an IPv4 address allocated. Thus, all IPv4 packets going from the v4 Network MUST be silently discarded.

Packets:

- IPv4/UDP (length: 36 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
UDP (length: 16)
SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To Calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- IPv4/TCP (length: 48 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 48
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 6
HeaderChecksum: To calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

TCP (length: 28)

SourcePort: (1000)
DestinationPort: (23)
SequenceNumber: **S/N**
AcknowledgmentNumber: **A/N**
DataOffset: 5
Reserverd: (0)
URGFlag: (0)
ACKFlag: **0**
PSHFlag: (0)
RSTFlag: (0)
SYNFlag: **SYNFLAG**
FINFlag: **0**
Window: (0)
Checksum: To calculate
UrgentPointer: (0)

Payload (length: 8)
Data: (01234567 89abcdef)

- ICMPv4 Echo Request (length: 36 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 36
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 1
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

ICMPv4 Echo Request (length: 16)

Type: 8
Code: 0
Checksum: To Calculate
Identifier: (512)
SequenceNumber: (0)
Payload (length: 8)
Data: (01234567 89abcdef)

Procedure:

0. No State is present in the NUT
1. TR forwards an “IPv4/UDP” Packet from TN2 to TN1.
2. TR forwards an “IPv4/TCP” Packet from TN2 to TN1. (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0)
3. TR forwards an “ICMPv4 Echo Request” Packet from TN2 to TN1.

Observable Results:

- **Step 1:** Because no state is present in the NUT, the NUT MUST silently discard this packet.
- **Step 2:** Because no state is present in the NUT, the NUT MUST silently discard this packet.
- **Step 3:** Because no state is present in the NUT, the NUT MUST silently discard this packet.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN1		1	IPv4/UDP <----------	TN2
TN1		2	IPv4/TCP **(**SYNFlag **is set,** S/N **is set to 1 and** A/N **is set to 0)** <----------	TN2
TN1		3	ICMPv4 Echo Request <----------	TN2

5.1.1.2 Address Translation (!*)

Purpose:

Check basic address translation, IPv4 Address allocation and UDP basic translation from a v6 network to the v4 Internet

References:

- **[RFC2766]** ***Section 5.2***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

IPv6 packets that are translated have a destination address of the form PREFIX::IPv4/96. Thus the low-order 32 bits of the IPv6 destination address is copied to the IPv4 destination address. This test checks that the correct translation address is done with such address format. Moreover it checks that the correct allocation address is done for TN1 when source address is TN1 IPv6 Global Address. If destination field if an IPv6 address from another network, packets SHOULD be silently discarded

Packets:

- IPv6/UDP (length: 56 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 16
NextHeader: 17
HopLimit : (64)
SourceAddress: **TN1 IPv6 Global Address**
DestinationAddress:
- **TN2 IPv6 Global NAT-PT Address**
- **3ffe:501:ffff:102::1 (**global IPv6 Address)

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- IPv4/UDP (length: 36 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 36
Identifier: 0
Reserved: 0
DF: 1

MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the correponding IPv6/UDP Packet]** Protocol: 17 HeaderChecksum: To calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

1. TN1 sends an “IPv6/UDP” Packet to TN2 (src: TN1 IPv6 Global Address, dest: TN2 IPv6 Global NAT-PT address) .
2. TN1 sends an “IPv6/UDP” Packet to an IPv6 Address without NAT-PT Prefix (src: TN1 IPv6 Global Address, dest: IPv6 Address 3ffe:501:ffff:102::1). This Packet MUST be send to the NUT. I.e the destination MAC address is the NUT MAC address.
3. **(!*)** TN1 sends an “IPv6/UDP” Packet to TN2 (src: TN1 Link-local Address, dest: TN2 IPv6 Global NAT-PT address) .

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv4/UDP” and sends it to TN2
- **Step 2:** NAT-PT SHOULD silently drop this Packet
- **(!*) Step 3:** The NUT translates this packet in “IPv4/UDP” and sends it to TN2

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN1	IPv6/UDP ---------->	1		TN2
TN1		Step1	IPv4/UDP ---------->	TN2
TN1	IPv6/UDP to **3ffe:501:ffff:102::1** ---------->	2		
TN1	IPv6/UDP ---------->	3 **(!*)**		TN2
TN1		**(!*)**Step3	IPv4/UDP ---------->	TN2

5.1.1.3 TrafficClass fields from IPv6 Header

Purpose:

Check correct translation of the TrafficClass Field of the IPv6 Header

References:

- **[RFC2765]** ***Section 5.2***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- The implementation should have the possibility to ignore the IPv6 Traffic Class and always set the IPv4 “TOS” field to 0

Discussion:

The IPv4 TOS Field is by default copied from the IPv6 Traffic Class field, Nevertheless, all implementations SHOULD provide the ability to ignore the IPv6 traffic class and always set the IPv4 "TOS" to zero.”

Packets:

- IPv6/UDP (length: 56 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: **255** FlowLabel: (0) PayloadLength: 16 NextHeader: 17 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv4/UDP (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: **0 or 255** TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the correponding IPv6/UDP Packet]** Protocol: 17

HeaderChecksum: To calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

1. TN1 sends an “IPv6/UDP” Packet to TN2 with TrafficClass set to 255.
2. The Implementation activates the ability to ignore the IPv6 TrafficClass field and always set the IPv4 “TOS” to 0.
3. TN1 sends an “IPv6/UDP” Packet to TN2 with TrafficClass set to 255.
4. The Implementation desactivates the ability to ignore the IPv6 TrafficClass field.

Observable Results:

- **Step 1:** The NUT Translates this packet in “IPv4/UDP” with TOS also set to 255 and sends it to TN2
- **Step 3:** The NUT Translates this packet in “IPv4/UDP” with TOS set to 0 and sends it to TN2

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
	IPv6/UDP (Traffic Class = 255)			
TN1	----+----------------------->	1		TN2
			IPv4/UDP (TOS = 255)	
TN1		Step1	---------------------------->	TN2
	2: Activation of the ability to ignore the IPv6 TrafficClass field			
	IPv6/UDP (Traffic Class = 255)			
TN1	---------------------------->	3		TN2
			IPv4/UDP (TOS = 0)	
TN1		Step3	---------------------------->	TN2
	4: Desactivation of the ability to ignore the IPv6 TrafficClass field			

5.1.1.4 IPv6 Options

Purpose:

Check Translation of IPv6 Options.

References:

- **[RFC2765]** ***section 4.1***
- **[RFC2766]** ***Section 5.3***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

If any of an IPv6 hop-by-hop options header, destination options header, or routing header with the Segments Left field equal to zero are present in the IPv6 packet, they are ignored..

Moreover, If a routing header with a non-zero Segments Left field is present then the packet MUST NOT be translated, and an ICMPv6 "parameter problem/ erroneous header field encountered" (Type 4/Code 0) error message, with the Pointer field indicating the first byte of the Segments Left field, SHOULD be returned to the sender."

Packets:

- IPv6/UDP 1 (with Hop by Hop Extension) (length: 64 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 24 NextHeader: 0 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Hop by Hop Option Header (length: 8) NextHeader = 17 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv6/UDP 2 (with Destination Option Header) (length: 64 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 24 NextHeader: 60 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Destination Option Header (length: 8) NextHeader = 17 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv6/UDP 3 (with Routing Header Type 0 with Segments Left =0) (length: 64 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 24 NextHeader: 43 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Routing Header (length: 8) NextHeader = 17 HeaderExtLength = 0 RoutingType = 0 SegmentsLeft = 0 Type-specificData = 0
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv6/UDP 4 (with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header) (length: 88 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 48 NextHeader: 0 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Hop by Hop Option Header (length: 8) NextHeader = 60 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Destination Option Header (length: 8) NextHeader = 43 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Routing Header (length: 8) NextHeader = 60 HeaderExtLength = 0 RoutingType = 0 SegmentsLeft = 0 Type-specificData = 0
Destination Option Header (length: 8) NextHeader = 17 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv6/UDP 5 (with Routing Header Type 0 with Segments Left > 0) (length: 96 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 56
NextHeader: 43
HopLimit : (64)
SourceAddress: **TN1 IPv6 Global Address**
DestinationAddress: **NUT IPv6 Global Address Link1**

Routing Header (length: 40)

NextHeader = 17
HeaderExtLength = 4
RoutingType = 0
SegmentsLeft = 2
Reserved = 0
Address = **TR IPv6 Global NAT-PT Address**
Address = **TN2 IPv6 Global NAT-PT Address**

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- IPv4/UDP (length: 36 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 36
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (63) **[one less than in the correponding IPv6/UDP Packet]**
Protocol: 17
HeaderChecksum: To calculate
SourceAddress: **TN1 IPv4 Address given by The NUT**
DestinationAddress: **TN2 IPv4 Address**

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- ICMPv6 Parameter Problem (length: 144 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: 0 FlowLabel: 0 PayloadLength: 104 NextHeader: 58 HopLimit : XXXXX SourceAddress: **NUT IPv6 Global Address Link1** DestinationAddress: **TN1 IPv6 Global Address**
ICMPv6 Parameter Problem (length: 104) Type: 4 Code: 0 Checksum: To Calculate Pointer: 43 **Payload (length: 96)** Data: **the corresponding IPv6/UDP 5 Packet**

Procedure:

1. TN1 sends an “IPv6/UDP 1” (with Hop by Hop Extension) to TN2.
2. TN1 sends an “IPv6/UDP 2” (with Destination Option Header) to TN2.
3. TN1 sends an “IPv6/UDP 3” (with Routing Header Type 0 with Segments Left =0) to TN2.
4. TN1 sends an “IPv6/UDP 4” (with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header) to TN2.
5. TN1 sends an “IPv6/UDP 5” (with Routing Header Type 0 with Segments Left>0) to TN2.

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv4/UDP” and sends it to TN2. In this new packet the UDP checksum has to be adjusted.
- **Step 2:** The NUT translates this packet in “IPv4/UDP” and sends it to TN2. In this new packet the UDP checksum has to be adjusted.
- **Step 3:** The NUT translates this packet in “IPv4/UDP” and sends it to TN2. In this new packet the UDP checksum has to be adjusted.
- **Step 4:** The NUT translates this packet in “IPv4/UDP” and sends it to TN2. In this new packet the UDP checksum has to be adjusted.
- **Step 5:** NAT-PT MUST drop this Packet and SHOULD send an “ICMPv6 Parameter Problem” packet to TN1.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN1	IPv6/UDP **(with Hop by Hop Extension)** ---------->	1		TN2
TN1		Step1	IPv4/UDP ---------->	TN2
TN1	IPv6/UDP **(with Destination Option Header)** ---------->	2		TN2
TN1		Step2	IPv4/UDP ---------->	TN2
TN1	IPv6/UDP **((with Routing Header Type 0 with Segments Left =0)** ---------->	3		TN2
TN1		Step3	IPv4/UDP ---------->	TN2
TN1	IPv6/UDP **(with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header)** ---------->	4		TN2
TN1		Step4	IPv4/UDP ---------->	TN2
TN1	IPv6/UDP **(with Routing Header Type 0 with Segments Left>0)** ---------->	5		TN2
TN1	ICMPv6 Parameter Problem <----------	Step5		

5.1.2 Upper Layer Translation

5.1.2.1 TCP Translation

Purpose:

Check translation of TCP Packets

References:

- **[RFC2766]** ***section 5.2***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

The TCP checksum should be adjusted to account for the address changes, going from V6 to V4 addresses.

Packets:

- IPv6/TCP (length: 60 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 20 NextHeader: 6 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
TCP (length: 20) SourcePort: (1000) DestinationPort: (23) SequenceNumber: **S/N** AcknowledgmentNumber: **A/N** DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: **ACKFlag** PSHFlag: (0) RSTFlag: (0) SYNFlag: **SYNFlag** FINFlag: **FINFlag** Window: (0) Checksum: To calculate UrgentPointer: (0) **Payload (length: 0)**

- IPv4/TCP (length: 40 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5

TypeOfService: (0) TotalLength: 40 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the IPv6/TCP]** Protocol: 6 HeaderChecksum: To calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
TCP (length: 20) SourcePort: (1000) DestinationPort: (23) SequenceNumber: **S/N** AcknowledgmentNumber: **A/N** DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: **ACKFLAG** PSHFlag: (0) RSTFlag: (0) SYNFlag: **SYNFLAG** FINFlag: **FINFLAG** Window: (0) Checksum: To calculate UrgentPointer: (0) **Payload (length: 0)**

Procedure:

Open a TCP Connection

1. TN1 sends an "IPv6/TCP" packet to TN2. (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0)
2. TR forwards an "IPv4/TCP" packet from TN2 to TN1. (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
3. TN1 sends an "IPv6/TCP" packet to TN2. (**ACKFlag** is set, **S/N** is set to 2 and **A/N** is set to 11)

Close the TCP Connection

4. TN1 sends an "IPv6/TCP" packet to TN2. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 2 and **A/N** is set to 11)
5. TR forwards an "IPv4/TCP" packet from TN2 to TN1. (**ACKFlag** is set, **S/N** is set to 11 and **A/N** is set to 3)
6. TR forwards an "IPv4/TCP" packet from TN2 to TN1. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 11 and **A/N** is set to 3)
7. TN1 sends an "IPv6/TCP" packet to TN2. (**ACKFlag** is set, **S/N** is set to 3 and **A/N** is set to 12)

Observable Results:

- **Step 1:** The NUT translates this packet in "IPv4/TCP" and sends it to TN2. In this new packet the TCP checksum has to be adjusted. (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0)
- **Step 2:** The NUT translates this packet in "IPv6/TCP" and sends it to TN1. In this new packet the TCP checksum has to be adjusted. (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
- **Step 3:** The NUT translates this packet in "IPv4/TCP" and sends it to TN2. In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 2 and **A/N** is set to 11)
- **Step 4:** The NUT translates this packet in "IPv4/TCP" and sends it to TN2. In this new packet the TCP checksum has to be adjusted. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 2 and **A/N** is set to 11)
- **Step 5:** The NUT translates this packet in "IPv6/TCP" and sends it to TN1. In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 11 and **A/N** is set to 3)

- **Step 6:** The NUT translates this packet in “IPv6/TCP” and sends it to TN1. In this new packet the TCP checksum has to be adjusted. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 11 and **A/N** is set to 3)
- **Step 7:** The NUT translates this packet in “IPv4/TCP” and sends it to TN2. In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 3 and **A/N** is set to 12)

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
		Open The TCP Connection		
TN1	IPv6/TCP **(SYN) S/N:1,A/N:0** ---------->	1 Step1	IPv4/TCP ---------->	TN2 TN2
TN1 TN1	IPv6/TCP <----------	2 Step2	IPv4/TCP **(SYN,ACK) S/N:10,A/N:2** <----------	TN2
TN1	IPv6/TCP **(ACK) S/N:2,A/N:11** ---------->	3 Step3	IPv4/TCP ---------->	TN2 TN2
		Close The TCP Connection		
TN1	IPv6/TCP **(FIN,ACK) S/N:2,A/N:11** ---------->	4 Step4	IPv4/TCP ---------->	TN2 TN2
TN1 TN1	IPv6/TCP <----------	5 Step5	IPv4/TCP **(ACK) S/N:11,A/N:3** <----------	TN2
TN1 TN1	IPv6/TCP <----------	6 Step6	IPv4/TCP **(FIN,ACK) S/N:11,A/N:3** <----------	TN2
TN1	IPv6/TCP **(ACK) S/N:3,A/N:11** ---------->	7 Step7	IPv4/TCP ---------->	TN2 TN2

5.1.2.2 UDP Translation

Purpose:

Check Translation of UDP Packets

References:

- **[RFC2766]** ***Section 5.2***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

The UDP checksum should be adjusted to account for the address changes, going from V6 to V4 addresses.

Packets:

- IPv6/UDP (length: 56 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 16 NextHeader: 17 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv4/UDP (length: 36 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the correponding IPv6/UDP Packet]** Protocol: 17 HeaderChecksum: To calculate SourceAddress: **TN1 IPv4 Address given by The NUT**

DestinationAddress: **TN2 IPv4 Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

1. TN1 sends an “IPv6/UDP” packet to TN2.

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv4/UDP” and sends it to TN2. In this new packet the UDP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
	IPv6/UDP			
TN1	---------->	1		TN2
			IPv4/UDP	
TN1		Step1	---------->	TN2

5.1.3 ICMPv4 Translation

An ICMPv4 Error Packet has sometimes an IPv4 Packet in its payload. In this part we will consider that the translation of the inner IP header has to be done. Indeed, what is the need to translate an ICMPv4 error message if the inner part is not also translated?

The ICMP message format is specified by the value of the Type field:

0	Echo reply.
1	Reserved.
2	Reserved.
3	Destination unreachable.
4	Source quench.
5	Redirect.
6	Alternate Host Address.
8	Echo request.
9	Router advertisement.
10	Router solicitation.
11	Time exceeded.
12	Parameter problem.
13	Timestamp request.
14	Timestamp reply.
15	Information request.
16	Information reply.
17	Address mask request.
18	Address mask reply.
19	Reserved (for security).
20-29	Reserved (for robustness experiment).
30	Traceroute.
31	Conversion error.
32	Mobile Host Redirect.
33	IPv6 Where-Are-You.
34	IPv6 I-Am-Here.
35	Mobile Registration Request.
36	Mobile Registration Reply.
37	Domain Name request.
38	Domain Name reply.
39	SKIP Algorithm Discovery Protocol.
40	Photuris, Security failures.
41-255	Reserved.

5.1.3.1 ICMPv4 Error Messages with UDP packet in payload

Purpose:

Check translation of ICMPv4 Error Messages with a UDP packet in payload

References:

- **[RFC2765],** ***Section 3.3***
- ***[RFC 2765] Page 15 :***

"….

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers."

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

This test checks the correct translation of ICMPv4 Error Messages with a UDP packet included in payload. The ICMP checksum should be adjusted to account for the address change, going from V4 to V6 addresses. The inner IP header has also to be translated.

Packets:

- IPv6/UDP (length: 48 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 8 NextHeader: 17 HopLimit : (64) **[Same as in IPv4/UDP]** SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
UDP (length: 8)
SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To calculate

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 / ICMPv6 Error Type 3 (length: 96 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]** SourceAddress: **TR IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**

ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big / ICMPv6 Time Exceeded (length: 56) Type: **TYPE** Code: **CODE** Checksum: To calculate Unused: 0 **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 4 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]** SourceAddress: **TR IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
ICMPv6 Parameter Problem (length: 56) Type: **4** Code: **CODE** Checksum: To calculate Pointer: **POINTER** **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- IPv4/UDP(length: 28 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 28 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
UDP (length: 8) SourcePort: (1000) DestinationPort: (1000)

Length: 8 Checksum: To Calculate

- ICMPv4 Error Type 3 / ICMPv4 Error Type 11 (length: 56 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 56 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TR IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Destination Unreachable **Or** **ICMPv4 Time Exceeded** **(length: 36)** Type: **3 or 11** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 28)** Data: **Same as IPv4/UDP Packet**

- ICMPv4 Error Type 12 (length: 56 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 56 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TR IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Parameter Problem **(length: 36)** Type: **12** Code: **CODE** Checksum: To Calculate Pointer: **POINTER** Unused: 0

Payload (length: 28) Data: **Same as IPv4/UDP Packet**

Procedure:

1. TN1 sends an IPv6/UDP Packet to TN2
2. Send “ICMPv4 Error Message” from TR to TN1 incrementing type and code. The inner part of these known packets contains an IPv4/UDP layer. Then, go back to Step 1.

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv4/UDP” and sends it to TN2. In this new packet the UDP checksum has to be adjusted.
- **Step 2:** These Packets must be silently dropped except for cases indicated in the table presented hereafter:

ICMPv4 Packet	ICMPv6 Corresponding Packet
Type = 3, Code = 0,1,5,6,7,8,11 or 12	Type = 1, Code = 0 (no route to destination)
Type = 3, Code = 2 (Protocol unreachable error)	Type = 4, Code = 1 (unrecognized Next Header type encountered) and make the Pointer point to the IPv6 Next Header field
Type = 3, Code = 3 (port unreachable)	Type = 1, Code = 4 (port unreachable)
Type = 3, Code = 4 (fragmentation needed and DF set)	Type = 2, Code = 0 (Too Big message) and the The MTU field needs to be adjusted
Type = 3, Code = 9, 10 (communication with destination host administratively prohibited)	Type = 1, Code = 1(communication with destination host administratively prohibited)
Type = 11 (Time Exceeded)	Type = 3 (Time Exceeded) and the code field is unchanged
Type = 12 (Parameter Problem)	Type = 4 and the pointer need to be adjusted to point to the corresponding field in the translated include IP header.

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner part of the translated Packet the UDP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN1	IPv6/UDP ---------->	1		TN2
TN1		Step1	IPv4/UDP ---------->	TN2
TN1		2	ICMPv4 Error <----------	TR
TN1	ICMPv6 Error (Some) <----------	Step2		TN2

5.1.3.2 ICMPv4 Error Messages with TCP packet in payload

Purpose:

Check translation of ICMPv4 Error Messages with a TCP packet in payload

References:

- **[RFC2766], *Section 5.3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

This test checks the correct translation of ICMPv4 Error Messages with a TCP packet included in payload. The ICMP checksum should be adjusted to account for the address change, going from V4 to V6 addresses. The inner IP header has also to be translated.

Packets:

- IPv6/TCP (length: 68 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 28 NextHeader: 6 HopLimit : (64) **[Same as in IPv4/TCP]** SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
TCP (length: 28)
SourcePort: (1000) DestinationPort: (23) SequenceNumber: (0) AcknowledgmentNumber: (0) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (0) PSHFlag: (0) RSTFlag: (0) SYNFlag: 1 FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0)
Payload (length: 8) Data: (01234567 89abcdef)

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 (length: 116 bytes)

IPv6 Header (length: 40)
Version: 6

TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 76
NextHeader: 58
HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]**
SourceAddress: **TR IPv6 Global NAT-PT Address**
DestinationAddress: **TN1 IPv6 Global Address**

ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big (length: 76)

Type: **TYPE**
Code: **CODE**
Checksum: To calculate
Unused: 0

Payload (length: 68)
Data: **Same as IPv6/TCP Packet**

- ICMPv6 Error Type 4 (length: 116 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 76
NextHeader: 58
HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]**
SourceAddress: **TR IPv6 Global NAT-PT Address**
DestinationAddress: **TN1 IPv6 Global Address**

ICMPv6 Parameter Problem (length: 76)

Type: **4**
Code: **CODE**
Checksum: To calculate
Pointer: **POINTER**

Payload (length: 68)
Data: **Same as IPv6/TCP Packet**

- IPv4/TCP (length: 48 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: 0
TotalLength: 48
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 6
HeaderChecksum: To Calculate
SourceAddress: **TN1 IPv4 Address given by The NUT**

DestinationAddress: **TN2 IPv4 Address**

TCP (length: 28)

SourcePort: (1000)
DestinationPort: (23)
SequenceNumber: (0)
AcknowledgmentNumber: (0)
DataOffset: 5
Reserverd: (0)
URGFlag: (0)
ACKFlag: (0)
PSHFlag: (0)
RSTFlag: (0)
SYNFlag: 1
FINFlag: (0)
Window: (0)
Checksum: To calculate
UrgentPointer: (0)

Payload (length: 8)
Data: (01234567 89abcdef)

- ICMPv4 Error Type 3 (length: 76 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 76
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 1
HeaderChecksum: To Calculate
SourceAddress: **TR IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

ICMPv4 Destination Unreachable (length: 56)

Type: **3**
Code: **CODE**
Checksum: To Calculate
Unused: 0

Payload (length: 48)
Data: **Same as IPv4/TCP Packet**

- ICMPv4 Error Type 12 (length: 76 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 76
Identifier: 0
Reserved: 0

DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TR IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Parameter Problem **(length: 56)** Type: **12** Code: **CODE** Checksum: To Calculate Pointer: **POINTER** Unused: 0 **Payload (length: 48)** Data: **Same as IPv4/TCP Packet**

Procedure:

1. TN1 sends an IPv6/TCP Packet to TN2
2. Send “ICMPv4 Error Message” described in the following from TR to TN1. The inner part of these packets contains an IPv4/TCP layer. Then, go back to Step 1.

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv4/TCP” and sends it to TN2. In this new packet the TCP checksum has to be adjusted.
- **Step 2:** These Packets must be translated according to the table presented hereafter:

ICMPv4 Packet	ICMPv6 Corresponding Packet
Type = 3, Code = 0,1,5,6,7,8,11 or 12	Type = 1, Code = 0 (no route to destination)
Type = 3, Code = 2 (Protocol unreachable error)	Type = 4, Code = 1 (unrecognized Next Header type encountered) and make the Pointer point to the IPv6 Next Header field
Type = 3, Code = 3 (port unreachable)	Type = 1, Code = 4 (port unreachable)
Type = 3, Code = 4 (fragmentation needed and DF set)	Type = 2, Code = 0 (Too Big message) and the The MTU field needs to be adjusted
Type = 3, Code = 9, 10 (communication with destination host administratively prohibited)	Type = 1, Code = 1(communication with destination host administratively prohibited)
Type = 12 (Parameter Problem)	Type=4 and the pointer need to be adjusted to point to the corresponding field in the translated include IP header.

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner part of the translated Packet the UDP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN1	IPv6/TCP ---------->	1		TN2
TN1		Step1	IPv4/TCP ---------->	TN2
TN1		2	ICMPv4 Error <----------	TR
TN1	ICMPv6 Error (Some) <----------	Step2		TN2

5.1.3.3 ICMPv4 Error Messages with IPv4 options in payload and header

Purpose:

Check translation of ICMPv4 Error Messages with IPv4 options included in payload and header.

References:

- **[RFC2765],** ***Section 3.1, 3.3***
- **[RFC 2765],** ***Page 15***

  "….

  The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers."

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

If IPv4 options are present in the IPv4 packet, they are ignored i.e., there is no attempt to translate them. However, if an unexpired source route option is present then the packet MUST instead be discarded, and an ICMPv4 "destination unreachable/source route failed" (Type 3/Code 5) error message SHOULD be returned to the sender.

This test checks the correct translation of ICMPv4 Error Messages with IPv4 Options included in payload and header. These Packets owes the following options: No Operation, Strict Source Route expired, Timestamp, Record Route, End of Option List. The inner part of these packets contains an IPv4/UDP layer with the same options (No Operation, Strict Source Route expired, Timestamp, Record Route, End of Option List). The ICMP checksum should be adjusted to account for the address change, going from V4 to V6 addresses. The inner IP header has also to be translated. These Packets must be translated according to the table presented in the following. Moreover, IPv4 Options MUST be ignored in the Header translation and in the Payload translation.

Packets:

- IPv6/UDP (length: 48 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 8 NextHeader: 17 HopLimit : (64) **[Same as in IPv4/UDP]** SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
UDP (length: 8) SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To calculate

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6

TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 56
NextHeader: 58
HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]**
SourceAddress: **TR IPv6 Global NAT-PT Address**
DestinationAddress: **TN1 IPv6 Global Address**

ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big (length: 56)

Type: **TYPE**
Code: **CODE**
Checksum: To calculate
Unused: 0

Payload (length: 48)
Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 4 (length: 96 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 56
NextHeader: 58
HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]**
SourceAddress: **TR IPv6 Global NAT-PT Address**
DestinationAddress: **TN1 IPv6 Global Address**

ICMPv6 Parameter Problem (length: 56)

Type: **4**
Code: **CODE**
Checksum: To calculate
Pointer: **POINTER**

Payload (length: 48)
Data: **Same as IPv6/UDP Packet**

- IPv4/UDP(length: 60 bytes)

IPv4 Header (length: 52)

Version: 4
IHL: 13
TypeOfService: (0)
TotalLength: 60
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN1 IPv4 Address given by The NUT**
DestinationAddress: **TN2 IPv4 Address**

NoOperation Option (length:1)
Type: 1

Strict Source Route Option (length:7)
Type: 137
Length: 7
Pointer:8
RouteData: **TN2 IPv4 Address**

Timestamp Option (length:8)
Type: 68
Length: 8
Pointer:5
Overflow: 0
Flag: 0
Timestamp: 170

NoOperation Option (length:1)
Type: 1

Record Route Option (length:11)
Type: 7
Length: 11
Pointer:4
RouteData: **0.0.0.0**
RouteData: **0.0.0.0**

EndofOptionList (length:1)
Type: 0

Padding = 000000

UDP (length: 8)

SourcePort: (1000)
DestinationPort: (1000)
Length: 8
Checksum: To Calculate

- ICMPv4 Error Type 3 (length: 88 bytes)

IPv4 Header (length: 52)

Version: 4
IHL: 13
TypeOfService: (0)
TotalLength: 88
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 1
HeaderChecksum: To Calculate
SourceAddress: **TR IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

NoOperation Option (length:1)
Type: 1

Strict Source Route Option (length:7)
Type: 137
Length: 7
Pointer:8
RouteData: **TN2 IPv4 Address**

Timestamp Option (length:8)
Type: 68
Length: 8
Pointer:5
Overflow: 0
Flag: 0
Timestamp: 170

NoOperation Option (length:1)
Type: 1

Record Route Option (length:11)
Type: 7
Length: 11
Pointer:4
RouteData: **0.0.0.0**
RouteData: **0.0.0.0**

EndofOptionList (length:1)
Type: 0

Padding = 000000

ICMPv4 Destination Unreachable (length: 36)

Type: **3**
Code: **CODE**
Checksum: To Calculate
Unused: 0

Payload (length: 60)
Data: **Same as IPv4/UDP Packet**

- ICMPv4 Error Type 12 (length: 88 bytes)

IPv4 Header (length: 52)

Version: 4
IHL: 13
TypeOfService: (0)
TotalLength: 88
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 1
HeaderChecksum: To Calculate
SourceAddress: **TR IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

NoOperation Option (length:1)
Type: 1

Strict Source Route Option (length:7)
Type: 137
Length: 7
Pointer:8
RouteData: **TN2 IPv4 Address**

Timestamp Option (length:8)
Type: 68
Length: 8
Pointer:5
Overflow: 0

Flag: 0 Timestamp: 170 **NoOperation Option (length:1)** Type: 1 **Record Route Option (length:11)** Type: 7 Length: 11 Pointer:4 RouteData: **0.0.0.0** RouteData: **0.0.0.0** **EndofOptionList (length:1)** Type: 0 **Padding = 000000**
ICMPv4 Parameter Problem **(length: 36)** Type: **12** Code: **CODE** Checksum: To Calculate Pointer: **POINTER** Unused: 0 **Payload (length: 60)** Data: **Same as IPv4/UDP Packet**

Procedure:

1. TN1 sends an IPv6/UDP Packet to TN2
2. Send “ICMPv4 Error Message” described in the following from TR to TN1. The inner part of these packets contains an IPv4/TCP layer. Moreover, Options (No Operation, Strict Source Route expired, Timestamp, Record Route, End of Option List) are present in the IPv4 Header and in the ICMPv4 Header of the ICMPv4 Error Packet. Then, go back to Step 1.

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv4/UDP” and sends it to TN2. In this new packet the UDP checksum has to be adjusted.
- **Step 2:** These Packets must be translated according to the table presented hereafter:

ICMPv4 Packet	ICMPv6 Corresponding Packet
Type = 3, Code = 0,1,5,6,7,8,11 or 12	Type = 1, Code = 0 (no route to destination)
Type = 3, Code = 2 (Protocol unreachable error)	Type = 4, Code = 1 (unrecognized Next Header type encountered) and make the Pointer point to the IPv6 Next Header field
Type = 3, Code = 3 (port unreachable)	Type = 1, Code = 4 (port unreachable)
Type = 3, Code = 4 (fragmentation needed and DF set)	Type = 2, Code = 0 (Too Big message) and the The MTU field needs to be adjusted
Type = 3, Code = 9, 10 (communication with destination host administratively prohibited)	Type = 1, Code = 1(communication with destination host administratively prohibited)
Type = 12 (Parameter Problem)	Type=4 and the pointer need to be adjusted to point to the corresponding field in the translated include IP header.

When Packet are translated, the options present in the IPv4 Header and in the ICMPv4 Header of the ICMPv4 Error Packet are removed. Moreover, the translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner Part of the translated Packet the UDP checksum has to be adjusted and the IPv4 Options have been removed.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
	IPv6/UDP			
TN1	---------->	1		TN2
			IPv4/UDP	
TN1		Step1	---------->	TN2
			ICMPv4 Error	
TN1		2	<----------	TR
	ICMPv6 Error (Some)			
TN1	<----------	Step2		TN2

5.1.4 ICMPv6 Translation

An ICMPv6 Error Packet has sometimes an IPv6 Packet in its payload. In this part we will consider that the translation of the inner IP header has to be done. Indeed, what is the need to translate an ICMPv6 error message if the inner part is not also translated?

The ICMPv6 message format is specified by the value of the Type field:

1	Destination unreachable.
2	Packet too big.
3	Time exceeded.
4	Parameter problem.
128	Echo request.
129	Echo reply.
130	Group Membership Query.
131	Group Membership Report.
132	Group Membership Reduction.
133	Router Solicitation.
134	Router Advertisement.
135	Neighbor Solicitation.
136	Neighbor Advertisement.
137	Redirect.
138	Router Renumbering.
139	ICMP Node Information Query.
140	ICMP Node Information Response.
141	Inverse Neighbor Discovery Solicitation Message.
142	Inverse Neighbor Discovery Advertisement Message.
143	Home Agent Address Discovery Request Message.
144	Home Agent Address Discovery Reply Message.
145	Mobile Prefix Solicitation.
146	Mobile Prefix Advertisement.

5.1.4.1 ICMPv6 Informational Messages

Purpose:

Check Translation of ICMPv6 Informational Messages. ICMPv6 Informational messages are ICMPv6 messages with a Type field value between 128 and 255.

References:

- **[RFC 2765]** ***Section 4.2***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

This test checks the correct translation of ICMPv6 Informational Messages. The ICMP checksum should be adjusted to account for the address change, going from V6 to V4 addresses. All ICMPv6 Informational messages except "ICMPv6 Echo Request" and "ICMPv6 Echo Reply" MUST be silently dropped.

Packets:

- ICMPv6 Echo Request (length: 56 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 16 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Echo Request (length: 16) Type: 128 Code: **5** Checksum: To calculate Identifier: (512) SequenceNumber: (0) **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv6 Echo Reply (length: 56 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 16 NextHeader: 58 HopLimit : (63) **[one less than in the ICMPv4 Echo Reply]** SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**

ICMPv6 Echo Reply (length: 16)

Type: 129
Code: **5**
Checksum: To calculate
Identifier: (512) **[one less than in the ICMPv6 Echo Request]**
SequenceNumber: (0) **[one less than in the ICMPv6 Echo Request]**

Payload (length: 8)
Data: (01234567 89abcdef)

- ICMPv6 MLD Query (length: 64 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 24
NextHeader: 58
HopLimit : (64)
SourceAddress: **TN1 IPv6 Global Address**
DestinationAddress: **TN2 IPv6 Global NAT-PT Address**

ICMPv6 MLD Query (length: 24)

Type: 130
Code: 0
Checksum: To Calculate
MaxResponseDelay: 30
Reserved: 0
MulticastAddress: **0 [General Query]**

- ICMPv6 MLD Done (length: 64 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 24
NextHeader: 58
HopLimit : (64)
SourceAddress: **TN1 IPv6 Global Address**
DestinationAddress: **TN2 IPv6 Global NAT-PT Address**

ICMPv6 MLD Done (length: 24)

Type: 132
Code: 0
Checksum: To Calculate
MaxResponseDelay: 0
Reserved: 0
MulticastAddress: **ff0E::1:FF00:0000**

- ICMPv6 MLD Report (length: 64 bytes)

IPv6 Header (length: 40)

Version: 6

TrafficClass: (0) FlowLabel: (0) PayloadLength: 24 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 MLD Report (length: 24) Type: 131 Code: 0 Checksum: To Calculate MaxResponseDelay: 0 Reserved: 0 MulticastAddress: **ff0E::1:FF00:0000**

- ICMPv6 Router Advertisement (length: 56 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 16 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Router Advertisement (length: 16) Type: 134 Code: 0 Checksum: To Calculate CurHopLimit: 0 Mflag: 0 Oflag: 0 Hflag: 0 Preference: 0 Reserved: 0 LifeTime: 1800 seconds ReachableTime: 0 RetransTimer: 0

- ICMPv6 Router Solicitation (length: 48 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 8 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Router Solicitation (length: 8) Type: 133 Code: 0

Checksum: To Calculate Reserved: 0

- ICMPv6 Neighbor Solicitation (length: 72 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 32 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Neighbor Solicitation (length: 32) Type: 135 Code: 0 Checksum: To Calculate Reserved: 0 TargetAddress: **TN2 IPv6 Global NAT-PT Address** **Option ICMPv6 Source Link Layer (length:8)** Type: 1 Length: 1 LinkLayerAddress: **TN1 MAC Address**

- ICMPv6 Neighbor Advertisement (length: 72 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 32 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Neighbor Advertisement (length: 32) Type: 136 Code: 0 Checksum: To Calculate Rflag: 0 Sflag: 0 Oflag: 0 Reserved: 0 TargetAddress: **TN1 IPv6 Global Address** **Option ICMPv6 Target Link Layer (length:8)** Type: 2 Length: 1 LinkLayerAddress: **TN1 MAC Address**

- ICMPv6 Redirect (length: 80 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 40 NextHeader: 58 HopLimit : 255 SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Redirect (length: 40) Type: 137 Code: 0 Checksum: To Calculate Reserved: 0 TargetAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN1Bis IPv6 Global Address**

- ICMPv6 Unknown Informational Message (length: 44 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 4 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 ANY (length: 4) Type: **TYPE** Code: **CODE** Checksum: To Calculate Data: **NONE**

- ICMPv4 Echo Reply (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Echo Reply (length: 16)

Type: 0
Code: **5**
Checksum: To Calculate
Identifier: (512) **[Same as in the ICMPv6 Echo Request]**
SequenceNumber: (0) **[Same as in the ICMPv6 Echo Request]**

Payload (length: 8)
Data: (01234567 89abcdef)

- ICMPv4 Echo Request (length: 36 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 36
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (63) **[one less than in the ICMPv6 Echo Request]**
Protocol: 1
HeaderChecksum: To Calculate
SourceAddress: **TN1 IPv4 Address given by The NUT**
DestinationAddress: **TN2 IPv4 Address**

ICMPv4 Echo Request (length: 16)

Type: 8
Code: **5**
Checksum: To Calculate
Identifier: (512) **[Same as in the ICMPv6 Echo Request]**
SequenceNumber: (0) **[Same as in the ICMPv6 Echo Request]**

Payload (length: 8)
Data: (01234567 89abcdef)

Procedure:

1. TN1 sends an “ICMPv6 Echo Request” to TN2.
2. TR Forwards an “ICMPv4 Echo Reply” from TN2 to TN1.
3. TN1 sends the following packets to TN2:
   - “ICMPv6 MLD Query Message”
   - “ICMPv6 MLD Done Message”
   - “ICMPv6 MLD Report Message”
   - “ICMPv6 Router Advertisement Message”
   - “ICMPv6 Router Solicitation Message”
   - “ICMPv6 Neighbor Advertisement Message”
   - “ICMPv6 redirect Message”
   - Some “ICMPv6 Unknown Informational Message” with different values “**TYPE**” and “**CODE**” for the type and code field (type = 255,code = 0; type = 0, code = 0 …)

Observable Results:

- **Step 1:** The NUT translates this packet in “ICMPv4 Echo Request” and sends it to TN2. In this new packet the ICMP checksum has to be adjusted.

- **Step 2:** The NUT translates this packet in “ICMPv6 Echo Reply” and sends it to TN1. In this new packet the ICMP checksum has to be adjusted.
- **Step 3:** These Packets must be silently dropped except for the cases indicated in the table presented hereafter:

ICMPv6 Packet	ICMPv4 Corresponding Packet
Type = 128 (Echo Request)	Type = 0 (Echo Request)
Type = 129 (Echo Reply)	Type = 8 (Echo Reply)

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN1	ICMPv6 Echo request ---------->	1		TN2
		Step1	ICMPv4 Echo Request ---------->	TN2
TN1		2	ICMPv4 Echo Reply <----------	TN2
TN1	ICMPv6 Echo Reply <----------	Step2		
TN1	• ICMPv6 MLD Query • ICMPv6 MLD Done • ICMPv6 MLD Report • ICMPv6 Router Advertisement • ICMPv6 Router Solicitation • ICMPv6 Neighbor Advertisement • ICMPv6 redirect • Some “ICMPv6 Unknown Informational Message” with different values “**TYPE**” and “**CODE**” for the type and code field (type = 255,code = 0; type = 0, code = 0 …) ---------->	3		TN2

5.1.4.2 ICMPv6 Error Messages with UDP packet in payload

Purpose:

Check Translation of ICMPv6 Error Messages when a UDP packet is included in payload. ICMPv6 error messages are ICMPv6 messages with a type field between 0 and 254.

References:

- **[RFC2765]** ***Section 4.2, 4.3***
- ***[RFC 2765] Page 15 :***

  "….

  The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers."

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

This test checks the correct translation of ICMPv6 Error Messages with a UDP packet included in payload. The ICMP checksum should be adjusted to account for the address change, going from V6 to V4 addresses. The inner IP header has also to be translated.

Packets:

- IPv6/UDP (length: 48 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 8 NextHeader: 17 HopLimit : (64) SourceAddress: **TN2 IPv6 Global NAT-PT Address or TN3 IPv6 Global Address** DestinationAddress: **TN3 IPv6 Global Address or TN2 IPv6 Global NAT-PT Address**
UDP (length: 8) SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To Calculate

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (64)

SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big (length: 56) Type: **TYPE** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 3 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Time Exceeded (length: 56) Type: **3** Code: **CODE** Checksum: To Calculate MTU: (1280) Unused: 0 **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 4 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Parameter Problem (length: 56) Type: **4** Code: **CODE** Checksum: To Calculate Pointer: **POINTER** **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Unknown Error Message (length: 44 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 4 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 ANY (length: 4) Type: **TYPE** Code: **CODE** Checksum: To Calculate Data: **NONE**

- IPv4/UDP(length: 28 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 28 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) **[Same as in IPv6/UDP]** Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address or TN3 IPv4 Address given by The NUT** DestinationAddress: **TN3 IPv4 Address given by The NUT or TN2 IPv4 Address**
UDP (length: 8) SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To Calculate

- ICMPv4 Error Type 3 / ICMPv4 Error Type 11 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0

FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Destination Unreachable / ICMPv4 Time Exceeded (length: 16) Type: **TYPE** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 8)** Data: **First 8 bytes of IPv4/UDP Packet**

- ICMPv4 Error Type 12 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Parameter Problem (length: 16) Type: 12 Code: **CODE** Checksum: To Calculate Pointer: **POINTER** **Payload (length: 8)** Data: **First 8 bytes of IPv4/UDP Packet**

Procedure:

1. TN1 sends an “IPv6/UDP” Packet from TN3 to TN2. The source address is TN3 IPv6 Global Address and the destination address is TN2 IPv6 Global NAT-PT Address.
2. TN2 replies with an “IPv4/UDP” Packet to TN3. The source address is TN2 IPv4 Address and the destination address is TN3 IPv4 Address given by The NUT.
3. Send “ICMPv6 Error Message” from TN1 to TN2 incrementing type and code. The inner part of these packets contains an IPv6/UDP layer. The source address of the inner “IPv6/UDP” packet is TN2 IPv6 Global NAT-PT Address and the destination address is TN3 IPv6 Global Address. Then, go back to Step 1.

Observable Results:

- **Step 1:** The NUT must translate this packet in “IPv4/UDP” and forward it to TN2. The source address is TN3 IPv4 Address given by The NUT and the destination address is TN2 IPv4 Address.
- **Step 2:** The NUT must translate this packet in “IPv6/UDP” and forward it to TN1. The source address is TN2 IPv6 Global NAT-PT Address and the destination address is TN3 IPv6 Global Address.

- **Step 3:** These Packets must be silently dropped except for the cases indicated in the table presented hereafter. The source address of the inner "IPv4/UDP" packet is TN2 IPv4 Address and the destination address is TN3 IPv4 Address given by The NUT.

ICMPv6 Packet	ICMPv4 Corresponding Packet
Type = 1, Code = 0 (no route to destination)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 1 (communication with destination administratively prohibited)	Type = 3, Code = 10 (communication with destination host administratively prohibited)
Type = 1, Code = 2 (beyond scope of source address)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 3 (address unreachable)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 4 (port unreachable)	Type = 3, Code = 3 (port unreachable)
Type = 2 (Packet Too Big)	Type = 3, Code = 4 And ajust the MTU Field
Type = 3 (Time Exceeded)	Type = 11, Code = unchanged
Type = 4, Code = 1	Type = 3, Code = 2 (protocol unreachable)
Type = 4, Code <> 1	Type = 12, Code = 0 and Update the Pointer (if Pointer was 0 -> 0 if Pointer was 4 -> 2 if Pointer was 6 -> 9 if Pointer was 7 -> 8 if Pointer was 8 -> 12 if Pointer was 24 -> 16)

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner part of the translated Packet the UDP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN3(Send from TN1)	IPv6/UDP ---------->	1 Step1	IPv4/UDP ---------->	TN2 TN2
TN3 TN3 (Send to TN1)	IPv6/UDP <----------	2 Step2	IPv4/UDP <----------	TN2
TN1	ICMPv6 Error ---------->	3 Step3	ICMPv4 Error (Some) ---------->	TN2 TN2

5.1.4.3 ICMPv6 Error Messages with TCP packet in payload

Purpose:

Check Translation of ICMPv6 Error Messages when a TCP packet is included in payload. ICMPv6 error messages are ICMPv6 messages with a type field between 0 and 254.

References:

- **[RFC2766] Section 5.3**
- **[RFC2765]** ***Page 21***

  " …

  ICMPv6 error messages:

  Destination Unreachable (Type 1)

  Set the Type field to 3. Translate the code field as follows:

  Code 0 (no route to destination): Set Code to 1 (host unreachable)."

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

This test checks the correct translation of ICMPv6 Error Messages with a TCP packet included in payload. The ICMP checksum should be adjusted to account for the address change, going from V6 to V4 addresses. The inner IP header has also to be translated.

Packets:

- IPv6/TCP (length: 68 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 28 NextHeader: 6 HopLimit : (64) SourceAddress: **TN2 IPv6 Global NAT-PT Address or TN3 IPv6 Global Address** DestinationAddress: **TN3 IPv6 Global Address or TN2 IPv6 Global NAT-PT Address**
TCP (length: 28)
SourcePort: (1000) DestinationPort: (23) SequenceNumber: **S/N** AcknowledgmentNumber: **A/N** DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: **ACKFlag** PSHFlag: (0) RSTFlag: (0) SYNFlag: **SYNFlag** FINFlag: **FINFlag** Window: (0) Checksum: To calculate

UrgentPointer: (0) **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 (length: 116 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 76 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big (length: 76) Type: **TYPE** Code: **CODE** Checksum: To calculate Unused: 0 **Payload (length: 68)** Data: **Same as IPv6/TCP Packet**

- ICMPv6 Error Type 3 (length: 116 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 76 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Time Exceeded (length: 76) Type: **3** Code: **CODE** Checksum: To calculate MTU: (1280) Unused: 0 **Payload (length: 68)** Data: **Same as IPv6/TCP Packet**

- ICMPv6 Error Type 4 (length: 116 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 76

NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Parameter Problem (length: 76) Type: **3** Code: **CODE** Checksum: To calculate Pointer: **POINTER** **Payload (length: 48)** Data: **Same as IPv6/TCP Packet**

- IPv4/TCP (length: 48 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 48 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) **[same as in IPv6/TCP]** Protocol: 6 HeaderChecksum: To calculate SourceAddress: **TN2 IPv4 Address or TN3 IPv4 Address given by The NUT** DestinationAddress: **TN3 IPv4 Address given by The NUT or TN2 IPv4 Address**
TCP (length: 28) SourcePort: (1000) DestinationPort: (23) SequenceNumber: **S/N** AcknowledgmentNumber: **A/N** DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: **ACKFlag** PSHFlag: (0) RSTFlag: (0) SYNFlag: **SYNFlag** FINFlag: **FINFlag** Window: (0) Checksum: To calculate UrgentPointer: (0) **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv4 Error Type 3 / ICMPv4 Error Type 11 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4

IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Destination Unreachable / ICMPv4 Time Exceeded (length: 16) Type: **TYPE** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 8)** Data: **First 8 bytes of IPv4/TCP Packet**

- ICMPv4 Error Type 12 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Parameter Problem (length: 16) Type: 12 Code: **CODE** Checksum: To Calculate Pointer: **POINTER** **Payload (length: 8)** Data: **First 8 bytes of IPv4/TCP Packet**

Procedure:

1. TN1 sends an “IPv6/TCP” Packet from TN3 to TN2. The source address is TN3 IPv6 Global Address and the destination address is TN2 IPv6 Global NAT-PT Address. (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0)
2. TN2 replies with an “IPv4/TCP” Packet to TN3. The source address is TN2 IPv4 Address and the destination address is TN3 IPv4 Address given by The NUT. (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
3. Send “ICMPv6 Error Message” from TN1 to TN2 incrementing type and code. The inner part of these packets contains an IPv6/TCP layer. The source address of the inner “IPv6/TCP” packet is TN2 IPv6 Global NAT-PT Address and the destination address is TN3 IPv6 Global Address. Then, go back to Step 1.

Observable Results:

- **Step 1:** The NUT must translate this packet in "IPv4/TCP" and forward it to TN2. The source address is TN3 IPv4 Address given by The NUT and the destination address is TN2 IPv4 Address.
- **Step 2:** The NUT must translate this packet in "IPv6/TCP" and forward it to TN1. The source address is TN2 IPv6 Global NAT-PT Address and the destination address is TN3 IPv6 Global Address.
- **Step 3:** These Packets must be translated according to the table presented hereafter. The source address of the inner "IPv4/TCP" packet is TN2 IPv4 Address and the destination address is TN3 IPv4 Address given by The NUT.

ICMPv6 Packet	ICMPv4 Corresponding Packet
Type = 1, Code = 0 (no route to destination)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 1 (communication with destination administratively prohibited)	Type = 3, Code = 10 (communication with destination host administratively prohibited)
Type = 1, Code = 2 (beyond scope of source address)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 3 (address unreachable)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 4 (port unreachable)	Type = 3, Code = 3 (port unreachable)
Type = 2 (Packet Too Big)	Type = 3, Code = 4 And ajust the MTU Field
Type = 3 (Time Exceeded)	Type = 11, Code = unchanged
Type = 4, Code = 1	Type = 3, Code = 2 (protocol unreachable)
Type = 4, Code <> 1	Type = 12, Code = 0 and Update the Pointer (if Pointer was 0 -> 0 if Pointer was 4 -> 2 if Pointer was 6 -> 9 if Pointer was 7 -> 8 if Pointer was 8 -> 12 if Pointer was 24 -> 16)

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner part of the translated Packet, the TCP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN3(Send from TN1)	IPv6/TCP ---------->	1 Step1	IPv4/TCP ---------->	TN2 TN2
TN3		2	IPv4/TCP <----------	TN2
TN3 (Send to TN1)	IPv6/TCP <----------	Step2		
TN1	ICMPv6 Error ---------->	3 Step3	ICMPv4 Error (Some) ---------->	TN2 TN2

5.1.4.4 ICMPv6 Error Messages with IPv6 options in payload and header

Purpose:

Check Translation of ICMPv6 Error Messages with IPv6 options included in payload and header. ICMPv6 error messages are ICMPv6 messages with a type field between 0 and 254.

References:

- **[RFC2765]** ***Section 4.2, 4.3***
- **[RFC 2765]** ***Page 15***

  "….

  The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers."
- **[RFC 2765]** ***Section 4.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

This test checks the correct translation of ICMPv6 Error Messages with IPv6 Options included in payload and header. These Packets owes the following extension headers: Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header. The inner part of these known packets contains an IPv6/UDP layer with the same options (Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header). The ICMP checksum should be adjusted to account for the address change, going from V6 to V4 addresses. The inner IP header has also to be translated. These Packets must be translated according to the table presented in the following. Moreover, IPv6 Options MUST be ignored in the Header translation and in the Payload translation.

Packets:

- IPv6/UDP (with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header) (length: 88 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 48 NextHeader: 0 HopLimit : (64) SourceAddress: **TN2 IPv6 Global NAT-PT Address or TN3 IPv6 Global Address** DestinationAddress: **TN3 IPv6 Global Address or TN2 IPv6 Global NAT-PT Address**
Hop by Hop Option Header (length: 8) NextHeader = 60 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Destination Option Header (length: 8)

NextHeader = 43 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Routing Header (length: 8) NextHeader = 60 HeaderExtLength = 0 RoutingType = 0 SegmentsLeft = 0 Type-specificData = 0
Destination Option Header (length: 8) NextHeader = 17 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 (with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header) (length: 168 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 128 NextHeader: 0 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Hop by Hop Option Header (length: 8) NextHeader = 60 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Destination Option Header (length: 8)

NextHeader = 43 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Routing Header (length: 8) NextHeader = 60 HeaderExtLength = 0 RoutingType = 0 SegmentsLeft = 0 Type-specificData = 0
Destination Option Header (length: 8) NextHeader = 58 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big (length: 96) Type: **TYPE** Code: **CODE** Checksum: To calculate Unused: 0 **Payload (length: 88)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 3 (with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header) (length: 168 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 128 NextHeader: 0 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Hop by Hop Option Header (length: 8) NextHeader = 60 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Destination Option Header (length: 8)

NextHeader = 43 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Routing Header (length: 8) NextHeader = 60 HeaderExtLength = 0 RoutingType = 0 SegmentsLeft = 0 Type-specificData = 0
Destination Option Header (length: 8) NextHeader = 58 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
ICMPv6 Time Exceeded (length: 96) Type: **3** Code: **CODE** Checksum: To calculate MTU: (1280) Unused: 0 **Payload (length: 88)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 4 (with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header) (length: 168 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 128 NextHeader: 0 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Hop by Hop Option Header (length: 8) NextHeader = 60 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Destination Option Header (length: 8)

NextHeader = 43 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Routing Header (length: 8) NextHeader = 60 HeaderExtLength = 0 RoutingType = 0 SegmentsLeft = 0 Type-specificData = 0
Destination Option Header (length: 8) NextHeader = 58 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
ICMPv6 Parameter Problem (length: 96) Type: **3** Code: **CODE** Checksum: To calculate Pointer: **POINTER** **Payload (length: 88)** Data: **Same as IPv6/UDP Packet**

- IPv4/UDP(length: 28 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 28 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) **[Same as in IPv6/UDP]** Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address or TN3 IPv4 Address given by The NUT** DestinationAddress: **TN3 IPv4 Address given by The NUT or TN2 IPv4 Address**
UDP (length: 8) SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To Calculate

- ICMPv4 Error Type 3 / ICMPv4 Error Type 11 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Destination Unreachable / ICMPv4 Time Exceeded (length: 16) Type: **TYPE** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 8)** Data: **First 8 bytes of IPv4/UDP Packet**

- ICMPv4 Error Type 12 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Parameter Problem (length: 16) Type: 12 Code: **CODE** Checksum: To Calculate Pointer: **POINTER** **Payload (length: 8)** Data: **First 8 bytes of IPv4/UDP Packet**

Procedure:

1. TN1 sends an “IPv6/UDP” Packet from TN3 to TN2. The source address is TN3 IPv6 Global Address and the destination address is TN2 IPv6 Global NAT-PT Address.

2. TN2 replies with an “IPv4/UDP” Packet to TN3. The source address is TN2 IPv4 Address and the destination address is TN3 IPv4 Address given by The NUT.
3. Send “ICMPv6 Error Message” described in the following from TN1 to TN2. These Packets owes the following extension headers: Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header. The inner part of these known packets contains an IPv6/UDP layer with the same options (Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header). The source address of the inner “IPv6/UDP” packet is TN2 IPv6 Global NAT-PT Address and the destination address is TN3 IPv6 Global Address. Then, go back to Step 1.

Observable Results:

- **Step 1:** The NUT must translate this packet in “IPv4/UDP” and forward it to TN2. The source address is TN3 IPv4 Address given by The NUT and the destination address is TN2 IPv4 Address.
- **Step 2:** The NUT must translate this packet in “IPv6/UDP” and forward it to TN1. The source address is TN2 IPv6 Global NAT-PT Address and the destination address is TN3 IPv6 Global Address.
- **Step 3:** These Packets must be translated according to the table presented hereafter. Moreover, IPv6 Options MUST be ignored in the Header translation and in the Payload translation. The source address of the inner “IPv4/UDP” packet is TN2 IPv4 Address and the destination address is TN3 IPv4 Address given by The NUT.

ICMPv6 Packet	ICMPv4 Corresponding Packet
Type = 1, Code = 0 (no route to destination)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 1 (communication with destination administratively prohibited)	Type = 3, Code = 10 (communication with destination host administratively prohibited)
Type = 1, Code = 2 (beyond scope of source address)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 3 (address unreachable)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 4 (port unreachable)	Type = 3, Code = 3 (port unreachable)
Type = 2 (Packet Too Big)	Type = 3, Code = 4 And ajust the MTU Field
Type = 3 (Time Exceeded)	Type = 11, Code = unchanged
Type = 4, Code = 1	Type = 3, Code = 2 (protocol unreachable)
Type = 4, Code <> 1	Type = 12, Code = 0 and Update the Pointer (if Pointer was 0 -> 0 if Pointer was 4 -> 2 if Pointer was 6 -> 9 if Pointer was 7 -> 8 if Pointer was 8 -> 12 if Pointer was 24 -> 16)

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner part of the translated Packet, the TCP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN3(Send from TN1)	IPv6/UDP ---------->	1 Step1	IPv4/UDP ---------->	TN2 TN2
TN3 TN3 (Send to TN1)	IPv6/UDP <----------	2 Step2	IPv4/UDP <----------	TN2
TN1	ICMPv6 Error ---------->	3 Step3	ICMPv4 Error (Some) ---------->	TN2 TN2

5.1.5 ICMPv6 Error Generation

5.1.5.1 HOP Limit set to 0 or 1 & ICMPv6 Time Exceeded Message

Purpose:

Check that the NUT discards packets it receives with HOP Limit set to 0 or 1.

References:

- **[RFC2765]** ***Section 4.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

This test checks that packet is correctly discarded when HOP Limit is set to 0 or 1. Moreover, the NUT SHOULD send an ICMPv6 "Time Exceeded" error.

Packets:

- IPv6/UDP with Hop Limit = 0 or 1(length: 56 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 16 NextHeader: 17 HopLimit: **0 or 1** SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv6 Time Exceeded Code 0 (length: 104 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: 0 FlowLabel: 0 PayloadLength: 64 NextHeader: 58 HopLimit : XXXXX SourceAddress: **NUT IPv6 Global Address or NUT Link-local IPv6 Address Link1** DestinationAddress: **TN1 IPv6 Global Address**

ICMPv6 Time Exceeded (length: 64)

Type: 3
Code: 0
Checksum: To Calculate
Unused: 0

Payload (length: 56)
Data: the corresponding IPv6/UDP Packet

Procedure:

1. TN1 sends an “IPv6/UDP” with Hop Limit set to 0 to TN2
2. TN1 sends an “IPv6/UDP” with Hop Limit set to 1 to TN2

Observable Results:

- **Step 1:** NAT-PT MUST drop this Packet and SHOULD send an “ICMPv6 Time Exceeded Code 0” Packet to TN1. The offending packet included in the “ICMPv6 Time Exceeded Code 0” should be the one which causes the error (ie the “ICMPv6 Echo Request” with Hop Limit set to 0).
- **Step 2:** NAT-PT MUST drop this Packet and SHOULD send an “ICMPv6 Time Exceeded Code 0” Packet to TN1. The offending packet included in the “ICMPv6 Time Exceeded Code 0” should be the one which causes the error (ie the “ICMPv6 Echo Request” with Hop Limit set to 1 or 0).

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
	IPv6/UDP With HOP LIMIT == 0			
TN1	---------->	1		TN2
	ICMPv6 Time Exceeded			
TN1	<----------	Step1		
	IPv6/UDP With HOP LIMIT == 1			
TN1	---------->	2		TN2
	ICMPv6 Time Exceeded			
TN1	<----------	Step2		

Possible Problem:

Possible problems can appear in this test case. The “ICMPv6 Time Exceeded Code 0” packet may not match the waited packet. The “ICMPv6 Time Exceeded Code 0” packet does not contain the full IPv6/UDP packet. According to **[RFC2463]**, when processing ICMPv6 messages, Implementations MUST observe the following rule: every ICMPv6 error message (type < 128) MUST includes as much of the IPv6 offending (invoking) packet (the packet that caused the error) as will fit without making the error message packet exceed the minimum IPv6 MTU. However, with the full IPv6/UDP packet, the length of ICMPv6 Time Exceeded Code 0 part is 104 bytes which is less than the minimum IPv6 MTU 1280 bytes.

5.1.6 MTU Handling & Fragmentation

5.1.6.1 Translation of Fragmented packets

Purpose:

Check that Fragmented IPv6/UDP Packets are correctly translated. In particular, check that the UDP checksum is well adjusted.

References:

- **[RFC2765]** ***Section 4.1, 4.2***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

Check the correct translation of IPv6 fragmented packets. In particular, check the following IPv4 field:

- Total Length: Payload length value from IPv6 header, minus 8 for the Fragment header, plus the size of the IPv4 header.
- Identification: Copied from the low-order 16-bits in the Identification field in the Fragment header.
- Flags: The More Fragments flag is copied from the M flag in the Fragment header. The Don't Fragments flag is set to zero allowing this packet to be fragmented by IPv4 routers.
- Fragment Offset: Copied from the Fragment Offset field in the Fragment Header.
- Protocol: Next Header value copied from Fragment header."

Packets:

- IPv6/UDP (1st Fragment) (length: 64 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 24 NextHeader: 44 HopLimit : (64) SourceAddress: **TN1 Global IPv6 Address** DestinationAddress: **TN2 Global IPv6 NAT-PT Address**
Fragment Header (length:8)
NextHeader: 17 Reserved1: 0 FragmentOffset: 0 Reserved2: 0 Mflag: 1 Identification: (0xFFEFFF01)
UDP (length: 16)
SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate

Payload (length: 8) Data: (01234567 89abcdef)

- IPv6/UDP (2nd Fragment) (length: 56 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 16 NextHeader: 44 HopLimit : (64) SourceAddress: **TN1 Global IPv6 Address** DestinationAddress: **TN2 Global IPv6 NAT-PT Address**
Fragment Header (length:8) NextHeader: 17 Reserved1: 0 FragmentOffset: 2 (in 8-octet units) Reserved2: 0 Mflag: 0 Identification: (0xFFEFFF01)
Payload (length: 8) Data: (01234567 89abcdef)

- IPv6/UDP (Unique Fragment) (length: 64 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 24 NextHeader: 44 HopLimit : (64) SourceAddress: **TN1 Global IPv6 Address** DestinationAddress: **TN2 Global IPv6 NAT-PT Address**
Fragment Header (length:8) NextHeader: 17 Reserved1: 0 FragmentOffset: 0 Reserved2: 0 Mflag: 0 Identification: (0xFFEFFF02)
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv4/UDP (1st Fragment) (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: (0xFF01) [low-order 16-bit of the Identification field in the Fragment Header of IPv6/UDP (1st Fragment)] Reserved: 0 DF: 0 MF: 1 FragmentOffset: 0 TTL: (63) **[one less than in the correponding IPv6/UDP (1st Fragment) Packet]** Protocol: 17 HeaderChecksum: To calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv4/UDP (2nd Fragment) (length: 28 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 28 Identifier: (0xFF01) [low-order 16-bit of the Identification field in the Fragment Header of IPv6/UDP (2nd Fragment)] Reserved: 0 DF: 0 MF: 0 FragmentOffset: 2 (in 8-octet units) TTL: (63) **[one less than in the correponding IPv6/UDP (2nd Fragment) Packet]** Protocol: 17 HeaderChecksum: To calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
Payload (length: 8) Data: (01234567 89abcdef)

- IPv4/UDP (Unique Fragment) (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0)

TotalLength: 36 Identifier: (0xFF02) [low-order 16-bit of the Identification field in the Fragment Header of IPv6/UDP (Unique Fragment)] Reserved: 0 DF: 0 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the correponding IPv6/UDP (Unique Fragment) Packet]** Protocol: 17 HeaderChecksum: To calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

1. TN1 sends an "IPv6/UDP (1st Fragment)" to TN2.
2. TN1 sends an "IPv6/UDP (2nd Fragment)" to TN2.
3. TN1 sends an "IPv6/UDP (Unique Fragment)" to TN2.

Observable Results:

- **Step 1:** The NUT translates this packet in "IPv4/UDP (1st Fragment)" and sends it to TN2. In this new packet the UDP checksum has to be adjusted.
- **Step 2:** The NUT translates this packet in "IPv4/UDP (2nd Fragment)" and sends it to TN2. In this new packet the UDP checksum has to be adjusted.
- **Step 3:** The NUT translates this packet in "IPv4/UDP (Unique Fragment)" and sends it to TN2. In this new packet the UDP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN1	IPv6/UDP (1st Fragment) ---------->	1		TN2
		Step1	IPv4/UDP (1st Fragment) ---------->	TN2
TN1	IPv6/UDP (2nd Fragment) ---------->	2		TN2
		Step2	IPv4/UDP (2nd Fragment) ---------->	TN2
TN1	IPv6/UDP (Unique Fragment) ---------->	3		TN2
		Step3	IPv4/UDP (Unique Fragment) ---------->	TN2

5.1.6.2 Fragmentation at IPv4 Level (**)

Purpose:

Check that IPv6 Packets of 1280 bytes are translated even when the IPv4 MTU on NUT is below 1280 bytes. In this case the resulting packet has to be fragmented. In this test, The IPv4 MTU on the NUT has to be set to 80 Bytes. Because NAT-PT is not required to handle PATH MTU Discovery, this test MUST be skipped if the NUT cannot modify this value.

References:

- **[RFC2765]** ***Section 3.5***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- The IPv4 MTU on NUT has to be set to 80 Bytes

Discussion:

The IPv4 MTU on NUT has to be set to 80. Because NAT-PT is not required to handle PATH MTU Discovery, this test MUST be skipped if the NUT cannot modify this value.

Because the Minimum IPv6 link MTU is 1280 bytes, It means that IPv6 Packets of 1280 bytes has to be fragmented at the IPv4 Level before to be forwarded by the NUT on the IPv4 Link.

Packets:

- IPv6/UDP (length: 1280 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 1240 NextHeader: 17 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
UDP (length: 1240) SourcePort: (1000) DestinationPort: (1000) Length: 1240 Checksum: To calculate **Payload (length: 1232)** Data: 1232 Bytes of data

- IPv4/UDP 1 Fragmented (1st Fragment) (length: 76 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 76 Identifier: XXXXX Reserved: 0 DF: 0 MF: 1

FragmentOffset: 0 TTL: (63) **[one less than in the IPv6/UDP Packet]** Protocol: 17 HeaderChecksum: To calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
UDP (length: 56) SourcePort: (1000) DestinationPort: (1000) Length: 1232 Checksum: To calculate **Payload (length: 48)** Data: **The 48 first bytes of the IPv6/UDP Packet**

- IPv4/UDP 2 Fragmented (length: 76 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 76 Identifier: XXXXX [Same as IPv4/UDP 1 Fragmented)] Reserved: 0 DF: 0 MF: 1 FragmentOffset: **FRAGMENT_OFFSET** (in 8-octet units) TTL: (63) Protocol: 17 HeaderChecksum: To calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
Payload (length: 56) Data: **The corresponding 56 bytes of the IPv6/UDP Packet**

- IPv4/UDP 3 Fragmented (Last Fragment) (length: 28 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 60 Identifier: XXXXX [Same as IPv4/UDP 1 Fragmented)] Reserved: 0 DF: 0 MF: 0 FragmentOffset: **154** (in 8-octet units) TTL: (63) **[one less than in the IPv6/UDP Packet]** Protocol: 17 HeaderChecksum: To calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
Payload (length: 8) Data: **The last 8 bytes of the IPv6/UDP Packet**

Procedure:

0. Set NUT's IPv4 MTU to 80
1. TN1 sends “IPv6/UDP” to TN2 (Packet size is 1280 bytes).

Observable Results:

- **Step 1:** The NUT fragmentes this packet in 23 IPv4/UDP and sends it to TN2. In each Fragment, the FragmentOffset value (**FRAGMENT_OFFSET)** is a multiple of 7. In the First Fragment it is 0 and it is 154 in the last Fragment.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4 MTU = 80] (Forwarded by TR)	Tester(IPv6)
	IPv6/UDP (1280 Bytes)			
TN1	---------->	1		TN2
			IPv4/UDP (23 Fragments)	
		Step1	---------->	TN2

5.1.7 FTP-ALG

The use of IP addresses and TCP Ports in FTP Packets with commands such as EPRT, EPSV, involve the need of having an FTP-ALG located on the NUT to facilitate transparent FTP between v6 and v4 nodes. This part give some conformance tests to check the correct translation of FTP packets from a v6 (TN1) to a v4 (TN2) node.

Each test will begin with the establishment of a passive TCP connection to the FTP server TN2 using port 21.

This is not really mandatory but because each packet going outside the v6 network has to go through the NUT, this can lead to some problems. Indeed the NUT can keep the status of live TCP connection and by this way, rejects all TCP packets when no state is available in the router. As a consequence, each test will have to end with the closing of this TCP connection.

5.1.7.1 EPRT

Purpose:

Check the correct translation of EPRT command

References:

- **[RFC2766]** ***Section 6.2***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- A FTP-ALG MUST be available on the NUT

Test Requirement:

NONE

Discussion:

According to **[RFC2766]**, it is recommended to have support for **[RFC2428]**. It involves that all v4 EPRT packets send from TN1 have to be translated into v6 EPRT packets. If the FTP application has not been upgraded to support EPRT and EPSV extensions to allow access to v4, the EPRT packet has to be translated into a PORT command.

Packets:

- IPv4/TCP/FTP EPRT (length: 71 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 71 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 6 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
TCP (length: 20)
SourcePort: (39426) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2)

DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (65535) Checksum: To calculate UrgentPointer: (0)
FTP – Command EPRT (length: 31) **Request: TN1_FTP_EPRT_v4**

- IPv4/TCP/FTP PORT (length: 67 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 67 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 6 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
TCP (length: 20) SourcePort: (39426) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (65535) Checksum: To calculate UrgentPointer: (0)
FTP – PORT Command (length: 27) **Request: TN1_FTP_PORT_v4**

- IPv6/TCP/FTP EPRT(length: 111 bytes)

IPv6 Header (length: 40) Version: 6

TrafficClass: (0) FlowLabel: 0 PayloadLength: 71 NextHeader: 6 HopLimit : (63) **[one less than in the IPv4/TCP/FTP EPRT]** SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
TCP (length: 20) SourcePort: (39426) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0)
FTP – command EPRT (length: 51) **Request: TN1_FTP_EPRT_v6**

TN1_FTP_EPRT_v6 : The EPRT command allows for the specification of an extended address for the data connection. The extended address MUST consist of the network protocol as well as the network and transport addresses. The syntax is “EPRT<space><d><net-prt><d><net-addr><d><tcp-port><d>” where:

- <d> is a delimiter character (the character "|" is recommended).
- <net-prt> is the protocol (value is 2 for IPv6)
- <net-addr> is **TN1 IPv6 Global NAT-PT Address**
- <tcp-port> is 39427

In our specific case we have “EPRT |2|3ffe:501:ffff:100:200:ff:fe00:a1a1|39427|”.

TN1_FTP_EPRT_v4 : The syntax is similar to the previous one except that:

- <net-prt> is the protocol IPv4 (value is 1 for IPv4)
- <net-addr> is **TN1 IPv4 Address**

In our specific case we have “EPRT |1|131.254.199.90|39427|”.

TN1_FTP_PORT_v4 : The syntax is “PORT<space>a1,a2,a3,a4,p1,p2”. It Specifies the host TN1 and port to which the server should connect for the next file transfer. This is interpreted as IP address a1.a2.a3.a4, port p1*256+p2. In our specific case we have “PORT 131,254,199,90,154,3”.

Procedure:

1. Establish a TCP connection from TN1 port 39426 to TN2 port 21 as defined in 5.1.2.1.
2. TN1 sends “IPv6/FTP/TCP EPRT” packet to TN2
3. Establish a TCP connection from TN2 port 20 to TN1 port 39427 as defined in 5.3.3.2
4. Close the TCP connection from TN2 port 20 to TN1 port 39427 as defined in 5.3.3.2
5. Close the TCP connection from TN1 port 39426 to TN2 port 21 as defined in 5.1.2.1

Observable Results:

- **Step 2:** The FTP-ALG on the NUT translates the packet in an “IPv4/TCP/FTP EPRT” packet before to forward it to TN2. If the FTP application has not been upgraded to support EPRT and EPSV extensions to allow access to v4, the NUT has to translate the packet in an “IPv4/TCP/FTP PORT” packet before to forward it to TN2.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
	1: Establishement of a TCP connection from TN1 port 39426 to TN2 port 21			
TN1	IPv6/FTP/TCP EPRT ---------->	2		TN2
TN1		Step2	IPv4/FTP/TCP EPRT ----------> **OR** IPv4/FTP/TCP PORT ---------->	TN2 TN2
	3: Establishement of a TCP connection from TN2 port 20 to TN1 port 39427			
	4: Closing of the TCP connection from TN2 port 20 to TN1 port 39427			
	5: Closing of the TCP connection from TN1 port 39426 to TN2 port 21			

5.1.7.2 EPSV

Purpose:

Check the correct translation of EPSV command

References:

- **[RFC2766]** ***Section 6.2***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- A FTP-ALG MUST be available on the NUT

Test Requirement:

NONE

Discussion:

According to **[RFC2766]**, it is recommended to have support for **[RFC2428]**. It involves that all v4 EPSV packets send from TN1 have to be translated into v6 EPSV packets. If the FTP application has not been upgraded to support EPSV and PASV extensions to allow access to v4, the EPSV packet has to be translated into a PASV command.

Packets:

- IPv4/TCP/FTP EPSV (length: 46 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 46 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the IPv6/TCP/FTP EPSV]** Protocol: 6 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
TCP (length: 20) SourcePort: (39526) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (65535) Checksum: To calculate UrgentPointer: (0)
FTP – Command EPSV (length: 6)

Request: "EPSV"

- IPv4/TCP/FTP PASV (length: 46 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 46 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the IPv6/TCP/FTP EPSV]** Protocol: 6 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
TCP (length: 20) SourcePort: (39526) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (65535) Checksum: To calculate UrgentPointer: (0)
FTP – Command PASV (length: 6) **Request: "PASV"**

- IPv4/TCP/FTP EPSV Response (length: 88 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 88 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 6 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**

TCP (length: 20) SourcePort: **21** DestinationPort: (39526) SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (65535) Checksum: To calculate UrgentPointer: (0)
FTP – Response EPSV (length: 48) **Response: TN2_FTP_EPSV_v4**

- IPv6/TCP/FTP EPSV Response (length: 108 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 68 NextHeader: 6 HopLimit : (63) **[one less than in the IPv6/TCP/FTP EPSV Response]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
TCP (length: 20) SourcePort: **21** DestinationPort: (39526) SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0)
FTP – Response EPSV (length: 48) **Response: TN2_FTP_EPSV_v6**

- IPv6/TCP/FTP EPSV(length: 66 bytes)

IPv6 Header (length: 40)

Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 26 NextHeader: 6 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
TCP (length: 20) SourcePort: (39526) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0)
FTP – Command EPSV (length: 6) **Request: "EPSV"**

TN2_FTP_EPSV_v6: The EPSV response has the following full syntax: "229 Entering Extended Passive Mode (<d><d><d><tcp-port><d>)" where:

- <d> is a delimiter character (the character "|" is recommended).
- <tcp-port> is the same than in the corresponding IPv4/TCP/FTP packet.

In our specific case we have "229 Entering Extended Passive Mode (|||55555|)".

TN2_FTP_PASV_v4: The full response syntax is "227 Entering Passive Mode (a1,a2,a3,a4,p1,p2)" where a1.a2.a3.a4 is the IP address and p1*256+p2 is the port number. It Specifies the host TN1 and port (for exemple 55555) to which the client should connect for the next file transfer. In our specific case we have "227 Entering Passive Mode (131,254,201,1,217,3)".

TN2_FTP_EPSV_v4: The syntax is similar to **TN2_FTP_EPSV_v6**.

Procedure:

1. Establish a TCP connection from TN1 port 39526 to TN2 port 21 as defined in 5.1.2.1.
2. TN1 sends "IPv6/FTP/TCP EPSV" packet to TN2
3. TR forwards an "IPv4/FTP/TCP EPSV Response" packet from TN2 to TN1 (giving TCP port 55555)
4. Establish a TCP connection from TN1 port 39527 to TN2 port 55555 as defined in 5.1.2.1.
5. Close the TCP connection from TN1 port 39527 to TN2 port 55555 as defined in 5.1.2.1.
6. Close the TCP connection from TN1 port 39526 to TN2 port 21 as defined in 5.1.2.1

Observable Results:

- **Step 2:** The FTP-ALG on the NUT translates the packet in an "IPv4/TCP/FTP EPSV" packet and forwards it to TN2. If the FTP application has not been upgraded to support EPRT and EPSV extensions to allow access to v4, the NUT has to translate the packet in an "IPv4/TCP/FTP PASV" packet before to forward it to TN2.

- **Step 3:** The FTP-ALG on the NUT translates the packet in an “IPv6/TCP/FTP EPSV Response” packet and forwards it to TN1.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
	1: Establishement of a TCP connection from TN1 port 39526 to TN2 port 21			
	IPv6/FTP/TCP EPSV			
TN1	---------->	2		TN2
			IPv4/FTP/TCP EPSV	
TN1		Step2	---------->	TN2
			OR	
			IPv4/FTP/TCP PASV	
			---------->	TN2
			IPv4/FTP/TCP EPSV Response	
TN1		3	<----------	TN2
	IPv6/FTP/TCP EPSV Response			
TN1	<----------	Step3		TN2
	4: Establishement of a TCP connection from TN1 port 39527 to TN2 port 55555			
	5: Closing of the TCP connection from from TN1 port 39527 to TN2 port 55555			
	6: Closing of the TCP connection from TN1 port 39526 to TN2 port 21			

5.1.8 DNS-ALG (*)

[RFC2874] defines A6 DNS records. However, because this RFC is in BCP status and because A6 records are not deployed we will put aside this kind of records to focus on AAAA records only

Topology:

In this part, TN1 will be considered as the DNS server from the IPv6 Side and TR will be the DNS server of the outside world.

- TN1 DNS and Reverse DNS Entries will be the following:

Entries	*DNS*
Tn1bis.irisa.fr a3a3.fe00.00ff.0200.0100.ffff.501.3ffe.ip6.int	TN1Bis: 3ffe:501:ffff:100:200:ff:fe00:a3a3 tn1bis.irisa.fr

- TR DNS and Reverse DNS Entries will be the following:

Entries	*DNS*
Tn2.irisa.fr 1.201.254.131.in-addr.arpa	TN2: 131.254.201.1 tn2.irisa.fr

5.1.8.1 DNS Query & DNS Response (*)

Purpose:

Check the correct translation of DNS query and response packet.

References:

- **[RFC2766]** ***Pages 11, 12***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- A DNS-ALG MUST be available on the NUT

Test Requirement:

NONE

Topology:

In this test case, TR will be the DNS server of the outside world.

Discussion:

If TN1 do a name look-up AAAA for TN2, a DNS-ALG available on the NUT MUST forward this unchanged query as well as an A query for TN2. If a AAAA record is available in the reply, it will be forwarded to TN1Bis without modification. Otherwise if there is an A record, The DNS-ALG MUST translate the reply adding the appropriate PREFIX before to forward it to TN1

Packets:

- IPv6/UDP/DNS Query (length: 83 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 43
NextHeader: 17
HopLimit : (64)
SourceAddress: **TN1 IPv6 Global Address**
DestinationAddress: **TR IPv6 Global NAT-PT Address**

UDP (length: 43)

SourcePort: (1000)
DestinationPort: **53**
Length: 43
Checksum: To calculate

DNS (length: 35)

Identifier: (514)
QR: 0
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 1
ANCount: 0
NSCount: 0
ARCount: 0

Question Entry (length: 23)
Name: **TN2 DNS**
Type: 28 **[AAAA]**
Class: 1

- IPv6/UDP/DNS Response (length: 128 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 88
NextHeader: 17
HopLimit : (63) **[one less than in the IPv4/UDP/DNS Response Packet]**
SourceAddress: **TR IPv6 Global NAT-PT Address**
DestinationAddress: **TN1 IPv6 Global Address**

UDP (length: 88)

SourcePort: (1000)
DestinationPort: **53**
Length: 88
Checksum: To calculate

DNS (length: 80)

Identifier: (515)
QR: 1
Opcode: 0
AA: 0

TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 1
ANCount: 1
NSCount: 0
ARCount: 0

DNS Question Entry (length: 23)
Name: **TN2 DNS**
Type: 28 **[AAAA]**
Class: 1

DNS Answer (length: 45)
Name: **TN2 DNS**
Type: 28 **[AAAA]**
Class: 1
TTL: (0)
Length: 16
Address: **TN2 IPv6 Global NAT-PT Address**

- IPv4/UDP/DNS Query (length: 86 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 86
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (63) **[one less than in the IPv6/UDP/DNS Query Packet]**
Protocol: 17
HeaderChecksum: To calculate
SourceAddress: **TN1 IPv4 Address given by NAT-PT Box**
DestinationAddress: **TR IPv4 Address**

UDP (length: 66)

SourcePort: (1000)
DestinationPort: **53**
Length: 66
Checksum: To Calculate

DNS (length: 58)

Identifier: (514)
QR: 0
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 2
ANCount: 0
NSCount: 0
ARCount: 0

DNS Question Entry (length: 23)
Name: **TN2 DNS**

Type: 1 **[A]**
Class: 1

DNS Question Entry (length: 23)
Name: **TN2 DNS**
Type: 28 **[AAAA]**
Class: 1

- IPv4/UDP/DNS Response (length: 119 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 119
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To calculate
SourceAddress: **TR IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by NAT-PT Box**

UDP (length: 99)

SourcePort: (1000)
DestinationPort: **53**
Length: 99
Checksum: To Calculate

DNS (length: 91)

Identifier: (515)
QR: 1
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 2
ANCount: 1
NSCount: 0
ARCount: 0

Question Entry (length: 23)
Name: **TN2 DNS**
Type: 1 **[A]**
Class: 1

Question Entry (length: 23)
Name: **TN2 DNS**
Type: 28 **[AAAA]**
Class: 1

Answer (length: 33)
Name: **TN2 DNS**
Type: 1 **[A]**
Class: 1
TTL: (0)
Length: 4
Address: **TN2 IPv4 Address**

Procedure:

1. TN1 sends “IPv6/UDP/DNS Query” to TR to get IP address of TN2. (Type is AAAA and Name is TN2 DNS).
2. TR sends “IPv4/UDP/DNS Response” to TN1 to give IPv4 address of TN2.

Observable Results:

- **Step1:** The DNS-ALG on the NAT-PT Box Device forwards the AAAA Query to TR in an “IPv4/UDP/DNS Query” Packet. This packet contains the previous AAAA Query as well as an A Query for TN2.
- **Step2:** The DNS-ALG on the NAT-PT Box Device forwards the Answer to TN1 in an “IPv6/UDP/DNS Response” Packet. This packet has been translated adding the appropriate PREFIX to TN2 IPv4 Address.

Test Sequence:

Tester	Link1 [IPv6]	RUT DNS-ALG	Link2 [IPv4]	Tester(IPv6)
TN1	IPv6/UDP/DNS Query ---------->	1		TR
		Step1	IPv4/UDP/DNS Query ---------->	TR
TN1		2	<---------- IPv4/UDP/DNS Response	TR
TN1	<---------- IPv6/UDP/DNS Response	Step2		

5.1.8.2 Inverse DNS Query & DNS Response (*)

Purpose:

Check the correct translation of Inverse DNS query and response packet.

References:

- **[RFC2766]**

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- A DNS-ALG MUST be available on the NUT

Test Requirement:

NONE

Topology:

In this test case, TR will be the DNS server of the outside world.

Discussion:

Since **[RFC3596]** deprecates references to IP6.INT in **[RFC2766]** section 4.1, we have to take into account the use of IP6.ARPA.

If TN1 do a name look-up PTR in order to get DNS Name of TN2, a DNS-ALG available on the NUT MUST replace the string “IP6.ARPA” with “IN-ADDR.ARPA” and the v6 address octet (in reverse order) with the corresponding v4 address octets in reverse order. The same has to be done for the response.

Packets:

- IPv6/UDP/DNS Inverse Query (length: 113 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 73
NextHeader: 17
HopLimit : (64)
SourceAddress: **TN1 IPv6 Global Address**
DestinationAddress: **TR IPv6 Global NAT-PT Address**

UDP (length: 73)

SourcePort: (1000)
DestinationPort: **53**
Length: 73
Checksum: To calculate

DNS (length: 65)

Identifier: (516)
QR: 0
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 1
ANCount: 0

NSCount: 0
ARCount: 0

Question Entry (length: 41)
Name: **TN2 IPv6 Reverse DNS**
Type: 12 **[PTR]**
Class: 1

- IPv6/UDP/DNS Inverse Response (length: 185 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 145
NextHeader: 17
HopLimit : (63) **[one less than in the IPv4/UDP/DNS Inverse Response Packet]**
SourceAddress: **TR IPv6 Global NAT-PT Address**
DestinationAddress: **TN1 IPv6 Global Address**

UDP (length: 145)

SourcePort: (1000)
DestinationPort: **53**
Length: 145
Checksum: To calculate

DNS (length: 137)

Identifier: (517)
QR: 1
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 1
ANCount: 1
NSCount: 0
ARCount: 0

Question Entry (length: 41)
Name: **TN2 IPv6 Reverse DNS**
Type: 12 **[PTR]**
Class: 1

DNS Answer (length: 74)
Name: **TN2 IPv6 Reverse DNS**
Type: 12 **[PTR]**
Class: 1
TTL: (0)
Length: 19
PTRDName: **TN2 DNS**

- IPv4/UDP/DNS Inverse Query (length: 72 bytes)

IPv4 Header (length: 20)

Version: 4

IHL: 5
TypeOfService: (0)
TotalLength: 72
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (63) **[one less than in the IPv6/UDP/DNS Inverse Query Packet]**
Protocol: 17
HeaderChecksum: To calculate
SourceAddress: **TN1 IPv4 Address given by NAT-PT Box**
DestinationAddress: **TR IPv4 Address**

UDP (length: 52)

SourcePort: (1000)
DestinationPort: **53**
Length: 52
Checksum: To Calculate

DNS (length: 45)

Identifier: (516)
QR: 0
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 2
ANCount: 0
NSCount: 0
ARCount: 0

Question Entry (length: 32)
Name: **TN2 IPv4 Reverse DNS**
Type: 12 **[PTR]**
Class: 1

- IPv4/UDP/DNS Inverse Response (length: 129 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 129
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TR IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by NAT-PT Box**

UDP (length: 109)

SourcePort: (1000)
DestinationPort: **53**

Length: 109
Checksum: To Calculate

DNS (length: 101)

Identifier: (517)
QR: 1
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 1
ANCount: 1
NSCount: 0
ARCount: 0

Question Entry (length: 32)
Name: **TN2 IPv4 Reverse DNS**
Type: 12 **[PTR]**
Class: 1

DNS Answer (length: 57)
Name: **TN2 IPv4 Reverse DNS**
Type: 12 **[PTR]**
Class: 1
TTL: (0)
Length: 19
PTRDName: **TN2 DNS**

Procedure:

1. TN1 sends “IPv6/UDP/DNS Reverse Query” to TR to get DNS Name of TN2. (Type is PTR and Name is TN2 IPv6 Reverse DNS).
2. TR sends “IPv4/UDP/DNS Reverse Response” to TN1 to give DNS Name of TN2.

Observable Results:

- **Step1:** The DNS-ALG on the NAT-PT Box Device translates the Query to TR in an “IPv4/UDP/DNS Reverse Query” Packet. This packet contains a PTR request for TN2 IPv4 Reverse DNS.
- **Step2:** The DNS-ALG on the NAT-PT Box Device forwards the Answer to TN1 in an “IPv6/UDP/DNS Reverse Response” Packet.

Test Sequence:

Tester	Link1 [IPv6]	RUT DNS-ALG	Link2 [IPv4]	Tester(IPv6)
	IPv6/UDP/DNS Reverse Query			
TN1	---------->	1		TR
			IPv4/UDP/DNS Reverse Query	
		Step1	---------->	TR
			IPv4/UDP/DNS Reverse Response	
TN1		2	<----------	TR
	IPv6/UDP/DNS Reverse Response			
TN1	<----------	Step2		

5.2 NAPT-PT

NAPT-PT is a variant of NAT-PT that translates transport identifiers such as TCP/UDP port numbers and ICMP identifiers in addition to IP header. As a consequence, NAPT-PT allows multiple IPv6 nodes to communicate with IPv4 nodes using a single public IPv4 address.

5.2.1 Basic NAT-PT

Because NAPT-PT is a variant of NAT-PT that translates transport identifiers, a NAPT-PT implementation MUST run test define in the Basic NAT-PT section (Section 5.1).

5.2.2 Upper Layer Translation

5.2.2.1 UDP Port Translation

Purpose:

Check the correct translation of transport UDP port numbers with NAPT-PT.

References:

- **[RFC2766]** ***section 3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- NAPT-PT has to be activated on the NUT

Discussion:

NAPT-PT is a variant of NAT-PT that translates transport identifiers such as TCP/UDP port numbers and ICMP identifiers in addition to IP header. Thus, in this test all IPv6/UDP traffic sent from TN1Bis and TN1Ter will be forwarded by the NUT using the same IPv4 source address and a different UDP source port.

Packets:

- IPv6/UDP (length: 56 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 16 NextHeader: 17 HopLimit : (64) **[or** (63), **one less than in the corresponding IPv6/UDP Packet if translated by the NUT]** SourceAddress: **…** DestinationAddress: **…**
UDP (length: 16) SourcePort: **…** DestinationPort: **…** Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv4/UDP (length: 36 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0

TTL: (63) **[one less than in the corresponding IPv6/UDP Packet if translated by the NUT; or** (64) **if sent by TN2]** Protocol: 17 HeaderChecksum: To calculate SourceAddress: **…** DestinationAddress: **…**
UDP (length: 16) SourcePort: **…** DestinationPort: **…** Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

1. TN1Bis sends an "IPv6/UDP" packet to TN2 with source port **1000** and destination port 23. The source address is **TN1Bis IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.**
2. TN1Ter sends an "IPv6/UDP" packet to TN2 with source port **1000** and destination port 23. The source address is **TN1Ter IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.**
3. TN1Bis sends an "IPv6/UDP" packet to TN2 with source port **1001** and destination port 23. The source address is **TN1Bis IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.**
4. TR forwards an "IPv4/UDP" packet from TN2 to TN1Bis with source port 23 and destination port **PORT1.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Bis IPv4 Address given by The NUT.**
5. TR forwards an "IPv4/UDP" packet from TN2 to TN1Ter with source port 23 and destination port **PORT2.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Ter IPv4 Address given by The NUT**. (same as **TN1Bis IPv4 Address given by The NUT**).
6. TN2 sends an "IPv4/UDP" packet to TN1Ter with source port 23 and destination port **PORT3.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Ter IPv4 Address given by The NUT**. (same as **TN1Bis IPv4 Address given by The NUT**).

Observable Results:

- **Step 1:** The NUT translates this packet in "IPv4/UDP" with source port **PORT1,** destination port 23 and sends it to TN2. The source address is **TN1Bis IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the UDP checksum has to be adjusted.
- **Step 2:** The NUT translates this packet in "IPv4/UDP" with source port **PORT2,** destination port 23 and sends it to TN2. The source address is **TN1Ter IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the UDP checksum has to be adjusted. In this new packet the UDP checksum has to be adjusted and the source address is the same as in the previous translated packet. **PORT1 and PORT2** MUST be different since the senders are different .
- **Step 3:** The NUT translates this packet in "IPv4/UDP" with source port **PORT3,** destination port 23 and sends it to TN2. The source address is **TN1Bis IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the UDP checksum has to be adjusted.
- **Step 4:** The NUT translates this packet in "IPv6/UDP" with source port 23, destination port **1000** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Bis IPv6 Global Address.**
- **Step 5:** The NUT translates this packet in "IPv6/UDP" with source port 23, destination port **1000** and sends it to TN1Ter. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Bis IPv6 Global Address.**
- **Step 6:** The NUT translates this packet in "IPv6/UDP" with source port 23, destination port **1001** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Bis IPv6 Global Address.**

Test Sequence:

Tester	Link1 [IPv6]	RUT DNS-ALG	Link2 [IPv4]	Tester(IPv6)
TN1Bis	IPv6/UDP **(src:1000,dst:23)** ---------->	1 Step1	IPv4/UDP**(src:PORT1,dst:23)** ---------->	TN2 TN2
TN1Ter	IPv6/UDP **(src:1000,dst:23)** ---------->	2 Step2	IPv4/UDP**(src: PORT2,dst:23)** ---------->	TN2 TN2
TN1Bis	IPv6/UDP **(src:1001,dst:23)** ---------->	3 Step3	IPv4/UDP**(src: PORT3,dst:23)** ---------->	TN2 TN2
TN1Bis	IPv6/UDP **(src:23,dst:1000)** <----------	4 Step4	IPv4/UDP**(src:23,dst: PORT1)** <----------	TN2
TN1Ter	IPv6/UDP **(src:23,dst:1000)** <----------	5 Step5	IPv4/UDP**(src:23,dst: PORT2)** <----------	TN2
TN1Bis	IPv6/UDP **(src:23,dst:1001)** <----------	6 Step6	IPv4/UDP**(src:23,dst: PORT3)** <----------	TN2

5.2.2.2 TCP Port Translation

Purpose:

Check the correct translation of transport TCP port numbers with NAPT-PT.

References:

- **[RFC2766]** ***Section 3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- NAPT-PT has to be activated on the NUT

Discussion:

NAPT-PT is a variant of NAT-PT that translates transport identifiers such as TCP/UDP port numbers and ICMP identifiers in addition to IP header. Thus, in this test all IPv6/TCP traffic sent from TN1Bis and TN1Ter will be forwarded by the NUT using the same IPv4 source address and a different TCP source port.

Packets:

- IPv6/TCP (length: 60 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 20 NextHeader: 6 HopLimit : (64) **[or** (63), **one less than in the corresponding IPv6/UDP Packet if translated by the NUT]** SourceAddress: **…** DestinationAddress: **…**
TCP (length: 20) SourcePort: **…** DestinationPort: **…** SequenceNumber: (0) AcknowledgmentNumber: (0) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: **ACKFlag** PSHFlag: (0) RSTFlag: (0) SYNFlag: **SYNFlag** FINFlag: **FINFlag** Window: (0) Checksum: To calculate UrgentPointer: (0) **Payload (length: 0)**

- IPv4/TCP (length: 40 bytes)

IPv4 Header (length: 20) Version: 4

IHL: 5 TypeOfService: (0) TotalLength: 40 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding IPv6/UDP Packet if translated by the NUT; or**(64) **if sent by TN2]** Protocol: 6 HeaderChecksum: To calculate SourceAddress: **…** DestinationAddress: **…**
TCP (length: 20) SourcePort: DestinationPort: (23) SequenceNumber: **S/N** AcknowledgmentNumber: **A/N** DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: **ACKFlag** PSHFlag: (0) RSTFlag: (0) SYNFlag: **SYNFlag** FINFlag: **FINFlag** Window: (0) Checksum: To calculate UrgentPointer: (0) **Payload (length: 0)**

Procedure:

Open a TCP Connection

1. TN1Bis sends an “IPv6/TCP” packet to TN2 with source port **1000** and destination port 23. The source address is **TN1Bis IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0)
2. TN1Ter sends an “IPv6/TCP” packet to TN2 with source port **1000** and destination port 23. The source address is **TN1Ter IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0)
3. TN1Bis sends an “IPv6/TCP” packet to TN2 with source port **1001** and destination port 23. The source address is **TN1Bis IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0)
4. TR forwards an “IPv4/TCP” packet from TN2 to TN1Bis with source port 23 and destination port **PORT1.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Bis IPv4 Address given by The NUT.** (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
5. TR forwards an “IPv4/TCP” packet from TN2 to TN1Ter with source port 23 and destination port **PORT2.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Ter IPv4 Address given by The NUT.** (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
6. TR forwards an “IPv4/TCP” packet from TN2 to TN1Bis with source port 23 and destination port **PORT3.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Bis IPv4 Address given by The NUT.** (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
7. TN1Bis sends an “IPv6/TCP” packet to TN2 with source port **1000** and destination port 23. The source address is **TN1Bis IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**ACKFlag** is set, **S/N** is set to 2 and **A/N** is set to 11)
8. TN1Ter sends an “IPv6/TCP” packet to TN2 with source port **1000** and destination port 23. The source address is **TN1Ter IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**ACKFlag** is set, **S/N** is set to 2 and **A/N** is set to 11)

9. TN1Bis sends an “IPv6/TCP” packet to TN2 with source port **1001** and destination port 23. The source address is **TN1Bis IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**ACKFlag** is set, **S/N** is set to 2 and **A/N** is set to 11)

Close the TCP Connection

10. TN1Bis sends an “IPv6/TCP” packet to TN2 with source port **1000** and destination port 23. The source address is **TN1Bis IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**FINFlag** and **ACKFlag** are set, **S/N** is set to 2 and **A/N** is set to 11)
11. TN1Ter sends an “IPv6/TCP” packet to TN2 with source port **1000** and destination port 23. The source address is **TN1Ter IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**FINFlag** and **ACKFlag** are set, **S/N** is set to 2 and **A/N** is set to 11)
12. TN1Bis sends an “IPv6/TCP” packet to TN2 with source port **1001** and destination port 23. The source address is **TN1Bis IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**FINFlag** and **ACKFlag** are set, **S/N** is set to 2 and **A/N** is set to 11)
13. TR forwards an “IPv4/TCP” packet from TN2 to TN1Bis with source port 23 and destination port **PORT1.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Bis IPv4 Address given by The NUT.** (**ACKFlag** is set, **S/N** is set to 11 and **A/N** is set to 3)
14. TR forwards an “IPv4/TCP” packet from TN2 to TN1Ter with source port 23 and destination port **PORT2.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Ter IPv4 Address given by The NUT.** (**ACKFlag** is set, **S/N** is set to 11 and **A/N** is set to 3)
15. TR forwards an “IPv4/TCP” packet from TN2 to TN1Bis with source port 23 and destination port **PORT3.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Bis IPv4 Address given by The NUT.** (**ACKFlag** is set, **S/N** is set to 11 and **A/N** is set to 3)
16. TR forwards an “IPv4/TCP” packet from TN2 to TN1Bis with source port 23 and destination port **PORT1.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Bis IPv4 Address given by The NUT.** (**FINFlag** and **ACKFlag** are set, **S/N** is set to 11 and **A/N** is set to 3)
17. TR forwards an “IPv4/TCP” packet from TN2 to TN1Ter with source port 23 and destination port **PORT2.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Ter IPv4 Address given by The NUT.** (**FINFlag** and **ACKFlag** are set, **S/N** is set to 11 and **A/N** is set to 3)
18. TR forwards an “IPv4/TCP” packet from TN2 to TN1Bis with source port 23 and destination port **PORT3.** The source address is **TN2 IPv4 Address** and the destination address is **TN1Bis IPv4 Address given by The NUT.** (**FINFlag** and **ACKFlag** are set, **S/N** is set to 11 and **A/N** is set to 3)
19. TN1Bis sends an “IPv6/TCP” packet to TN2 with source port **1000** and destination port 23. The source address is **TN1Bis IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**ACKFlag** is set, **S/N** is set to 3 and **A/N** is set to 12)
20. TN1Ter sends an “IPv6/TCP” packet to TN2 with source port **1000** and destination port 23. The source address is **TN1Ter IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**ACKFlag** is set, **S/N** is set to 3 and **A/N** is set to 12)
21. TN1Bis sends an “IPv6/TCP” packet to TN2 with source port **1001** and destination port 23. The source address is **TN1Bis IPv6 Global Address** and the destination address is **TN2 IPv6 Global NAT-PT Address.** (**ACKFlag** is set, **S/N** is set to 3 and **A/N** is set to 12)

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv4/TCP” with source port **PORT1,** destination port 23 and sends it to TN2. The source address is **TN1Bis IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0).
- **Step 2:** The NUT translates this packet in “IPv4/TCP” with source port **PORT2,** destination port 23 and sends it to TN2. The source address is **TN1Ter IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0). **PORT1 and PORT2** MUST be different since the senders are different .
- **Step 3:** The NUT translates this packet in “IPv4/TCP” with source port **PORT3,** destination port 23 and sends it to TN2. The source address is **TN1Bis IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0).
- **Step 4:** The NUT translates this packet in “IPv6/TCP” with source port 23, destination port **1000** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Bis IPv6 Global Address.** In this new packet the TCP checksum has to be adjusted. (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)

- **Step 5:** The NUT translates this packet in “IPv6/TCP” with source port 23, destination port **1000** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Ter IPv6 Global Address.** In this new packet the TCP checksum has to be adjusted. (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
- **Step 6:** The NUT translates this packet in “IPv6/TCP” with source port 23, destination port **1001** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Bis IPv6 Global Address.** In this new packet the TCP checksum has to be adjusted. (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
- **Step 7:** The NUT translates this packet in “IPv4/TCP” with source port **PORT1,** destination port 23 and sends it to TN2. The source address is **TN1Bis IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 2 and **A/N** is set to 11)
- **Step 8:** The NUT translates this packet in “IPv4/TCP” with source port **PORT2,** destination port 23 and sends it to TN2. The source address is **TN1Ter IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 2 and **A/N** is set to 11)
- **Step 9:** The NUT translates this packet in “IPv4/TCP” with source port **PORT3,** destination port 23 and sends it to TN2. The source address is **TN1Bis IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 2 and **A/N** is set to 11)
- **Step 10:** The NUT translates this packet in “IPv4/TCP” with source port **PORT1,** destination port 23 and sends it to TN2. The source address is **TN1Bis IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 2 and **A/N** is set to 11)
- **Step 11:** The NUT translates this packet in “IPv4/TCP” with source port **PORT2,** destination port 23 and sends it to TN2. The source address is **TN1Ter IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 2 and **A/N** is set to 11)
- **Step 12:** The NUT translates this packet in “IPv4/TCP” with source port **PORT3,** destination port 23 and sends it to TN2. The source address is **TN1Bis IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 2 and **A/N** is set to 11)
- **Step 13:** The NUT translates this packet in “IPv6/TCP” with source port 23, destination port **1000** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Bis IPv6 Global Address.** In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 11 and **A/N** is set to 3)
- **Step 14:** The NUT translates this packet in “IPv6/TCP” with source port 23, destination port **1000** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Ter IPv6 Global Address.** In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 11 and **A/N** is set to 3)
- **Step 15:** The NUT translates this packet in “IPv6/TCP” with source port 23, destination port **1001** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Bis IPv6 Global Address.** In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 11 and **A/N** is set to 3)
- **Step 16:** The NUT translates this packet in “IPv6/TCP” with source port 23, destination port **1000** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Bis IPv6 Global Address.** In this new packet the TCP checksum has to be adjusted. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 11 and **A/N** is set to 3)
- **Step 17:** The NUT translates this packet in “IPv6/TCP” with source port 23, destination port **1000** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Ter IPv6 Global Address.** In this new packet the TCP checksum has to be adjusted. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 11 and **A/N** is set to 3)
- **Step 18:** The NUT translates this packet in “IPv6/TCP” with source port 23, destination port **1001** and sends it to TN1Bis. The source address is **TN2 IPv6 Global NAT-PT Address** and the destination address is **TN1Bis IPv6 Global Address.** In this new packet the TCP checksum has to be adjusted. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 11 and **A/N** is set to 3)
- **Step 19:** The NUT translates this packet in “IPv4/TCP” with source port **PORT1,** destination port 23 and sends it to TN2. The source address is **TN1Bis IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 3 and **A/N** is set to 12)

- **Step 20:** The NUT translates this packet in “IPv4/TCP” with source port **PORT2,** destination port 23 and sends it to TN2. The source address is **TN1Ter IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 3 and **A/N** is set to 12)
- **Step 21:** The NUT translates this packet in “IPv4/TCP” with source port **PORT3,** destination port 23 and sends it to TN2. The source address is **TN1Bis IPv4 Address given by The NUT** and the destination address is **TN2 IPv4 Address.** In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 3 and **A/N** is set to 12)

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
		Open The TCP Connections		
TN1BIS	IPv6/TCP **(SYN) S/N:1,A/N:0** ----------> **(src:1000,dst:23)**	1 Step1	IPv4/TCP ----------> **(src:PORT1,dst:23)**	TN2 TN2
TN1TER	IPv6/TCP **(SYN) S/N:1,A/N:0** ----------> **(src:1000,dst:23)**	2 Step2	IPv4/TCP ----------> **(src:PORT2,dst:23)**	TN2 TN2
TN1BIS	IPv6/TCP **(SYN) S/N:1,A/N:0** ----------> **(src:1001,dst:23)**	3 Step3	IPv4/TCP ----------> **(src:PORT3,dst:23)**	TN2 TN2
TN1BIS		4	IPv4/TCP **(SYN,ACK) S/N:10,A/N:2** <---------- **(src:23,dst: PORT1)**	TN2
TN1BIS	IPv6/TCP <---------- **(src: 23,dst: 1000)**	Step4		
TN1TER		5	IPv4/TCP **(SYN,ACK) S/N:10,A/N:2** <---------- **(src:23,dst: PORT2)**	TN2
TN1TER	IPv6/TCP <---------- **(src: 23,dst: 1000)**	Step5		
TN1BIS		6	IPv4/TCP **(SYN,ACK) S/N:10,A/N:2** <---------- **(src:23,dst: PORT3)**	TN2
TN1BIS	IPv6/TCP <---------- **(src: 23,dst: 1001)**	Step6		
TN1BIS	IPv6/TCP **(ACK) S/N:2,A/N:11** ----------> **(src: 1000,dst: 23)**	7 Step7	IPv4/TCP ----------> **(src:PORT1,dst:23)**	TN2 TN2
TN1TER	IPv6/TCP **(ACK) S/N:2,A/N:11** ----------> **(src: 1000,dst: 23)**	8 Step8	IPv4/TCP ----------> **(src:PORT2,dst:23)**	TN2 TN2
TN1BIS	IPv6/TCP **(ACK) S/N:2,A/N:11** ----------> **(src: 1001,dst: 23)**	9 Step9	IPv4/TCP ----------> **(src:PORT3,dst:23)**	TN2 TN2

Close The TCP Connections

TN1BIS	IPv6/TCP **(FIN,ACK) S/N:2,A/N:11** ----------> **(src: 1000,dst: 23)**	10 Step10	IPv4/TCP ----------> **(src:PORT1,dst:23)**	TN2 TN2
TN1TER	IPv6/TCP **(FIN,ACK) S/N:2,A/N:11** ----------> **(src: 1000,dst: 23)**	11 Step11	IPv4/TCP ----------> **(src:PORT2,dst:23)**	TN2 TN2
TN1BIS	IPv6/TCP **(FIN,ACK) S/N:2,A/N:11** ----------> **(src: 1001,dst: 23)**	12 Step12	IPv4/TCP ----------> **(src:PORT3,dst:23)**	TN2 TN2
TN1BIS		13	IPv4/TCP **(ACK) S/N:11,A/N:3** <---------- **(src:23,dst: PORT1)**	TN2
TN1BIS	IPv6/TCP <---------- **(src:23,dst: 1000)**	Step13		
TN1TER		14	IPv4/TCP **(ACK) S/N:11,A/N:3** <---------- **(src:23,dst: PORT2)**	TN2
TN1TER	IPv6/TCP <---------- **(src:23,dst: 1000)**	Step14		
TN1BIS		15	IPv4/TCP **(ACK) S/N:11,A/N:3** <---------- **(src:23,dst: PORT3)**	TN2
TN1BIS	IPv6/TCP <---------- **(src:23,dst: 1001)**	Step15		
TN1BIS		16	IPv4/TCP **(FIN,ACK) S/N:11,A/N:3** <---------- **(src:23,dst: PORT1)**	TN2
TN1BIS	IPv6/TCP <---------- **(src:23,dst: 1000)**	Step16		
TN1TER		17	IPv4/TCP **(FIN,ACK) S/N:11,A/N:3** <---------- **(src:23,dst: PORT1)**	TN2
TN1TER	IPv6/TCP <---------- **(src:23,dst: 1000)**	Step17		
TN1BIS		18	IPv4/TCP **(FIN,ACK) S/N:11,A/N:3** <---------- **(src:23,dst: PORT1)**	TN2
TN1BIS	IPv6/TCP <---------- **(src:23,dst: 1001)**	Step18		
TN1BIS	IPv6/TCP **(ACK) S/N:3,A/N:11** ---------->	19		TN2

	(src:1000,dst: 23)			
		Step19	IPv4/TCP ----------> **(src:23,dst: PORT1)**	TN2
TN1TER	IPv6/TCP **(ACK) S/N:3,A/N:11** ----------> **(src:1000,dst: 23)**	20		TN2
		Step20	IPv4/TCP ----------> **(src:23,dst: PORT2)**	TN2
TN1BIS	IPv6/TCP **(ACK) S/N:3,A/N:11** ----------> **(src:1001,dst: 23)**	21		TN2
		Step21	IPv4/TCP ----------> **(src:23,dst: PORT3)**	TN2

5.2.3 ICMPv6 Translation

5.2.3.1 ICMPv6 Identifier Translation

Purpose:

Check the correct translation of ICMPv6 Identifiers with NAPT-PT.

References:

- **[RFC2766]** ***Section 2.2.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- NAPT-PT has to be activated on the NUT

Discussion:

NAPT-PT is a variant of NAT-PT that translates transport identifiers such as TCP/UDP port numbers and ICMP identifiers in addition to IP header. Thus, in this test all ICMPv6 traffic sent from TN1Bis and TN1Ter will be forwarded by the NUT using the same IPv4 source address and a different ICMP identifier.

The ICMP checksum should be adjusted to account for the address and port change. All ICMPv6 Informational messages except “ICMPv6 Echo Request” and “ICMPv6 Echo Reply” MUST be silently dropped.

Packets:

- ICMPv6 Echo Request (length: 56 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 16 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1Bis/TN1Ter IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Echo Request (length: 16) Type: 128 Code: **5** Checksum: To calculate Identifier: **1000** SequenceNumber: (0) **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv6 Echo Reply (length: 56 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 16 NextHeader: 58 HopLimit : (63) **[one less than in the ICMPv4 Echo Reply]**

SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1Bis/TN1Ter IPv6 Global Address**
ICMPv6 Echo Reply (length: 16) Type: 129 Code: **5** Checksum: To calculate Identifier: **1000** SequenceNumber: (0) **[Same as in the ICMPv4 Echo Reply]** **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv4 Echo Reply (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1Bis/TN1Ter IPv4 Address given by The NUT**
ICMPv4 Echo Reply (length: 16) Type: 0 Code: **5** Checksum: To Calculate Identifier: **ID1/ID2** SequenceNumber: (0) **[Same as in the ICMPv6 Echo Request]** **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv4 Echo Request (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the ICMPv6 Echo Request]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1Bis/TN1Ter IPv4 Address given by The**

NUT DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Echo Request (length: 16) Type: 8 Code: **5** Checksum: To Calculate Identifier: **ID1/ID2** SequenceNumber: (0) **[Same as in the ICMPv6 Echo Request]** **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

- TN1Bis sends an “ICMPv6 Echo Request” to TN2 with identifier **1000**
- TN1Ter sends an “ICMPv6 Echo Request” to TN2 with identifier **1000**
- TN1Bis sends an “ICMPv6 Echo Request” to TN2 with identifier **1001**
- TR forwards an “ICMPv4 Echo Reply” from TN2 to TN1Bis with identifier **1000**
- TR forwards an “ICMPv4 Echo Reply” from TN2 to TN1Ter with identifier **1000**
- TR forwards an “ICMPv4 Echo Reply” from TN2 to TN1Bis with identifier **1001**

Observable Results:

- **Step 1:** The NUT translates this packet in “ICMPv4 Echo Request” with identifier **ID1** and sends it to TN2. In this new packet the ICMP checksum has to be adjusted.
- **Step 2:** The NUT translates this packet in “ICMPv4 Echo Request” with identifier **ID2** and sends it to TN2. In this new packet the ICMP checksum has to be adjusted. **ID1** and **ID2** MUST be different.
- **Step 3:** The NUT translates this packet in “ICMPv4 Echo Request” with identifier **ID3** and sends it to TN2. In this new packet the ICMP checksum has to be adjusted.
- **Step 4:** The NUT translates this packet in “ICMPv6 Echo Reply” with identifier **1000** and sends it to TN1Bis. In this new packet the ICMP checksum has to be adjusted.
- **Step 5:** The NUT translates this packet in “ICMPv6 Echo Reply” with identifier **1000** and sends it to TN1Ter. In this new packet the ICMP checksum has to be adjusted.
- **Step 6:** The NUT translates this packet in “ICMPv6 Echo Reply” with identifier **1001** and sends it to TN1Bis. In this new packet the ICMP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT DNS-ALG	Link2 [IPv4]	Tester(IPv6)
TN1Bis	ICMPv6 Echo Request **(ID: 1000)** ---------->	1 Step1	ICMPv4 Echo Request **(ID: ID1)** ---------->	TN2 TN2
TN1Ter	ICMPv6 Echo Request **(ID: 1000)** ---------->	2 Step2	ICMPv4 Echo Request **(ID: ID2)** ---------->	TN2 TN2
TN1Bis	ICMPv6 Echo Request **(ID: 1001)** ---------->	3 Step3	ICMPv4 Echo Request **(ID: ID3)** ---------->	TN2 TN2
TN1Bis	ICMPv6 Echo Reply **(ID: 1000)** <----------	4 Step4	ICMPv4 Echo Reply **(ID: ID1)** <----------	TN2
TN1Ter	ICMPv6 Echo Reply **(ID: 1000)** <----------	5 Step5	ICMPv4 Echo Reply **(ID: ID2)** <----------	TN2
TN1Bis	ICMPv6 Echo Reply **(ID: 1001)** <----------	6 Step6	ICMPv4 Echo Reply **(ID: ID3)** <----------	TN2

5.2.4 FTP-ALG Extension

5.2.4.1 TCP Port translation in EPRT command

Purpose:

Check the correct translation of tcp port with EPRT Command

References:

- **[RFC2766]** ***Section 6.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- NAPT-PT has to be activated on the NUT
- A FTP-ALG MUST be available on the NUT

Test Requirement:

NONE

Discussion:

The main goal of this test is to check translation of EPRT Commands in the case of NAPT-PT. As defined in 0, TCP Port from the IPv6 Side will be translated. Moreover, TCP ports in commands arguments have also to be translated.

Procedure:

1. Establish a TCP connection from TN1Bis and TN1Ter port 39626 to TN2 port 21 as defined in 5.1.2.1
2. Do the previous test concerning EPRT Command (Section 5.1.7.1) using TN1Bis and TN1Ter with:
   - a TCP connection from TN2 port 20 to TN1Bis port **PORT1**
   - a TCP connection from TN2 port 20 to TN1Ter port **PORT2**
4. Close the TCP connection from TN2 port 20 to TN1Bis and TN1Ter port **PORT1** and **PORT2** as defined in 0
5. Close the TCP connection from TN1Bis and TN1Ter port 39626 to TN2 port 21 as defined in 5.1.2.1

Observable Results:

- **Step 2:**
  - As defined in 0, TCP Port from the IPv6 Side will be translated in TCP packets used for the connection. We remind that TN1Bis and TN1Ter will use the same IPv4 address.
  - The IPv4/TCP/FTP EPRT Packet forwarded by the NUT will use the same IPv4 source address for each FTP session but two distinct ports will be given in the EPRT Command. The ports will be different for each IPv6 address. We will have **PORT1** and **PORT2**
  - The Connection will be established with TN1Bis and TN1Ter using port 39627.

5.2.4.2 TCP Port translation in EPSV command

Purpose:

Check the correct translation correct tcp port translation with EPSV Command

References:

- **[RFC2766]** ***Section 6.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- NAPT-PT has to be activated on the NUT
- A FTP-ALG MUST be available on the NUT

Test Requirement:

NONE

Discussion:

The main goal of this test is to check translation of EPSV Commands in the case of NAPT-PT. As defined in 0, TCP Port from the IPv6 Side will be translated. Moreover, TCP ports in commands arguments have also to be translated.

Procedure:

1. Establish a TCP connection from TN1Bis and TN1Ter port 39626 to TN2 port 21 as defined in 5.1.2.1
2. Do the previous test concerning EPSV Command (Section 5.1.7.2) using TN1Bis and TN1Ter with
   - a TCP connection from TN1Bis port 39627 to TN2 port 55555
   - a TCP connection from TN1Ter port 39627 to TN2 port 55555
3. Close the TCP connection from TN1Bis and TN1Ter port 39627 to TN2 port 55555
4. Close the TCP connection from TN1Bis and TN1Ter port 39626 to TN2 port 21 as defined in 5.1.2.1

Observable Results:

- **Step 2:** As defined in 0, TCP Port from the IPv6 Side will be translated in TCP packets used for the connection. We remind that TN1Bis and TN1Ter will use the same IPv4 address. Thus, two distinct ports will be use for FTP data connections from the IPv4 side. The ports will be different for each IPv6 address. We will have **PORT1** and **PORT2**

  Observable Results will be identical to those described in Section 5.1.7.2 taking into account TN1Bis and TN1Ter and the TCP port translation described in Section 0 of the TCP Header.

5.3 Bi-directional-NAT-PT

With Bi-directional-NAT-PT, sessions can be initiated from hosts in V4 network as well as the V6 network. V6 network addresses are bound to V4 addresses, statically or dynamically as connections are established in either direction. For V4 Address assignment for incoming connections (V4 to V6) at least two solutions may be choosed:

- Either a DNS Request is done from a v4 host for the corresponding v6 host. If a V4 address is not previously assigned to this V6 node, NAT-PT would assign one at this time.
- Either we create a State in the NUT for the corresponding v6 hosts before testing. As a Consequent, further sessions could be initiated from hosts in V4 network. This State may be established either statically either dynamically.

The following packet may be used to establish dynamically a state for TN in the NUT prior to test. This packet will be sent from TN to TN2.

- IPv6/UDP STATE (length: 56 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 16 NextHeader: 17 HopLimit : (64) SourceAddress: **TN IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

Bi-directional NAT-PT is not a strict requirement. In its simplest form, NAT-PT is only a one way connectivity. Nevertheless, an implementation which supports Bi-directional NAT-PT must include one of the Unidirectionnal NAT-PT mechanism: i.e. either Basic NAT-PT or NAPT-PT.

5.3.1 Unidirectionnal NAT-PT

Because an implementation which supports Bi-directional NAT-PT must include one of the Unidirectionnal NAT-PT mechanism, test for this mechanism has to be run prior specifics test cases for Bi-directional NAT-PT. It means that according to the mechanism used, either test cases for Basic NAT-PT or NAPT-PT MUST be run.

5.3.2 Basic Translation

5.3.2.1 TOS fields from IPv4 Header

Purpose:

Check the correct translation of the TOS Field.

References:

- **[RFC2765]** ***Section 3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1
- The implementation should have the possibility to ignore the IPv4 TOS field and always set the IPv6 traffic class to zero.

Discussion:

The IPv6 Traffic Class field is by default copied from the IPv4 TOS field, Nevertheless, all implementations SHOULD provide the ability to ignore the IPv4 TOS field class and always set the IPv6 Traffic Class field to 0.

Packets:

- IPv4/UDP (length: 36 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: **TOS**
TotalLength: 36
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To Calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- IPv6/UDP (length: 56 bytes)

IPv6 Header (length: 40)

Version: 6

TrafficClass: **TRAFFIC_CLASS** FlowLabel: 0 PayloadLength: 16 NextHeader: 17 HopLimit : (63) **[one less than in the IPv6/UDP 1]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To Calculate **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

0. Create a State in the NUT for TN1
1. TR forwards an “IPv4/UDP” Packet from TN2 to TN1 with TOS Field set to 255.
2. The Implementation activates the ability to ignore the IPv4 TOS field and always set the IPv6 Traffic Class field to 0.
3. TR forwards an “IPv4/UDP” Packet from TN2 to TN1 with TOS Field set to 255.
4. The Implementation desactivates the ability to ignore the IPv4 TOS field.

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv6/UDP” with TOS also set to 255 and sends it to TN1
- **Step 3:** The NUT translates this packet in “IPv6/UDP” with TOS set to 0 and sends it to TN1

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/UDP (TOS = 255)	
TN1		1	<----------	TN2
	IPv6/UDP (Traffic Class = 255)			
TN1	<----------	Step1		
	2: Activation of the ability to ignore the IPv4 TOS field			
			IPv4/UDP (TOS = 255)	
TN1		3	<----------	TN2
	IPv6/UDP (Traffic Class = 0)			
TN1	<----------	Step3		
	4: Desactivation of the ability to ignore the IPv4 TOS field			

5.3.2.2 IPv4 Options

Purpose:

Check translation of IPv4 Options

References:

- **[RFC2765]** ***Section 3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

If IPv4 options are present in the IPv4 packet, they are ignored i.e., there is no attempt to translate them. However, if an unexpired source route option is present then the packet MUST instead be discarded, and an ICMPv4 "destination unreachable/source route failed" (Type 3/Code 5) error message SHOULD be returned to the sender.

Packets:

- IPv4/UDP 1 (with Timestamp Option) (length: 44 bytes)

IPv4 Header (length: 28)

Version: 4
IHL: 7
TypeOfService: (0)
TotalLength: 44
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

Timestamp Option (length:8)
Type: 68
Length: 8
Pointer:5
Overflow: 0
Flag: 0
Timestamp: 170

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To Calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- IPv4/UDP 2 (with Record Route Option) (length: 48 bytes)

IPv4 Header (length: 32)

Version: 4
IHL: 8
TypeOfService: (0)
TotalLength: 48
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

NoOperation Option (length:1)
Type: 1

Record Route Option (length:11)
Type: 7
Length: 11
Pointer:4
RouteData: **0.0.0.0**
RouteData: **0.0.0.0**

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To Calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- IPv4/UDP 3 (with Loose Source Route Option) (length: 44 bytes)

IPv4 Header (length: 28)

Version: 4
IHL: 7
TypeOfService: (0)
TotalLength: 44
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

NoOperation Option (length:1)
Type: 1

Loose Source Route Option (length:7)
Type: 131
Length: 7
Pointer:4
RouteData: **TN1 IPv4 Address given by The NUT**

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To Calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- IPv4/UDP 4 (with Expired Loose Source Route Option) (length: 44 bytes)

IPv4 Header (length: 28)

Version: 4
IHL: 7
TypeOfService: (0)
TotalLength: 44
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

NoOperation Option (length:1)
Type: 1

Loose Source Route Option (length:7)
Type: 131
Length: 7
Pointer:8
RouteData: **TN2 IPv4 Address**

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To Calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- IPv4/UDP 5 (with Strict Source Route Option) (length: 44 bytes)

IPv4 Header (length: 28)

Version: 4
IHL: 7
TypeOfService: (0)
TotalLength: 44
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**

DestinationAddress: **TN1 IPv4 Address given by The NUT**

NoOperation Option (length:1)
Type: 1

Strict Source Route Option (length:7)
Type: 137
Length: 7
Pointer:4
RouteData: **TN1 IPv4 Address given by The NUT**

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To Calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- IPv4/UDP 6 (with Expired Strict Source Route Option) (length: 44 bytes)

IPv4 Header (length: 28)

Version: 4
IHL: 7
TypeOfService: (0)
TotalLength: 44
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

NoOperation Option (length:1)
Type: 1

Strict Source Route Option (length:7)
Type: 137
Length: 7
Pointer:8
RouteData: **TN2 IPv4 Address**

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To Calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- IPv4/UDP 7 (with Opt: Timestamp, strict source routing expired, record route, end of list) (length: 68 bytes)

IPv4 Header (length: 52)

Version: 4 IHL: 13 TypeOfService: (0) TotalLength: 68 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT** **NoOperation Option (length:1)** Type: 1 **Strict Source Route Option (length:7)** Type: 137 Length: 7 Pointer:8 RouteData: **TN2 IPv4 Address** **Timestamp Option (length:8)** Type: 68 Length: 8 Pointer:5 Overflow: 0 Flag: 0 Timestamp: 170 **NoOperation Option (length:1)** Type: 1 **Record Route Option (length:11)** Type: 7 Length: 11 Pointer:4 RouteData: **0.0.0.0** RouteData: **0.0.0.0** **EndofOptionList (length:1)** Type: 0 **Padding = 000000**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To Calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv4 Destination Unreachable (length: 64 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 64 Identifier: XXXXX Reserved: 0

DF: 1 MF: 0 FragmentOffset: 0 TTL: XXXXX Protocol: 1 HeaderChecksum: XXXXX SourceAddress: **NUT IPv4 Address Link2** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Destination Unreachable (length: 44) Type: 3 Code: 5 Checksum: To Calculate Unused: 0 **Payload (length: 36)** **The Ipv4 Header plus the first 8 bytes of IPv4/UDP 3 (with Loose Source Route Option)** Or **The Ipv4 Header plus the first 8 bytes of IPv4/UDP 5 (with Strict Source Route Option)**

- IPv6/UDP (length: 56 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 16 NextHeader: 17 HopLimit : (63) **[one less than in the corresponding IPv4/UDP]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To Calculate **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

0. Create a State in the NUT for TN1
1. TN2 Sends “IPv4/UDP 1” Packet (with Opt Timestamp) to TN1.
2. TN2 Sends “IPv4/UDP 2” Packet (with Opt Record Route) to TN1.
3. TN2 Sends “IPv4/UDP 3” Packet (with Opt Loose Route) to TN1.
4. TN2 Sends “IPv4/UDP 4” Packet (with Opt Loose Route expired) to TN1.
5. TN2 Sends “IPv4/UDP 5” Packet (with Opt Strict Route) to TN1.
6. TN2 Sends “IPv4/UDP 6” Packet (with Opt Strict Route Expired) to TN1.
7. TN2 Sends “IPv4/UDP 7”Packet (with Opt: Timestamp, strict source routing expired, record route, end of list) to TN1.

Observable Results:

- **Step 1:** The NUT must discard this option and forward the translated packet “IPv6/UDP” to TN1.
- **Step 2:** The NUT must discard this option and forward the translated packet “IPv6/UDP” to TN1.
- **Step 3:** TN2 should receive “ICMPv4 Destination Unreachable” with IPv4/UDP 3 as payload from The NUT (type 3, code 5).
- **Step 4:** The NUT must discard this option and forward the translated packet “IPv6/UDP” to TN1
- **Step 5:** TN2 should receive “ICMPv4 Destination Unreachable” with IPv4/UDP 5 as payload from The NUT (type 3, code 5).
- **Step 6:** The NUT must discard this option and forward the translated packet “IPv6/UDP” to TN1.
- **Step 7:** The NUT must discard these options and forward the translated “IPv6/UDP” packet to TN1

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN1		1	IPv4/UDP 1 **(with Opt Timestamp)** <----------	TN2
TN1	IPv6/UDP <----------	Step1		
TN1		2	IPv4/UDP 2 **(with Opt Record Route)** <----------	TN2
TN1	IPv6/UDP <----------	Step2		
TN1		3	IPv4/UDP 3 **(with Opt Loose Route)** <----------	TN2
		Step3	ICMPv4 Destination Unreachable (type 3, code 5) ---------->	TN2
TN1		4	IPv4/UDP 4 **(with Opt Loose Route expired)** <----------	TN2
TN1	IPv6/UDP <----------	Step4		
TN1		5	IPv4/UDP 5 **(with Opt Strict Route)** <----------	TN2
		Step5	ICMPv4 Destination Unreachable (type 3, code 5). ---------->	TN2
TN1		6	IPv4/UDP 6 **(with Opt Strict Route Expired)** <----------	TN2
TN1	IPv6/UDP <----------	Step6		
TN1		7	IPv4/UDP 7 **(with Opt: Timestamp, strict source routing expired, record route, end of list)** <----------	TN2
TN1	IPv6/UDP <----------	Step7		

5.3.3 Upper Layer Translation

5.3.3.1 UDP Packet without UDP checksum

Purpose:

Check translation of IPv4/UDP Packet with checksum set to 0.

References:

- **[RFC2765],** ***Section 3.2***
- **[RFC2766],** ***Section 5.3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

UDP Checksum may be set to 0 if used with IPv4. It is not the case in IPv6. This test checks the correct calculation of IPv6 UDP Checksum when the UDP checksum was set to 0 in the IPv4 Packet to translate.

Packets:

- IPv4/UDP (With UDP Checksum set to 0) (length: 36 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: **0** **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv6/UDP (length: 56 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0)

FlowLabel: 0 PayloadLength: 16 NextHeader: 17 HopLimit : (63) **[one less than in the corresponding IPv4/UDP]** SourceAddress: **TN2 Global IPv6 NAT-PT Address** DestinationAddress: **TN1 Global IPv6 Address**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: Calculated UDP Checksum **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

0. Create a State in the NUT for TN1
1. TN2 Sends “IPv4/UDP” Packet to TN1. The UDP Checksum is set to 0.

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv6/UDP” Packet and sends it to TN1. In this packet, the UDP checksum has to be calculated.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/UDP **(UDP Checksum == 0)**	
TN1		1	<----------	TN2
	IPv6/UDP			
TN1	<----------	Step1		

5.3.3.2 TCP Translation

Purpose:

Check correct translation of TCP Packets

References:

- **[RFC2766], *Section 5.3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

The TCP checksum should be adjusted to account for the address changes, going from V4 to V6 addresses.

Packets:

- IPv4/TCP (length: 40 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: 0
TotalLength: 40
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 6
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

TCP (length: 28)

SourcePort: (1000)
DestinationPort: (23)
SequenceNumber: **SequenceNumber**
AcknowledgmentNumber: **AcknowledgmentNumber**
DataOffset: 5
Reserverd: (0)
URGFlag: (0)
ACKFlag: **ACKFlag**
PSHFlag: (0)
RSTFlag: (0)
SYNFlag: **SYNFlag**
FINFlag: **FINFlag**
Window: (0)
Checksum: To calculate
UrgentPointer: (0)

Payload (length: 0)

- IPv6/TCP (length: 60 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 20 NextHeader: 6 HopLimit : (63) **[one less than in the corresponding IPv4/TCP]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
TCP (length: 20) SourcePort: (1000) DestinationPort: (23) SequenceNumber: **SequenceNumber** AcknowledgmentNumber: **AcknowledgmentNumber** DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: **ACKFlag** PSHFlag: (0) RSTFlag: (0) SYNFlag: **SYNFlag** FINFlag: **FINFlag** Window: (0) Checksum: To calculate UrgentPointer: (0) **Payload (length: 0)**

Procedure:

0. Create a State in the NUT for TN1

Open a TCP Connection

1. TN2 sends an "IPv4/TCP" packet to TN1. (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0)
2. TN1 sends an "IPv6/TCP" packet to TN2. (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
3. TN2 sends an "IPv4/TCP" packet to TN1. (**ACKFlag** is set, **S/N** is set to 2 and **A/N** is set to 11)

Close the TCP Connection

4. TN2 sends an "IPv4/TCP" packet to TN1. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 2 and **A/N** is set to 11)
5. TN1 sends an "IPv6/TCP" packet to TN2. (**ACKFlag** is set, **S/N** is set to 11 and **A/N** is set to 3)
6. TN1 sends an "IPv6/TCP" packet to TN2. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 11 and **A/N** is set to 3)
7. TN2 sends an "IPv4/TCP" packet to TN1. (**ACKFlag** is set, **S/N** is set to 3 and **A/N** is set to 12)

Observable Results:

- **Step 1:** The NUT translates this packet in "IPv6/TCP" and sends it to TN1. In this new packet the TCP checksum has to be adjusted. (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0)
- **Step 2:** The NUT translates this packet in "IPv4/TCP" and sends it to TN2. In this new packet the TCP checksum has to be adjusted. (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
- **Step 3:** The NUT translates this packet in "IPv6/TCP" and sends it to TN1. In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 2 and **A/N** is set to 11)
- **Step 4:** The NUT translates this packet in "IPv6/TCP" and sends it to TN1. In this new packet the TCP checksum has to be adjusted. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 2 and **A/N** is set to 11)
- **Step 5:** The NUT translates this packet in "IPv4/TCP" and sends it to TN2. In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 11 and **A/N** is set to 3)
- **Step 6:** The NUT translates this packet in "IPv4/TCP" and sends it to TN2. In this new packet the TCP checksum has to be adjusted. (**FINFlag** and **ACKFlag** are set, **S/N** is set to 11 and **A/N** is set to 3)

- **Step 7:** The NUT translates this packet in “IPv6/TCP” and sends it to TN1. In this new packet the TCP checksum has to be adjusted. (**ACKFlag** is set, **S/N** is set to 3 and **A/N** is set to 12)

Test Sequence:

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
		Open The TCP Connection		
TN1		1	IPv4/TCP **(SYN) S/N:1,A/N:0** <----------	TN2
TN1	IPv6/TCP <----------	Step1		
TN1	IPv6/TCP **(SYN,ACK) S/N:10,A/N:2** ---------->	2		TN2
		Step2	IPv4/TCP ---------->	TN2
TN1		3	IPv4/TCP **(ACK) S/N:2,A/N:11** <----------	TN2
TN1	IPv6/TCP <----------	Step3		
		Close The TCP Connection		
TN1		4	IPv4/TCP **(FIN,ACK) S/N:2,A/N:11** <----------	TN2
TN1	IPv6/TCP <----------	Step4		
TN1	IPv6/TCP **(ACK) S/N:11,A/N:3** ---------->	5		TN2
		Step5	IPv4/TCP ---------->	TN2
TN1	IPv6/TCP **(FIN,ACK) S/N:11,A/N:3** ---------->	6		TN2
		Step6	IPv4/TCP ---------->	TN2
TN1		7	IPv4/TCP **(ACK) S/N:3,A/N:11** <----------	TN2
TN1	IPv6/TCP <----------	Step7		TN2

5.3.4 ICMPv4 Translation

An ICMPv4 Error Packet has sometimes an IPv4 Packet in its payload. In this part we will consider that the translation of the inner IP header has to be done. Indeed, what is the need to translate an ICMPv4 error message if the inner part is not also translated?

The ICMP message format is specified by the value of the Type field:

0	Echo reply.
1	Reserved.
2	Reserved.
3	Destination unreachable.
4	Source quench.
5	Redirect.
6	Alternate Host Address.
8	Echo request.
9	Router advertisement.
10	Router solicitation.
11	Time exceeded.
12	Parameter problem.
13	Timestamp request.
14	Timestamp reply.
15	Information request.
16	Information reply.
17	Address mask request.
18	Address mask reply.
19	Reserved (for security).
20-29	Reserved (for robustness experiment).
30	Traceroute.
31	Conversion error.
32	Mobile Host Redirect.
33	IPv6 Where-Are-You.
34	IPv6 I-Am-Here.
35	Mobile Registration Request.
36	Mobile Registration Reply.
37	Domain Name request.
38	Domain Name reply.
39	SKIP Algorithm Discovery Protocol.
40	Photuris, Security failures.
41-255	Reserved.

5.3.4.1 ICMPv4 Informational Messages

Purpose:

Check translation of ICMPv4 Informational Messages.

References:

- **[RFC2765],** ***Section 3.3***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

This test checks the correct translation of ICMPv4 Informational Messages. The ICMP checksum should be adjusted to account for the address change, going from V4 to V6 addresses. All ICMPv4 Informational messages except “ICMPv4 Echo Request” and “ICMPv4 Echo Reply” MUST be silently dropped.

Packets:

- ICMPv4 Echo Request (length: 36 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 36 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Echo Request (length: 16) Type: 8 Code: 0 Checksum: To calculate Identifier: (612) SequenceNumber: (0) **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv4 Echo Reply (length: 36 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 36 Identifier: (0)

Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Echo Reply]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Echo Reply (length: 16) Type: 0 Code: 0 Checksum: To Calculate Identifier: (612) **[same as in the corresponding ICMPv4 Echo Request]** SequenceNumber: (0) **[same as in the corresponding ICMPv4 Echo Request]** **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv4 Information Request (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 36 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Information Request (length: 16) Type: 15 Code: 0 Checksum: To Calculate Identifier: (0) SequenceNumber: (0)

- ICMPv4 Information Reply (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 36 Identifier: (0) Reserved: 0 DF: 1 MF: 0

FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Information Reply (length: 16) Type: 16 Code: 0 Checksum: To Calculate Identifier: (0) SequenceNumber: (0)

- ICMPv4 Timestamp Request (length: 40 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 40 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Timestamp Request (length: 20) Type: 13 Code: 0 Checksum: To Calculate Identifier: (0) SequenceNumber: (0) OriginateTimestamp: 0 Receive timestamp: 0 Transmit timestamp: 0

- ICMPv4 Timestamp Reply (length: 40 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 40 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**

ICMPv4 Timestamp Reply (length: 20) Type: 14 Code: 0 Checksum: To Calculate Identifier: (0) SequenceNumber: (0) OriginateTimestamp: 0 Receive timestamp: 0 Transmit timestamp: 0

- ICMPv4 Address Mask Request (length: 32 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 32 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Address Mask Request (length: 12) Type: 17 Code: 0 Checksum: To Calculate Identifier: (0) SequenceNumber: (0) Address mask:: 0xFFFFFF00

- ICMPv4 Address Mask Reply (length: 32 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 32 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Address Mask Request (length: 12) Type: 17

Code: 0 Checksum: To Calculate Identifier: (0) SequenceNumber: (0) Address mask: **0xFFFFFF00**

- ICMPv4 Router Advertisement (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 36 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Router Advertisement (length: 16) Type: 17 Code: 0 Checksum: To Calculate Advertisement count: Address Entry size: Lifetime: Address[0]: Address[1]:

- ICMPv4 Router Solicitation (length: 28 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 28 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Router Solicitation (length: 8) Type: 17 Code: 0 Reserved: 0

- ICMPv4 Unknown Informational Messages (length: 28 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: 0
TotalLength: 28
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 1
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

ICMPv4 (length: 8)

Type: **TYPE**
Code: **CODE**
Checksum: To calculate
Data: 0

- ICMPv6 Echo Request (length: 56 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: 0
PayloadLength: 16
NextHeader: 58
HopLimit : (63) **[one less than in the corresponding ICMPv4 Echo Request]**
SourceAddress: **TN2 IPv6 Global NAT-PT Address**
DestinationAddress: **TN1 IPv6 Global Address**

ICMPv6 Echo Request (length: 16)

Type: 128
Code: 0
Checksum: To calculate
Identifier: (612) **[same as in the corresponding ICMPv4 Echo Request]**
SequenceNumber: (0) **[same as in the corresponding ICMPv4 Echo Request]**

Payload (length: 8)
Data: (01234567 89abcdef)

- ICMPv6 Echo Reply (length: 56 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: 0
PayloadLength: 16
NextHeader: 58
HopLimit : (64)
SourceAddress: **TN2 IPv6 Global NAT-PT Address**

DestinationAddress: **TN1 IPv6 Global Address**
ICMPv6 Echo Reply (length: 16) Type: 129 Code: 0 Checksum: To calculate Identifier: (612) **[same as in the corresponding ICMPv4 Echo Request]** SequenceNumber: (0) **[same as in the corresponding ICMPv4 Echo Request]** **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

0. Create a State in the NUT for TN1
1. TN2 sends “ICMPv4 Echo Request” to TN1
2. TN1 sends “ICMPv6 Echo Reply” to TN2
3. TN2 sends the following packets to TN1:
   - “ICMPv4 Information Request”
   - “ICMPv4 Information Reply”
   - “ICMPv4 Timestamp Request”
   - “ICMPv4 Timestamp Reply”
   - “ICMPv4 Mask Request”
   - “ICMPv4 Mask Reply”
   - “ICMPv4 router advertisement”
   - “ICMPv4 router solicitation”
   - some “ICMPv4 Unknown Informational Messages” with different values “**TYPE**” and “**CODE**” for the type and code fields

Observable Results:

- **Step 1:** The NUT translates this packet in “ICMPv6 Echo Request” and sends it to TN1. In this packet, the ICMP checksum has to be adjusted.
- **Step 2:** The NUT translates this packet in “ICMPv4 Echo Reply” and sends it to TN2. In this packet, the ICMP checksum has to be adjusted.
- **Step 3:** These Packets must be silently dropped except for the cases indicated in the table presented hereafter:

ICMPv4 Packet	ICMPv6 Corresponding Packet
Type = 0 (Echo Request)	Type = 128 (Echo Request)
Type = 8 (Echo Reply)	Type = 129 (Echo Reply)

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN1		1	ICMPv4 Echo Request <----------	TN2
TN1	ICMPv6 Echo Request <----------	Step1		
TN1	ICMPv6 Echo Reply ---------->	2		TN2
		Step2	ICMPv4 Echo Reply ---------->	TN2
TN1			• ICMPv4 Information Request • ICMPv4 Information Reply • ICMPv4 Timestamp Request • ICMPv4 Timestamp Reply • ICMPv4 Mask Request • ICMPv4 Mask Reply • ICMPv4 router advertisement • ICMPv4 router solicitation • some “ICMPv4 Unknown Informational Messages” with different values “**TYPE**” and “**CODE**” for the type and code fields <----------	
		3		TN2

5.3.4.2 ICMPv4 Error Messages with UDP packet in payload

Purpose:

Check translation of ICMPv4 Error Messages with a UDP packet in payload

References:

- **[RFC2765],** ***Section 3.3***
- **[RFC 2765]** ***Page 15 :***

"….

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers."

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

This test checks the correct translation of ICMPv4 Error Messages with a UDP packet included in payload. The ICMP checksum should be adjusted to account for the address change, going from V4 to V6 addresses. The inner IP header has also to be translated.

Packets:

- IPv6/UDP (length: 48 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 8 NextHeader: 17 HopLimit : (64) **[Same as in IPv4/UDP]** SourceAddress: **TN1 IPv6 Global Address or TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address or TN1 IPv6 Global Address**
UDP (length: 8) SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To calculate

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 / ICMPv6 Error Type 3 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]**

SourceAddress: **TR IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big / ICMPv6 Time Exceeded **(length: 56)** Type: **TYPE** Code: **CODE** Checksum: To calculate Unused: 0 **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 4 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]** SourceAddress: **TR IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
ICMPv6 Parameter Problem (length: 56) Type: **4** Code: **CODE** Checksum: To calculate Pointer: **POINTER** **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- IPv4/UDP(length: 28 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 28 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT or TN2 IPv4 Address** DestinationAddress: **TN2 IPv4 Address or TN1 IPv4 Address given by The NUT**

UDP (length: 8) SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To Calculate

- ICMPv4 Error Type 3 / ICMPv4 Error Type 11 (length: 56 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 56 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TR IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Destination Unreachable **Or** **ICMPv4 Time Exceeded** **(length: 36)** Type: **3 or 11** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 28)** Data: **Same as IPv4/UDP Packet**

- ICMPv4 Error Type 12 (length: 56 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 56 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TR IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Parameter Problem **(length: 36)** Type: **12** Code: **CODE**

Checksum: To Calculate
Pointer: **POINTER**
Unused: 0

Payload (length: 28)
Data: **Same as IPv4/UDP Packet**

Procedure:

0. Create a State in the NUT for TN1.
1. TN2 sends an IPv4/UDP Packet to TN1. The source address is TN2 IPv4 Address and the destination address is TN1 IPv4 Address given by The NUT.
2. TN1 replies with an IPv6/UDP Packet to TN2. The source address is TN1 IPv6 Global Address and the destination address is TN2 IPv6 Global NAT-PT Address.
3. Send "ICMPv4 Error Message" from TR to TN1 incrementing type and code. The inner part of these known packets contains an IPv4/UDP layer. The source address of the inner "IPv4/UDP" packet is TN1 IPv4 Address given by The NUT and the destination address is TN2 IPv4 Address. Go back to step 1.

Observable Results:

- **Step 1:** The NUT translates this packet in "IPv6/UDP" and sends it to TN1. In this new packet the UDP checksum has to be adjusted. The source address is TN2 IPv6 Global NAT-PT Address and the destination address is TN1 IPv6 Global Address.
- **Step 2:** The NUT translates this packet in "IPv4/UDP" and sends it to TN2. In this new packet the UDP checksum has to be adjusted. The source address is TN1 IPv4 Address given by The NUT and the destination address is TN2 IPv4 Address.
- **Step 3:** These Packets must be silently dropped except for cases indicated in the table presented hereafter. The source address of the inner "IPv6/UDP" layer is TN1 IPv6 Global Address and the destination address is TN2 IPv6 Global NAT-PT Address.

ICMPv4 Packet	ICMPv6 Corresponding Packet
Type = 3, Code = 0,1,5,6,7,8,11 or 12	Type = 1, Code = 0 (no route to destination)
Type = 3, Code = 2 (Protocol unreachable error)	Type = 4, Code = 1 (unrecognized Next Header type encountered) and make the Pointer point to the IPv6 Next Header field
Type = 3, Code = 3 (port unreachable)	Type = 1, Code = 4 (port unreachable)
Type = 3, Code = 4 (fragmentation needed and DF set)	Type = 2, Code = 0 (Too Big message) and the The MTU field needs to be adjusted
Type = 3, Code = 9, 10 (communication with destination host administratively prohibited)	Type = 1, Code = 1(communication with destination host administratively prohibited)
Type = 11 (Time Exceeded)	Type = 3 (Time Exceeded) and the code field is unchanged
Type = 12 (Parameter Problem)	Type = 4 and the pointer need to be adjusted to point to the corresponding field in the translated include IP header.

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner part of the translated Packet the UDP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/UDP	
TN1		1	<----------	TN2
	IPv6/UDP			
TN1	<----------	Step1		
	IPv6/UDP			
TN1	---------->	2		TN2
			IPv4/UDP	
		Step2	---------->	TN2
			ICMPv4 Error	
TN1		3	<----------	TR
	ICMPv6 Error (Some)			
TN1	<----------	Step3		TN2

5.3.4.3 ICMPv4 Error Messages with TCP packet in payload

Purpose:

Check translation of ICMPv4 Error Messages with a TCP packet in payload

References:

- **[RFC2766], *Section 5.3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

This test checks the correct translation of ICMPv4 Error Messages with a TCP packet included in payload. The ICMP checksum should be adjusted to account for the address change, going from V4 to V6 addresses. The inner IP header has also to be translated.

Packets:

- IPv6/TCP (length: 68 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 28 NextHeader: 6 HopLimit : (64) **[Same as in IPv4/TCP]** SourceAddress: **TN1 IPv6 Global Address or TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address or TN1 IPv6 Global Address**
TCP (length: 28) SourcePort: (1000) DestinationPort: (23) SequenceNumber: (0) AcknowledgmentNumber: (0) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (0) PSHFlag: (0) RSTFlag: (0) SYNFlag: 1 FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0) **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 (length: 116 bytes)

IPv6 Header (length: 40)

Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 76 NextHeader: 58 HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]** SourceAddress: **TR IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big (length: 76) Type: **TYPE** Code: **CODE** Checksum: To calculate Unused: 0 **Payload (length: 68)** Data: **Same as IPv6/TCP Packet**

- ICMPv6 Error Type 4 (length: 116 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 76 NextHeader: 58 HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]** SourceAddress: **TR IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
ICMPv6 Parameter Problem (length: 76) Type: **4** Code: **CODE** Checksum: To calculate Pointer: **POINTER** **Payload (length: 68)** Data: **Same as IPv6/TCP Packet**

- IPv4/TCP (length: 48 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 48 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 6

HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT or TN2 IPv4 Address** DestinationAddress: **TN2 IPv4 Address or TN1 IPv4 Address given by The NUT**
TCP (length: 28) SourcePort: (1000) DestinationPort: (23) SequenceNumber: (0) AcknowledgmentNumber: (0) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (0) PSHFlag: (0) RSTFlag: (0) SYNFlag: 1 FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0) **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv4 Error Type 3 (length: 76 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 76 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TR IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Destination Unreachable (length: 56) Type: **3** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 48)** Data: **Same as IPv4/TCP Packet**

- ICMPv4 Error Type 12 (length: 76 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5

TypeOfService: (0) TotalLength: 76 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TR IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
ICMPv4 Parameter Problem **(length: 56)** Type: **12** Code: **CODE** Checksum: To Calculate Pointer: **POINTER** Unused: 0 **Payload (length: 48)** Data: **Same as IPv4/TCP Packet**

Procedure:

0. Create a State in the NUT for TN1.
1. TN2 sends an IPv4/TCP Packet to TN1. The source address is TN2 IPv4 Address and the destination address is TN1 IPv4 Address given by The NUT. (**SYNFlag** is set, **S/N** is set to 1 and **A/N** is set to 0)
2. TN1 replies with an IPv6/TCP Packet to TN2. The source address is TN1 IPv6 Global Address and the destination address is TN2 IPv6 Global NAT-PT Address. (**SYNFlag** and **ACKFlag** are set, **S/N** is set to 10 and **A/N** is set to 2)
3. Send “ICMPv4 Error Message” from TR to TN1 incrementing type and code. The inner part of these known packets contains an IPv4/TCP layer. The source address of the inner “IPv4/TCP” packet is TN1 IPv4 Address given by The NUT and the destination address is TN2 IPv4 Address. Go back to step 1.

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv6/TCP” and sends it to TN1. In this new packet the TCP checksum has to be adjusted. The source address is TN2 IPv6 Global NAT-PT Address and the destination address is TN1 IPv6 Global Address.
- **Step 2:** The NUT translates this packet in “IPv4/TCP” and sends it to TN2. In this new packet the TCP checksum has to be adjusted. The source address is TN1 IPv4 Address given by The NUT and the destination address is TN2 IPv4 Address.
- **Step 3:** These Packets must be translated according to the table presented hereafter. The source address of the inner “IPv6/TCP” layer is TN1 IPv6 Global Address and the destination address is TN2 IPv6 Global NAT-PT Address.

ICMPv4 Packet	ICMPv6 Corresponding Packet
Type = 3, Code = 0,1,5,6,7,8,11 or 12	Type = 1, Code = 0 (no route to destination)
Type = 3, Code = 2 (Protocol unreachable error)	Type = 4, Code = 1 (unrecognized Next Header type encountered) and make the Pointer point to the IPv6 Next Header field
Type = 3, Code = 3 (port unreachable)	Type = 1, Code = 4 (port unreachable)
Type = 3, Code = 4 (fragmentation needed and DF set)	Type = 2, Code = 0 (Too Big message) and the The MTU field needs to be adjusted
Type = 3, Code = 9, 10 (communication with destination host administratively prohibited)	Type = 1, Code = 1(communication with destination host administratively prohibited)

Type = 12 (Parameter Problem)	Type=4 and the pointer need to be adjusted to point to the corresponding field in the translated include IP header.

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner part of the translated Packet the UDP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/TCP	
TN1		1	<----------	TN2
	IPv6/TCP			
TN1	<----------	Step1		
	IPv6/TCP			
TN1	---------->	2		TN2
			IPv4/TCP	
		Step2	---------->	TN2
			ICMPv4 Error	
TN1		3	<----------	TR
	ICMPv6 Error (Some)			
TN1	<----------	Step3		TN2

5.3.4.4 ICMPv4 Error Messages with IPv4 options in payload and header

Purpose:

Check translation of ICMPv4 Error Messages with IPv4 options included in payload and header.

References:

- **[RFC2765],** ***Section 3.1, 3.3***

  [RFC 2765], ***Page 15***

  "….

  The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers."

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

If IPv4 options are present in the IPv4 packet, they are ignored i.e., there is no attempt to translate them. However, if an unexpired source route option is present then the packet MUST instead be discarded, and an ICMPv4 "destination unreachable/source route failed" (Type 3/Code 5) error message SHOULD be returned to the sender.

This test checks the correct translation of ICMPv4 Error Messages with IPv4 Options included in payload and header. These Packets owes the following options: No Operation, Strict Source Route expired, Timestamp, Record Route, End of Option List. The inner part of these packets contains an IPv4/UDP layer with the same options (No Operation, Strict Source Route expired, Timestamp, Record Route, End of Option List). The ICMP checksum should be adjusted to account for the address change, going from V4 to V6 addresses. The inner IP header has also to be translated. These Packets must be translated according to the table presented in the following. Moreover, IPv4 Options MUST be ignored in the Header translation and in the Payload translation.

Packets:

- IPv6/UDP (length: 48 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 8 NextHeader: 17 HopLimit : (64) **[Same as in IPv4/UDP]** SourceAddress: **TN1 IPv6 Global Address or TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address or TN1 IPv6 Global Address**
UDP (length: 8) SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To calculate

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 (length: 96 bytes)

IPv6 Header (length: 40)

Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big (length: 56) Type: **TYPE** Code: **CODE** Checksum: To calculate Unused: 0 **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 4 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (63) **[one less than in the corresponding ICMPv4 Error]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
ICMPv6 Parameter Problem (length: 56) Type: **4** Code: **CODE** Checksum: To calculate Pointer: **POINTER** **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- IPv4/UDP(length: 60 bytes)

IPv4 Header (length: 52) Version: 4 IHL: 13 TypeOfService: (0) TotalLength: 60 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT or TN2**

IPv4 Address
DestinationAddress: **TN2 IPv4 Address or TN1 IPv4 Address given by The NUT**

NoOperation Option (length:1)
Type: 1

Strict Source Route Option (length:7)
Type: 137
Length: 7
Pointer:8
RouteData: **TN2 IPv4 Address**

Timestamp Option (length:8)
Type: 68
Length: 8
Pointer:5
Overflow: 0
Flag: 0
Timestamp: 170

NoOperation Option (length:1)
Type: 1

Record Route Option (length:11)
Type: 7
Length: 11
Pointer:4
RouteData: **0.0.0.0**
RouteData: **0.0.0.0**

EndofOptionList (length:1)
Type: 0

Padding = 000000

UDP (length: 8)

SourcePort: (1000)
DestinationPort: (1000)
Length: 8
Checksum: To Calculate

- ICMPv4 Error Type 3 (length: 88 bytes)

IPv4 Header (length: 52)

Version: 4
IHL: 13
TypeOfService: (0)
TotalLength: 88
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 1
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

NoOperation Option (length:1)
Type: 1

Strict Source Route Option (length:7)
Type: 137

Length: 7
Pointer:8
RouteData: **TN2 IPv4 Address**

Timestamp Option (length:8)
Type: 68
Length: 8
Pointer:5
Overflow: 0
Flag: 0
Timestamp: 170

NoOperation Option (length:1)
Type: 1

Record Route Option (length:11)
Type: 7
Length: 11
Pointer:4
RouteData: **0.0.0.0**
RouteData: **0.0.0.0**

EndofOptionList (length:1)
Type: 0

Padding = 000000

ICMPv4 Destination Unreachable (length: 36)

Type: **3**
Code: **CODE**
Checksum: To Calculate
Unused: 0

Payload (length: 60)
Data: **Same as IPv4/UDP Packet**

- ICMPv4 Error Type 12 (length: 88 bytes)

IPv4 Header (length: 52)

Version: 4
IHL: 13
TypeOfService: (0)
TotalLength: 88
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 1
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

NoOperation Option (length:1)
Type: 1

Strict Source Route Option (length:7)
Type: 137
Length: 7
Pointer:8
RouteData: **TN2 IPv4 Address**

Timestamp Option (length:8)

Type: 68 Length: 8 Pointer:5 Overflow: 0 Flag: 0 Timestamp: 170 **NoOperation Option (length:1)** Type: 1 **Record Route Option (length:11)** Type: 7 Length: 11 Pointer:4 RouteData: **0.0.0.0** RouteData: **0.0.0.0** **EndofOptionList (length:1)** Type: 0 **Padding = 000000**
ICMPv4 Parameter Problem **(length: 36)** Type: **12** Code: **CODE** Checksum: To Calculate Pointer: **POINTER** Unused: 0 **Payload (length: 60)** Data: **Same as IPv4/UDP Packet**

Procedure:

0. Create a State in the NUT for TN1.
1. TN2 sends an IPv4/UDP Packet to TN1. The source address is TN2 IPv4 Address and the destination address is TN1 IPv4 Address given by The NUT.
2. TN1 replies with an IPv6/UDP Packet to TN2. The source address is TN1 IPv6 Global Address and the destination address is TN2 IPv6 Global NAT-PT Address.
3. Send “ICMPv4 Error Message” described in the following from TN2 to TN1. The inner part of these packets contains an IPv4/UDP layer. Moreover, Options (No Operation, Strict Source Route expired, Timestamp, Record Route, End of Option List) are present in the IPv4 Header and in the ICMPv4 Header of the ICMPv4 Error Packet. The source address of the inner “IPv4/UDP” packet is TN1 IPv4 Address given by The NUT and the destination address is TN2 IPv4 Address. Go back to step 1.

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv6/UDP” and sends it to TN1. In this new packet the UDP checksum has to be adjusted. The source address is TN2 IPv6 Global NAT-PT Address and the destination address is TN1 IPv6 Global Address.
- **Step 2:** The NUT translates this packet in “IPv4/UDP” and sends it to TN2. In this new packet the UDP checksum has to be adjusted. The source address is TN1 IPv4 Address given by The NUT and the destination address is TN2 IPv4 Address.
- **Step 3:** These Packets must be translated according to the table presented hereafter. The source address of the inner “IPv6/UDP” layer is TN1 IPv6 Global Address and the destination address is TN2 IPv6 Global NAT-PT Address.

ICMPv4 Packet	**ICMPv6 Corresponding Packet**
Type = 3, Code = 0,1,5,6,7,8,11 or 12	Type = 1, Code = 0 (no route to destination)
Type = 3, Code = 2 (Protocol unreachable error)	Type = 4, Code = 1 (unrecognized Next Header type encountered) and make the Pointer point to the IPv6 Next Header field

Type = 3, Code = 3 (port unreachable)	Type = 1, Code = 4 (port unreachable)
Type = 3, Code = 4 (fragmentation needed and DF set)	Type = 2, Code = 0 (Too Big message) and the The MTU field needs to be adjusted
Type = 3, Code = 9, 10 (communication with destination host administratively prohibited)	Type = 1, Code = 1(communication with destination host administratively prohibited)
Type = 12 (Parameter Problem)	Type=4 and the pointer need to be adjusted to point to the corresponding field in the translated include IP header.

When Packet are translated, the options present in the IPv4 Header and in the ICMPv4 Header of the ICMPv4 Error Packet are removed. Moreover, the translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner Part of the translated Packet the UDP checksum has to be adjusted and the IPv4 Options have been removed.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/UDP	
TN1		1	<----------	TN2
	IPv6/UDP			
TN1	<----------	Step1		
	IPv6/UDP			
TN1	---------->	2		TN2
			IPv4/UDP	
		Step2	---------->	TN2
			ICMPv4 Error	
TN1		3	<----------	TR
	ICMPv6 Error (Some)			
TN1	<----------	Step3		TN2

5.3.5 ICMPv6 Translation

An ICMPv6 Error Packet has sometimes an IPv6 Packet in its payload. In this part we will consider that the translation of the inner IP header has to be done. Indeed, what is the need to translate an ICMPv6 error message if the inner part is not also translated?

The ICMPv6 message format is specified by the value of the Type field:

1	Destination unreachable.
2	Packet too big.
3	Time exceeded.
4	Parameter problem.
128	Echo request.
129	Echo reply.
130	Group Membership Query.
131	Group Membership Report.
132	Group Membership Reduction.
133	Router Solicitation.
134	Router Advertisement.
135	Neighbor Solicitation.
136	Neighbor Advertisement.
137	Redirect.
138	Router Renumbering.
139	ICMP Node Information Query.
140	ICMP Node Information Response.
141	Inverse Neighbor Discovery Solicitation Message.
142	Inverse Neighbor Discovery Advertisement Message.
143	Home Agent Address Discovery Request Message.
144	Home Agent Address Discovery Reply Message.
145	Mobile Prefix Solicitation.
146	Mobile Prefix Advertisement.

5.3.5.1 ICMPv6 Error Messages with UDP packet in payload

Purpose:

Check Translation of ICMPv6 Error Messages when a UDP packet is included in payload. ICMPv6 error messages are ICMPv6 messages with a type field between 0 and 254.

References:

- **[RFC2765]** ***Section 4.2, 4.3***

  [RFC 2765] Page 15 :

  "….

  The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers."

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1 and TN3

Discussion:

This test checks the correct translation of ICMPv6 Error Messages with a UDP packet included in payload. The ICMP checksum should be adjusted to account for the address change, going from V6 to V4 addresses. The inner IP header has also to be translated.

Packets:

- IPv6/UDP (length: 48 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 8 NextHeader: 17 HopLimit : (64) SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN3 IPv6 Global Address**
UDP (length: 8) SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To Calculate

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**

ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big (length: 56) Type: **TYPE** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 3 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Time Exceeded (length: 56) Type: **3** Code: **CODE** Checksum: To Calculate MTU: (1280) Unused: 0 **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 4 (length: 96 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 56 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Parameter Problem (length: 56) Type: **4** Code: **CODE** Checksum: To Calculate Pointer: **POINTER** **Payload (length: 48)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Unknown Error Message (length: 44 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 4 NextHeader: 58 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 ANY (length: 4)
Type: **TYPE** Code: **CODE** Checksum: To Calculate Data: **NONE**

- IPv4/UDP(length: 28 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 28 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) **[Same as in IPv6/UDP]** Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN3 IPv4 Address given by The NUT**
UDP (length: 8)
SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To Calculate

- ICMPv4 Error Type 3 / ICMPv4 Error Type 11 (length: 36 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT**

DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Destination Unreachable / ICMPv4 Time Exceeded (length: 16) Type: **TYPE** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 8)** Data: **First 8 bytes of IPv4/UDP Packet**

- ICMPv4 Error Type 12 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Parameter Problem (length: 16) Type: 12 Code: **CODE** Checksum: To Calculate Pointer: **POINTER** **Payload (length: 8)** Data: **First 8 bytes of IPv4/UDP Packet**

Procedure:

0. Create a State in the NUT for TN1 and TN3
1. TN2 sends an “IPv4/UDP” Packet to TN3
2. Send “ICMPv6 Error Message” from TN1 to TN2 incrementing type and code. The inner part of these packets contains an IPv6/UDP layer. Go back to step 1.

Observable Results:

- **Step 1:** The NUT must translate this packet in “IPv6/UDP” and forward it to TN1.
- **Step 2:** These Packets must be silently dropped except for the cases indicated in the table presented hereafter:

ICMPv6 Packet	ICMPv4 Corresponding Packet
Type = 1, Code = 0 (no route to destination)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 1 (communication with destination administratively prohibited)	Type = 3, Code = 10 (communication with destination host administratively prohibited)
Type = 1, Code = 2 (beyond scope of source address)	Type = 3, Code = 1 (host unreachable)

Type = 1, Code = 3 (address unreachable)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 4 (port unreachable)	Type = 3, Code = 3 (port unreachable)
Type = 2 (Packet Too Big)	Type = 3, Code = 4 And ajust the MTU Field
Type = 3 (Time Exceeded)	Type = 11, Code = unchanged
Type = 4, Code = 1	Type = 3, Code = 2 (protocol unreachable)
Type = 4, Code <> 1	Type = 12, Code = 0 and Update the Pointer (if Pointer was 0 -> 0 if Pointer was 4 -> 2 if Pointer was 6 -> 9 if Pointer was 7 -> 8 if Pointer was 8 -> 12 if Pointer was 24 -> 16)

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner part of the translated Packet the UDP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/UDP	
TN3		1	<----------	TN2
	IPv6/UDP			
TN3 (Send to TN1)	<----------	Step1		
	ICMPv6 Error			
TN1	---------->	2		TN2
			ICMPv4 Error (Some)	
		Step2	---------->	TN2

5.3.5.2 ICMPv6 Error Messages with TCP packet in payload

Purpose:

Check Translation of ICMPv6 Error Messages when a TCP packet is included in payload. ICMPv6 error messages are ICMPv6 messages with a type field between 0 and 254.

References:

- **[RFC2766] Section 5.3**

  [RFC2765] Page 21

  " …

  ICMPv6 error messages:

  Destination Unreachable (Type 1)

  Set the Type field to 3. Translate the code field as follows:

  Code 0 (no route to destination): Set Code to 1 (host unreachable)."

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

NONE

Discussion:

This test checks the correct translation of ICMPv6 Error Messages with a TCP packet included in payload. The ICMP checksum should be adjusted to account for the address change, going from V6 to V4 addresses. The inner IP header has also to be translated.

Packets:

- IPv6/TCP (length: 68 bytes)

IPv6 Header (length: 40)
Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 28 NextHeader: 6 HopLimit : (64) SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN3 IPv6 Global Address**
TCP (length: 28)
SourcePort: (1000) DestinationPort: (23) SequenceNumber: **S/N** AcknowledgmentNumber: **A/N** DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: **ACKFlag** PSHFlag: (0) RSTFlag: (0) SYNFlag: **SYNFlag** FINFlag: **FINFlag** Window: (0) Checksum: To calculate UrgentPointer: (0)

Payload (length: 8)
Data: (01234567 89abcdef)

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 (length: 116 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 76
NextHeader: 58
HopLimit : (64)
SourceAddress: **TN1 IPv6 Global Address**
DestinationAddress: **TN2 IPv6 Global NAT-PT Address**

ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big (length: 76)

Type: **TYPE**
Code: **CODE**
Checksum: To calculate
Unused: 0

Payload (length: 68)
Data: **Same as IPv6/TCP Packet**

- ICMPv6 Error Type 3 (length: 116 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 76
NextHeader: 58
HopLimit : (64)
SourceAddress: **TN1 IPv6 Global Address**
DestinationAddress: **TN2 IPv6 Global NAT-PT Address**

ICMPv6 Time Exceeded (length: 76)

Type: **3**
Code: **CODE**
Checksum: To calculate
MTU: (1280)
Unused: 0

Payload (length: 68)
Data: **Same as IPv6/TCP Packet**

- ICMPv6 Error Type 4 (length: 116 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 76
NextHeader: 58
HopLimit : (64)

SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv6 Parameter Problem (length: 76) Type: **3** Code: **CODE** Checksum: To calculate Pointer: **POINTER** **Payload (length: 48)** Data: **Same as IPv6/TCP Packet**

- IPv4/TCP (length: 48 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 48 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) **[same as in IPv6/TCP]** Protocol: 6 HeaderChecksum: To calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN3 IPv4 Address given by The NUT**
TCP (length: 28) SourcePort: (1000) DestinationPort: (23) SequenceNumber: **S/N** AcknowledgmentNumber: **A/N** DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: **ACKFlag** PSHFlag: (0) RSTFlag: (0) SYNFlag: **SYNFlag** FINFlag: **FINFlag** Window: (0) Checksum: To calculate UrgentPointer: (0) **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv4 Error Type 3 / ICMPv4 Error Type 11 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0

Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Destination Unreachable / ICMPv4 Time Exceeded (length: 16) Type: **TYPE** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 8)** Data: **First 8 bytes of IPv4/TCP Packet**

- ICMPv4 Error Type 12 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Parameter Problem (length: 16) Type: 12 Code: **CODE** Checksum: To Calculate Pointer: **POINTER** **Payload (length: 8)** Data: **First 8 bytes of IPv4/TCP Packet**

Procedure:

0. Create a State in the NUT for TN1 and TN3
1. TN2 sends an “IPv4/TCP” Packet to TN3
2. Send “ICMPv6 Error Message” from TN1 to TN2 incrementing type and code. The inner part of these packets contains an IPv6/TCP layer. Go back to step 1.

Observable Results:

- **Step 1:** The NUT must translate this packet in “IPv6/TCP” and forward it to TN1.
- **Step 2:** These Packets must be translated according to the table presented hereafter:

ICMPv6 Packet	ICMPv4 Corresponding Packet

Type = 1, Code = 0 (no route to destination)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 1 (communication with destination administratively prohibited)	Type = 3, Code = 10 (communication with destination host administratively prohibited)
Type = 1, Code = 2 (beyond scope of source address)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 3 (address unreachable)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 4 (port unreachable)	Type = 3, Code = 3 (port unreachable)
Type = 2 (Packet Too Big)	Type = 3, Code = 4 And ajust the MTU Field
Type = 3 (Time Exceeded)	Type = 11, Code = unchanged
Type = 4, Code = 1	Type = 3, Code = 2 (protocol unreachable)
Type = 4, Code <> 1	Type = 12, Code = 0 and Update the Pointer (if Pointer was 0 -> 0 if Pointer was 4 -> 2 if Pointer was 6 -> 9 if Pointer was 7 -> 8 if Pointer was 8 -> 12 if Pointer was 24 -> 16)

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner part of the translated Packet, the TCP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/TCP	
TN3		1	<----------	TN2
	IPv6/TCP			
TN3 (Send to TN1)	<----------	Step1		
	ICMPv6 Error			
TN1	---------->	2		TN2
			ICMPv4 Error (Some)	
		Step2	---------->	TN2

5.3.5.3 ICMPv6 Error Messages with IPv6 options in payload and header

Purpose:

Check Translation of ICMPv6 Error Messages with IPv6 options included in payload and header. ICMPv6 error messages are ICMPv6 messages with a type field between 0 and 254.

References:

- **[RFC2765]** ***Section 4.2, 4.3***
- **[RFC 2765]** ***Page 15***

  "….

  The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers."
- **[RFC 2765]** ***Section 4.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1 and TN3

Discussion:

This test checks the correct translation of ICMPv6 Error Messages with IPv6 Options included in payload and header. These Packets owes the following extension headers: Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header. The inner part of these known packets contains an IPv6/UDP layer with the same options (Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header). The ICMP checksum should be adjusted to account for the address change, going from V6 to V4 addresses. The inner IP header has also to be translated. These Packets must be translated according to the table presented in the following. Moreover, IPv6 Options MUST be ignored in the Header translation and in the Payload translation.

Packets:

- IPv6/UDP (with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header) (length: 88 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 48 NextHeader: 0 HopLimit : (64) SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN3 IPv6 Global Address**
Hop by Hop Option Header (length: 8) NextHeader = 60 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Destination Option Header (length: 8) NextHeader = 43

HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Routing Header (length: 8) NextHeader = 60 HeaderExtLength = 0 RoutingType = 0 SegmentsLeft = 0 Type-specificData = 0
Destination Option Header (length: 8) NextHeader = 17 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- ICMPv6 Error Type 1 / ICMPv6 Error Type 2 (with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header) (length: 168 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 128 NextHeader: 0 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Hop by Hop Option Header (length: 8) NextHeader = 60 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Destination Option Header (length: 8) NextHeader = 43 HeaderExtLength = 0

Opt_PadN (length:6) OptionType = 1 OptDataLength = 4 Pad = 00000000
Routing Header (length: 8) NextHeader = 60 HeaderExtLength = 0 RoutingType = 0 SegmentsLeft = 0 Type-specificData = 0
Destination Option Header (length: 8) NextHeader = 58 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
ICMPv6 Destination Unreachable / ICMPv6 Packet Too Big (length: 96) Type: **TYPE** Code: **CODE** Checksum: To calculate Unused: 0 **Payload (length: 88)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 3 (with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header) (length: 168 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 128 NextHeader: 0 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Hop by Hop Option Header (length: 8) NextHeader = 60 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Destination Option Header (length: 8) NextHeader = 43 HeaderExtLength = 0

Opt_PadN (length:6) OptionType = 1 OptDataLength = 4 Pad = 00000000
Routing Header (length: 8) NextHeader = 60 HeaderExtLength = 0 RoutingType = 0 SegmentsLeft = 0 Type-specificData = 0
Destination Option Header (length: 8) NextHeader = 58 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
ICMPv6 Time Exceeded (length: 96) Type: **3** Code: **CODE** Checksum: To calculate MTU: (1280) Unused: 0 **Payload (length: 88)** Data: **Same as IPv6/UDP Packet**

- ICMPv6 Error Type 4 (with Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header) (length: 168 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 128 NextHeader: 0 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
Hop by Hop Option Header (length: 8) NextHeader = 60 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
Destination Option Header (length: 8) NextHeader = 43 HeaderExtLength = 0

Opt_PadN (length:6) OptionType = 1 OptDataLength = 4 Pad = 00000000
Routing Header (length: 8) NextHeader = 60 HeaderExtLength = 0 RoutingType = 0 SegmentsLeft = 0 Type-specificData = 0
Destination Option Header (length: 8) NextHeader = 58 HeaderExtLength = 0 **Opt_PadN (length:6)** OptionType = 1 OptDataLength = 4 Pad = 00000000
ICMPv6 Parameter Problem (length: 96) Type: **3** Code: **CODE** Checksum: To calculate Pointer: **POINTER** **Payload (length: 88)** Data: **Same as IPv6/UDP Packet**

- IPv4/UDP(length: 28 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 28 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) **[Same as in IPv6/UDP]** Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN3 IPv4 Address given by The NUT**
UDP (length: 8) SourcePort: (1000) DestinationPort: (1000) Length: 8 Checksum: To Calculate

- ICMPv4 Error Type 3 / ICMPv4 Error Type 11 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Destination Unreachable / ICMPv4 Time Exceeded (length: 16) Type: **TYPE** Code: **CODE** Checksum: To Calculate Unused: 0 **Payload (length: 8)** Data: **First 8 bytes of IPv4/UDP Packet**

- ICMPv4 Error Type 12 (length: 36 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: 0 Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the corresponding ICMPv6 Error]** Protocol: 1 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
ICMPv4 Parameter Problem (length: 16) Type: 12 Code: **CODE** Checksum: To Calculate Pointer: **POINTER** **Payload (length: 8)** Data: **First 8 bytes of IPv4/UDP Packet**

Procedure:

0. Create a State in the NUT for TN1 and TN3
1. TN2 sends an “IPv4/UDP” Packet to TN3

2. Send “ICMPv6 Error Message” described in the following from TN1 to TN2. These Packets owes the following extension headers: Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header. The inner part of these known packets contains an IPv6/UDP layer with the same options (Hop by Hop Extension, Destination Option Header, Routing Header Type 0 with Segments Left = 0, Destination Option Header). Go back to step 1.

Observable Results:

- **Step 1:** The NUT must translate this packet in “IPv6/UDP” and forward it to TN1.
- **Step 2:** These Packets must be translated according to the table presented hereafter. Moreover, IPv6 Options MUST be ignored in the Header translation and in the Payload translation.

ICMPv6 Packet	ICMPv4 Corresponding Packet
Type = 1, Code = 0 (no route to destination)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 1 (communication with destination administratively prohibited)	Type = 3, Code = 10 (communication with destination host administratively prohibited)
Type = 1, Code = 2 (beyond scope of source address)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 3 (address unreachable)	Type = 3, Code = 1 (host unreachable)
Type = 1, Code = 4 (port unreachable)	Type = 3, Code = 3 (port unreachable)
Type = 2 (Packet Too Big)	Type = 3, Code = 4 And ajust the MTU Field
Type = 3 (Time Exceeded)	Type = 11, Code = unchanged
Type = 4, Code = 1	Type = 3, Code = 2 (protocol unreachable)
Type = 4, Code <> 1	Type = 12, Code = 0 and Update the Pointer (if Pointer was 0 -> 0 if Pointer was 4 -> 2 if Pointer was 6 -> 9 if Pointer was 7 -> 8 if Pointer was 8 -> 12 if Pointer was 24 -> 16)

The translation of the inner IP header can be done by recursively invoking the function that translated the outer IP headers. Thus, in the inner part of the translated Packet, the TCP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
TN3		1	IPv4/UDP <----------	TN2
TN3 (Send to TN1)	IPv6/UDP <----------	Step1		
TN1	ICMPv6 Error ---------->	2		TN2
		Step2	ICMPv4 Error (Some) ---------->	TN2

5.3.6 ICMPv4 Error Generation

5.3.6.1 TTL set to 0 or 1 & ICMPv4 Time Exceeded Message

Purpose:

Check that the NUT discards packets it receives with TT set to 0 or 1. In this case the NUT MUST send an ICMPv4 TTL exceeded packet.

References:

- **[RFC2765],** ***Section 3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

This test checks that packet is correctly discarded when TTL is set to 0. Moreover, the NUT SHOULD send an ICMPv4 "ttl exceeded" error.

Packets:

- IPv4/UDP (with TTL = 1 or 0) (length: 36 bytes)

IPv4 Header (length: 28)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 36
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: **0 or 1**
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv6 Global NAT-PT Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

UDP (length: 16)

SourcePort: (1000)
DestinationPort: (1000)
Length: 16
Checksum: To Calculate

Payload (length: 8)
Data: (01234567 89abcdef)

- ICMPv4 Time Exceeded (length: 36 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: 0

TotalLength: 36 Identifier: XXXXX Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: XXXXX Protocol: 41 HeaderChecksum: XXXXX SourceAddress: **NUT IPv4 Address Link 2** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
ICMPv4 Time Exceeded (length: 16) Type: 11 Code: 0 Checksum: To Calculate Unused: 0 **Payload (length: 28)** Data: **the IPv4 Header of the invoking packet + the First 8 bytes of the corresponding IPv4/UDP**

Procedure:

0. Create a State in the NUT for TN1
1. TR forwards an “IPv4/UDP” Packet from TN2 to TN1 with TTL = 0.
2. TR forwards an “IPv4/UDP” Packet from TN2 to TN1 with TTL = 1.

Observable Results:

- **Step 1:** The NUT must send an “ICMPv4 Time Exceeded” Code 0 Packet to TN2.
- **Step 2:** The NUT must send an “ICMPv4 Time Exceeded” Code 0 Packet to TN2. In the Payload, the TTL of the offending packet is set to 0 or 1.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/UDP **(TTL == 0)**	
TN1		1	<----------	TN2
			ICMPv4 Time Exceeded	
		Step1	---------->	TN2
			IPv4/UDP **(TTL == 1)**	
TN1		2	<----------	TN2
			ICMPv4 Time Exceeded	
		Step2	---------->	TN2

5.3.7 MTU Handling & Fragmentation

5.3.7.1 Translation of Fragmented packets

Purpose:

Check that IPv4 fragmented packets are correctly translated

References:

- **[RFC2765]** ***Section 3.1***

  "If there is need to add a fragment header (the DF bit is not set or the packet is a fragment) the header fields are set as above with the following exceptions:

  IPv6 fields:

  - Payload Length: Total length value from IPv4 header, plus 8 for the fragment header, minus the size of the IPv4 header and IPv4 options, if present.
  - Next Header: Fragment Header (44)

  Fragment header fields:

  - Next Header: Protocol field copied from IPv4 header.
  - Fragment Offset: Fragment Offset copied from the IPv4 header.
  - M flag: More Fragments bit copied from the IPv4 header.
  - Identification: The low-order 16 bits copied from the Identification field in the IPv4 header. The high-order 16 bits set to zero."

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

Check that IPv4 fragmented packets are correctly translated

Packets:

- IPv4/UDP (1st Fragment) (length: 36 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 36 Identifier: (0xFFEFFF03) Reserved: (0) DF: 1 MF: 1 FragmentOffset: 0 TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
UDP (length: 16)
SourcePort: (1000) DestinationPort: (1000)

Length: 16 Checksum: To Calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv4/UDP (2nd Fragment) (length: 28 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 28 Identifier: (0xFFEFFF03) Reserved: (0) DF: 1 MF: 0 FragmentOffset: 2 (in 8-octet units) TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
Payload (length: 8) Data: (01234567 89abcdef)

- IPv6/UDP (1st Fragment) (length: 64 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 24 NextHeader: 44 HopLimit : (63) **[one less than in the corresponding IPv4/UDP (1st Fragment)]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
Fragment Header (length:8) NextHeader: 17 Reserved1: 0 FragmentOffset: 0 Reserved2: 0 Mflag: 1 Identification: (0xFF03)
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To Calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv6/UDP (2nd Fragment) (length: 56 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 16 NextHeader: 44 HopLimit : (63) **[one less than in the corresponding IPv4/UDP (2nd Fragment)]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
Fragment Header (length:8) NextHeader: 17 Reserved1: 0 FragmentOffset: 2 (in 8-octet units) Reserved2: 0 Mflag: 1 Identification: (0xFF03)
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To Calculate **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

0. Create a State in the NUT for TN1
1. TN2 sends an "IPv4/UDP (1st Fragment)" Packet to TN1 through The NUT
2. TN2 sends an "IPv4/UDP (2nd Fragment)" Packet to TN1 through The NUT

Observable Results:

- **Step 1:** The NUT translates this packet in "IPv6/UDP (1st Fragment)" and sends it to TN1. In this new packet the UDP checksum has to be adjusted.
- **Step 2:** The NUT translates this packet in "IPv6/UDP (2st Fragment)" and sends it to TN1.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/UDP **(1st Fragment)**	
TN1		1	<----------	TN2
	IPv6/UDP **(1st Fragment)**			
TN1	<----------	Step1		
			IPv4/UDP **(2nd Fragment)**	
TN1		2	<----------	TN2
	IPv6/UDP **(2nd Fragment)**			
TN1	<----------	Step2		

5.3.7.2 IPv4/UDP Fragmented Packet without UDP checksum

Purpose:

Check translation of IPv4/UDP Fragmented Packet with UDP checksum set to 0.

References:

- **[RFC2765],** ***Section 3.2***
- **[RFC2766],** ***Section 5.3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

When the checksum of a V4 UDP packet is set to zero, NAT-PT MUST evaluate the checksum in its entirety for the V6-translated UDP packet. If a V4 UDP packet with a checksum of zero arrives in fragments, NAT-PT MUST await all the fragments until they can be assembled into a single non-fragmented packet and evaluate the checksum prior to forwarding the translated V6 UDP packet.

Packets:

- IPv4/UDP 1 (1st Fragment) (length: 540 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 540
Identifier: (0xFFEFFF04)
Reserved: (0)
DF: 1
MF: 1
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

UDP (length: 520)

SourcePort: (1000)
DestinationPort: (1000)
Length: 536
Checksum: **0**

Payload (length: 512)
512 bytes of Data

- IPv4/UDP 2 (2nd Fragment) (length: 28 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)

TotalLength: 28 Identifier: (0xFFEFFF04) Reserved: (0) DF: 1 MF: 1 FragmentOffset: 520 (in 8-octet units) TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
Payload (length: 8) 8 bytes of Data

- IPv4/UDP 3 (Last Fragment) (length: 28 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 28 Identifier: (0xFFEFFF04) Reserved: (0) DF: 1 MF: 0 FragmentOffset: 528 (in 8-octet units) TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
Payload (length: 8) 8 bytes of Data

- IPv6/UDP (length: 584 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 544 NextHeader: 44 HopLimit : (63) **[one less than in the corresponding IPv4/UDP (1st Fragment)]** SourceAddress: **TN2 Global IPv6 NAT-PT Address** DestinationAddress: **TN1 Global IPv6 Address**
Fragment Header (length:8) NextHeader: 17 Reserved1: 0 FragmentOffset: 0 Reserved2: 0 Mflag: 0 Identification: (0xFF04)
UDP (length: 536)

SourcePort: (1000)
DestinationPort: (1000)
Length: 536
Checksum: Calculated UDP Checksum

Payload (length: 528)
Data: **The 528 bytes of data of IPv4/UDP 1,2 & 3**

Procedure:

0. Create a State in the NUT for TN1
1. TN2 Sends "IPv4/UDP 1 (1st Fragment)", "IPv4/UDP 2 (2nd Fragment)", "IPv4/UDP 3 (last Fragment)" to TN1. The UDP Checksum is set to 0 in "IPv4/UDP 1".

Observable Results:

- **Step 1:** NAT-PT MUST await all the fragments until they can be assembled into a single non-fragmented packet and evaluate the checksum prior to forwarding the translated "IPv6/UDP" packet to TN1. In this packet, the UDP checksum has to be calculated.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/UDP **(1st Fragment)**	
TN1			<----------	TN2
			IPv4/UDP **(2nd Fragment)**	
TN1			<----------	TN2
			IPv4/UDP **(Last Fragment)**	
TN1		1	<----------	TN2
	IPv6/UDP			
TN1	<----------	Step1		

5.3.7.3 DF set to 0 and Fragmentation of IPv6 packets

Purpose:

Check that IPv4 packet with DF not set is correctly translated. The IPv6 resulting packet must not exceeds 1280 bytes. If the resulting packet exceeds 1280 bytes, it has to be fragmented prior to be forwarded. Even when this packet is not larger than 1280 bytes, a fragment header is added to the IPv6 translated Packet.

References:

- **[RFC2765],** ***Section 3.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

This test checks that IPv4 packet with DF not set is correctly translated. The IPv6 resulting packet must not exceed the IPv6 Minimum MTU, 1280 bytes. If the resulting packet exceeds 1280 bytes, it has to be fragmented prior to be forwarded. Even when this packet is not larger than 1280 bytes, a fragment header is added to the IPv6 translated Packet.

Packets

- IPv4/UDP 1 (It will result in an IPv6 Packet larger than 1280 bytes) (length: 1436 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: (0) TotalLength: 1436 Identifier: (0xFFEFFF05) Reserved: 0 DF: 0 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
UDP (length: 1416) SourcePort: (1000) DestinationPort: (1000) Length: 1416 Checksum: To Calculate **Payload (length: 8)** Data: **1408 Bytes of data**

- IPv4/UDP 2 (length: 36 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: (0)

TotalLength: 36 Identifier: (0xFFEFFF06) Reserved: 0 DF: 0 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 17 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To Calculate **Payload (length: 8)** Data: (01234567 89abcdef)

- IPv6/UDP 1 (1st Fragment) (length: 1280 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 1240 NextHeader: 44 HopLimit : (63) **[one less than in the IPv4/UDP 1]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
Fragment Header (length:8) NextHeader: 17 Reserved1: 0 FragmentOffset: 0 Reserved2: 0 Mflag: 1 Identification: (0xFF05)
UDP (length: 1232) SourcePort: (1000) DestinationPort: (1000) Length: 1416 Checksum: To Calculate **Payload (length: 1224)** Data: **the first 1224 Bytes of data of IPv4/UDP 1**

- IPv6/UDP 2 (2nd Fragment) (length: 224 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 184

NextHeader: 44 HopLimit : (63) **[one less than in the IPv4/UDP 1]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
Fragment Header (length:8) NextHeader: 17 Reserved1: 0 FragmentOffset: 154 Reserved2: 0 Mflag: 0 Identification: (0xFF05)
Payload (length: 176) Data: **the last 176 Bytes of data of IPv4/UDP 1**

- IPv6/UDP 3 (with Fragment Header) (length: 64 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 24 NextHeader: 44 HopLimit : (63) **[one less than in the IPv4/UDP 2]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
Fragment Header (length:8) NextHeader: 17 Reserved1: 0 FragmentOffset: 0 Reserved2: 0 Mflag: 0 Identification: (0xFF06)
UDP (length: 16) SourcePort: (1000) DestinationPort: (1000) Length: 16 Checksum: To Calculate **Payload (length: 8)** Data: (01234567 89abcdef)

Procedure:

0. Create a State in the NUT for TN1
1. TN2 sends “IPv4/UDP 1” (It will result in an IPv6 Packet larger than 1280 bytes) with DF=0 to TN1
2. TN2 sends “IPv4/UDP 2” with DF=0 to TN1

Observable Results:

- **Step 1:** The NUT translates this packet in “IPv6/UDP 1 (1st Fragment)”, “IPv6/UDP 2 (2nd Fragment)” and sends it to TN1. In the packet “IPv6/UDP 1”, the UDP checksum has to be adjusted.
- **Step 2:** The NUT translates this packet in “IPv6/UDP 3” with a Fragment Header and sends it to TN1. In this resulting packet, the UDP checksum has to be adjusted.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
			IPv4/UDP 1 **(DF == 0)**	
TN1		1	<----------	TN2
	IPv6/UDP 1 (1st Fragment)			
TN1	<----------			
	IPv6/UDP 2 (2nd Fragment)	Step1		
TN1	<----------			
			IPv4/UDP 2 **(DF == 0)**	
TN1		2	<----------	TN2
	IPv6/UDP 3			
TN1	<----------	Step2		

5.3.8 FTP-ALG

The use of IP addresses and TCP Ports in FTP Packets with commands such as PORT, PASV, EPRT, EPSV, involve the need of having an FTP-ALG located on the NUT to facilitate transparent FTP between v4 and v6 nodes. This part give some conformance tests to check the correct translation of FTP packets from a v4 (TN2) to a v6 (TN1) node.

Each test will begin with the establishment of a passive TCP connection to the FTP server TN1 using port 21.

This is not really mandatory but because each packet going into the v6 network has to go through the NUT, this can lead to some problems. Indeed the NUT can keep the status of live TCP connection and by this way, rejects all TCP packets when no state is available in the router. As a consequence, each test will have to end with the closing of this TCP connection.

5.3.8.1 PORT

Purpose:

Check the correct translation of PORT command into EPRT command

References:

- **[RFC2766]** ***Section 6.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- A FTP-ALG MUST be available on the NUT

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

According to **[RFC2428]**, the FTP commands PORT and PASV are replaced with EPRT and EPSV. Nevertheless because a v4 node may not have the EPRT and EPSV commands implemented, it may originates an FTP session using PORT or PASV command.

Thus, in active mode FTP between TN2 and TN1 and from the IPv4 side, the client (TN2) connects from a random unprivileged port (N > 1024) to the FTP server's command port 21. Then, the client starts listening to port N+1 and sends the FTP command PORT N+1 to the FTP server. The server will then connect back to the client's specified data port from its local data port, which is port 20.

The FTP-ALG has to translate this PORT command into EPRT command before to forward it to the v6 node TN1.

Packets:

- IPv4/TCP/FTP PORT (length: 67 bytes)

IPv4 Header (length: 20)
Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 67 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (64) Protocol: 6 HeaderChecksum: To Calculate SourceAddress: **TN2 IPv4 Address** DestinationAddress: **TN1 IPv4 Address given by The NUT**
TCP (length: 20)

SourcePort: (49226) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (65535) Checksum: To calculate UrgentPointer: (0)
FTP – PORT Command (length: 27) **Request: TN2_FTP_PORT_v4**

- IPv6/TCP/FTP EPRT(length: 105 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 65 NextHeader: 6 HopLimit : (63) **[one less than in the IPv4/TCP/FTP PORT]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
TCP (length: 20) SourcePort: (49226) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0)
FTP EPRT Command (length: 45) **Request: TN2_FTP_EPRT_v6**

TN2_FTP_PORT_v4 : The syntax is “PORT<space>a1,a2,a3,a4,p1,p2”. It Specifies the host TN2 and port to which the server should connect for the next file transfer. This is interpreted as IP address a1.a2.a3.a4, port p1*256+p2. In our specific case we have “PORT 131,254,201,1,192,75”.

TN2_FTP_EPRT_v6 : The EPRT command allows for the specification of an extended address for the data connection. The extended address MUST consist of the network protocol as well as the network and transport addresses. The syntax is “EPRT<space><d><net-prt><d><net-addr><d><tcp-port><d>” where:

- <d> is a delimiter character (the character "|" is recommended).
- <net-prt> is the protocol (value is 2 for IPv6)
- <net-addr> is **TN2 IPv6 Global NAT-PT Address**
- <tcp-port> is the same than in the corresponding IPv4/TCP/FTP packet.

In our specific case we have “EPRT |2|3ffe:501:41c:c1ad::83fe:c901|49227|”.

Procedure:

0. Create a State in the NUT for TN1.
1. Establish a TCP connection from TN2 port 49226 to TN1 port 21. as defined in 5.3.3.2.
2. TN2 sends “IPv4/FTP/TCP PORT” packet to TN1
3. Establish a TCP connection from TN1 port 20 to TN2 port 49227 as defined in 5.1.2.1.
4. Close the TCP connection from TN1 port 20 to TN2 port 49227 as defined in 5.1.2.1.
5. Close the TCP connection from TN2 port 49226 to TN1 port 21 as defined in 5.3.3.2

Observable Results:

- **Step 1:** The FTP-ALG on the NUT translates the packet in an “IPv6/TCP/FTP EPRT” packet.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
1: Establishement of a TCP connection from TN2 port 49226 to TN1 port 21				
TN1		2	IPv4/FTP/TCP PORT <----------	TN2
TN1	IPv6/FTP/TCP EPRT <----------	Step2		TN2
3: Establishement of a TCP connection from TN1 port 20 to TN2 port 49227				
4: Closing of the TCP connection from TN1 port 20 to TN1 port 49227				
5: Closing of the TCP connection from TN2 port 49226 to TN1 port 21				

5.3.8.2 PASV

Purpose:

Check the correct translation of PASV command into EPSV command

References:

- **[RFC2766]** ***Section 6.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- A FTP-ALG MUST be available on the NUT

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

According to **[RFC2428]**, the FTP commands PORT and PASV are replaced with EPRT and EPSV. Nevertheless because a v4 node may not have the EPRT and EPSV commands implemented, it may originates an FTP session using PORT or PASV command.

In passive mode FTP between TN2 and TN1 and from the IPv4 side, the client (TN2) initiates both connections (data and command) to the server, solving the problem of firewalls filtering the incoming data port connection to the client from the server. When opening an FTP connection, the client opens two random unprivileged ports locally (N > 1024 and N+1). The first port contacts the server on port 21, but instead of then issuing a PORT command and allowing the server to connect back to its data port, the client will issue the PASV command. The result of this is that the server then opens a random unprivileged port (P > 1024) and sends the PORT P command back to the client. The client then initiates the connection from port N+1 to port P on the server to transfer data.

The FTP-ALG has to translate this PASV command into EPSV command before to forward it to the v6 node TN1.

Packets:

- IPv4/TCP/FTP PASV (length: 46 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: 0
TotalLength: 46
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 6
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

TCP (length: 20)

SourcePort: (49326)
DestinationPort: **21**
SequenceNumber: (1)
AcknowledgmentNumber: (2)
DataOffset: 5
Reserverd: (0)
URGFlag: (0)
ACKFlag: (1)

PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (65535) Checksum: To calculate UrgentPointer: (0)
FTP – Command PASV (length: 6) **Request: "PASV"**

- IPv4/TCP/FTP PASV Response (length: 90 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 90 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the IPv6/TCP/FTP EPSV Response]** Protocol: 6 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
TCP (length: 20) SourcePort: **21** DestinationPort: (49326) SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (65535) Checksum: To calculate UrgentPointer: (0)
FTP – Response PASV (length: 50) **Response: TN1_FTP_PASV_v4**

- IPv6/TCP/FTP EPSV Response (length: 108 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 68

NextHeader: 6 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
TCP (length: 20) SourcePort: **21** DestinationPort: (49326) SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0)
FTP – Response EPSV (length: 48) **Response: TN1_FTP_EPSV_v6**

- IPv6/TCP/FTP EPSV(length: 66 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 26 NextHeader: 6 HopLimit : (63) **[one less than in the IPv4/TCP/FTP PASV]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
TCP (length: 20) SourcePort: (49326) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0)
FTP - Command EPSV (length: 6) **Request: "EPSV"**

TN1_FTP_PASV_v4: The full response syntax is “227 Entering Passive Mode (a1,a2,a3,a4,p1,p2)” where a1.a2.a3.a4 is the IP address and p1*256+p2 is the port number. It Specifies the host TN1 and port (for exemple 55555) to which the client should connect for the next file transfer. In our specific case we have “227 Entering Passive Mode (131,254,199,90,217,3)”.

TN1_FTP_EPSV_v6: The EPSV response has the following full syntax: “229 Entering Extended Passive Mode (<d><d><d><tcp-port><d>)” where:

- <d> is a delimiter character (the character "|" is recommended).
- <tcp-port> is the same than in the corresponding IPv4/TCP/FTP packet.

In our specific case we have “229 Entering Extended Passive Mode (|||55555|)”.

Procedure:

0. Create a State in the NUT for TN1
1. Establish a TCP connection from TN2 port 49326 to TN1 port 21 as defined in 5.3.3.2.
2. TN2 sends “IPv4/FTP/TCP PASV” packet to TN1
3. TN1 sends “IPv6/FTP/TCP PASV Response” packet to TN2 (giving TCP port 55555)
4. Establish a TCP connection from TN2 port 49327 to TN1 port 55555 as defined in 5.3.3.2..
5. Close the TCP connection from TN2 port 49327 to TN1 port 55555 as defined in 5.3.3.2.
6. Close the TCP connection from TN2 port 49326 to TN1 port 21 as defined in 5.3.3.2

Observable Results:

- **Step 2**: The FTP-ALG on the NUT translates the packet in an “IPv6/TCP/FTP EPSV” packet and forwards it to TN1.
- **Step 3:** The FTP-ALG on the NUT translates the packet in an “IPv4/TCP/FTP EPSV Response” packet and forwards it to TN2.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
1: Establishement of a TCP connection from TN2 port 49326 to TN1 port 21				
TN1		2	IPv4/FTP/TCP PASV <----------	TN2
TN1	IPv6/FTP/TCP EPSV <----------	Step2		TN2
TN1	IPv6/FTP/TCP PASV Response ---------->	3		TN2
TN1		Step3	IPv4/FTP/TCP EPSV Response ---------->	TN2
4: Establishement of a TCP connection from TN2 port 49327 to TN1 port 55555				
5: Closing of the TCP connection from from TN2 port 49327 to TN1 port 55555				
6: Closing of the TCP connection from TN2 port 49326 to TN1 port 21				

5.3.8.3 EPRT

Purpose:

Check the correct translation of EPRT command

References:

- **[RFC2766]** ***Section 6.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- A FTP-ALG MUST be available on the NUT

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

According to **[RFC2428]**, the FTP commands PORT and PASV are replaced with EPRT and EPSV.

Thus, in active mode FTP between TN2 and TN1 and from the IPv4 side, the client (TN2) connects from a random unprivileged port (N > 1024) to the FTP server's command port 21. Then, the client starts listening to port N+1 and sends the FTP command EPRT with port N+1 to the FTP server. The server will then connect back to the client's specified data port from its local data port, which is port 20.

The FTP-ALG has to translate this EPRT command before to forward it to the v6 node TN1.

Packets:

- IPv4/TCP/FTP EPRT (length: 70 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: 0
TotalLength: 70
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 6
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

TCP (length: 20)

SourcePort: (49426)
DestinationPort: **21**
SequenceNumber: (1)
AcknowledgmentNumber: (2)
DataOffset: 5
Reserverd: (0)
URGFlag: (0)
ACKFlag: (1)
PSHFlag: (1)
RSTFlag: (0)
SYNFlag: (0)
FINFlag: (0)
Window: (65535)
Checksum: To calculate
UrgentPointer: (0)

FTP – Command EPRT (length: 30) **Request: TN2_FTP_EPRT_v4**

- IPv6/TCP/FTP EPRT(length: 105 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 65 NextHeader: 6 HopLimit : (63) **[one less than in the IPv4/TCP/FTP EPRT]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
TCP (length: 20) SourcePort: (49426) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0)
FTP – command EPRT (length: 45) **Request: TN2_FTP_EPRT_v6**

TN2_FTP_EPRT_v6 : The EPRT command allows for the specification of an extended address for the data connection. The extended address MUST consist of the network protocol as well as the network and transport addresses. The syntax is “EPRT<space><d><net-prt><d><net-addr><d><tcp-port><d>” where:

- <d> is a delimiter character (the character "|" is recommended).
- <net-prt> is the protocol (value is 2 for IPv6)
- <net-addr> is **TN2 IPv6 Global NAT-PT Address**
- <tcp-port> is the same than in the corresponding IPv4/TCP/FTP packet.

In our specific case we have “EPRT |2|3ffe:501:41c:c1ad::83fe:c901|49427|”.

TN2_FTP_EPRT_v4: The syntax is similar to the previous one except that:

- <net-prt> is the protocol IPv4 (value is 1 for IPv4)
- <net-addr> is TN2 IPv4 Address

In our specific case we have “EPRT |1|131.254.201.1|49427|”.

Procedure:

0. Create a State in the NUT for TN1
1. Establish a TCP connection from TN2 port 49426 to TN1 port 21 as defined in 5.3.3.2.

2. TN2 sends “IPv4/FTP/TCP EPRT” packet to TN1
3. Establish a TCP connection from TN1 port 20 to TN2 port 49427 as defined in 5.1.2.1.
4. Close the TCP connection from TN1 port 20 to TN2 port 49427 as defined in 5.1.2.1.
5. Close the TCP connection from TN2 port 49426 to TN1 port 21 as defined in 5.3.3.2

Observable Results:

- **Step 2:** The FTP-ALG on the NUT translates the packet in an “IPv6/TCP/FTP EPRT” packet.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
1: Establishement of a TCP connection from TN2 port 49426 to TN1 port 21				
TN1		2	IPv4/FTP/TCP EPRT <----------	TN2
TN1	IPv6/FTP/TCP EPRT <----------	Step2		TN2
3: Establishement of a TCP connection from TN1 port 20 to TN2 port 49427				
4: Closing of the TCP connection from TN1 port 20 to TN1 port 49427				
5: Closing of the TCP connection from TN2 port 49426 to TN1 port 21				

5.3.8.4 EPSV

Purpose:

Check the correct translation of EPSV command

References:

- **[RFC2766]** ***Section 6.1***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- A FTP-ALG MUST be available on the NUT

Test Requirement:

- Create a State in the NUT for TN1

Discussion:

According to **[RFC2428]**, the FTP commands PORT and PASV are replaced with EPRT and EPSV.

In passive mode FTP between TN2 and TN1 and from the IPv4 side, the client (TN2) initiates both connections (data and command) to the server, solving the problem of firewalls filtering the incoming data port connection to the client from the server. When opening an FTP connection, the client opens two random unprivileged ports locally (N > 1024 and N+1). The first port contacts the server on port 21, but instead of then issuing an EPRT command and allowing the server to connect back to its data port, the client will issue the EPSV command. The result of this is that the server then opens a random unprivileged port (P > 1024) and sends the EPRT P command back to the client. The client then initiates the connection from port N+1 to port P on the server to transfer data.

The FTP-ALG has to translate this EPSV command before to forward it to the v6 node TN1.

Packets:

- IPv4/TCP/FTP EPSV (length: 46 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: 0
TotalLength: 46
Identifier: (0)
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 6
HeaderChecksum: To Calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by The NUT**

TCP (length: 20)

SourcePort: (49526)
DestinationPort: **21**
SequenceNumber: (1)
AcknowledgmentNumber: (2)
DataOffset: 5
Reserverd: (0)
URGFlag: (0)
ACKFlag: (1)
PSHFlag: (1)
RSTFlag: (0)
SYNFlag: (0)

FINFlag: (0) Window: (65535) Checksum: To calculate UrgentPointer: (0)
FTP – Command EPSV (length: 6) **Request: "EPSV"**

- IPv4/TCP/FTP EPSV Response (length: 88 bytes)

IPv4 Header (length: 20) Version: 4 IHL: 5 TypeOfService: 0 TotalLength: 88 Identifier: (0) Reserved: 0 DF: 1 MF: 0 FragmentOffset: 0 TTL: (63) **[one less than in the IPv6/TCP/FTP EPSV Response]** Protocol: 6 HeaderChecksum: To Calculate SourceAddress: **TN1 IPv4 Address given by The NUT** DestinationAddress: **TN2 IPv4 Address**
TCP (length: 20) SourcePort: **21** DestinationPort: (49526) SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (65535) Checksum: To calculate UrgentPointer: (0)
FTP – Response EPSV (length: 48) **Response: TN1_FTP_EPSV_v4**

- IPv6/TCP/FTP EPSV Response (length: 108 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 68 NextHeader: 6 HopLimit : (64)

SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
TCP (length: 20) SourcePort: **21** DestinationPort: (49526) SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0)
FTP – Response EPSV (length: 48) **Response: TN1_FTP_EPSV_v6**

- IPv6/TCP/FTP EPSV(length: 66 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: 0 PayloadLength: 26 NextHeader: 6 HopLimit : (63) **[one less than in the IPv4/TCP/FTP EPSV]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
TCP (length: 20) SourcePort: (49526) DestinationPort: **21** SequenceNumber: (1) AcknowledgmentNumber: (2) DataOffset: 5 Reserverd: (0) URGFlag: (0) ACKFlag: (1) PSHFlag: (1) RSTFlag: (0) SYNFlag: (0) FINFlag: (0) Window: (0) Checksum: To calculate UrgentPointer: (0)
FTP – Command EPSV (length: 6) **Request: "EPSV"**

TN1_FTP_EPSV_v6 : The EPSV response has the following full syntax: "229 Entering Extended Passive Mode (<d><d><d><tcp-port><d>)" where:

- <d> is a delimiter character (the character "|" is recommended).
- <tcp-port> is the same than in the corresponding IPv4/TCP/FTP packet.

In our specific case we have “229 Entering Extended Passive Mode (|||55555|)”.

TN1_FTP_EPSV_v4 : The syntax is similar to the previous one.

Procedure:

0. Create a State in the NUT for TN1
1. Establish a TCP connection from TN2 port 49526 to TN1 port 21 as defined in 5.3.3.2.
2. TN2 sends “IPv4/FTP/TCP EPSV” packet to TN1
3. TN1 sends “IPv6/FTP/TCP EPSV Response” packet to TN2 (giving TCP port 55555)
4. Establish a TCP connection from TN2 port 49527 to TN1 port 55555 as defined in 5.3.3.2
5. Close the TCP connection from TN2 port 49527 to TN1 port 55555 as defined in 5.3.3.2
6. Close the TCP connection from TN2 port 49526 to TN1 port 21 as defined in 5.3.3.2

Observable Results:

- **Step 2:** The FTP-ALG on the NUT translates the packet in an “IPv6/TCP/FTP EPSV” packet and forwards it to TN1.
- **Step 3:** The FTP-ALG on the NUT translates the packet in an “IPv4/TCP/FTP EPSV Response” packet and forwards it to TN2.

Test Sequence:

Tester	Link1 [IPv6]	RUT	Link2 [IPv4] (Forwarded by TR)	Tester(IPv6)
	1: Establishement of a TCP connection from TN2 port 49526 to TN1 port 21			
			IPv4/FTP/TCP EPSV	
TN1		2	<----------	TN2
	IPv6/FTP/TCP EPSV			
TN1	<----------	Step2		TN2
	IPv6/FTP/TCP EPSV Response			
TN1	---------->	3		TN2
			IPv4/FTP/TCP EPSV Response	
TN1		Step3	---------->	TN2
	4: Establishement of a TCP connection from TN2 port 49527 to TN1 port 55555			
	5: Closing of the TCP connection from from TN2 port 49527 to TN1 port 55555			
	6: Closing of the TCP connection from TN2 port 49526 to TN1 port 21			

5.3.9 DNS-ALG (*)

[RFC2874] defines A6 DNS records. However, because this RFC is in BCP status and because A6 records are not deployed we will put aside this kind of records to focus on AAAA records only.

Topology:

In this part, TN1 will be considered as the DNS server from the IPv6 Side and TR will be the DNS server of the outside world.

- TN1 DNS and Reverse DNS Entries will be the following:

Entries	*DNS*
Tn1bis.irisa.fr a3a3.fe00.00ff.0200.0100.ffff.501.3ffe.ip6.int	TN1Bis: 3ffe:501:ffff:100:200:ff:fe00:a3a3 tn1bis.irisa.fr

- TR DNS and Reverse DNS Entries will be the following:

Entries	*DNS*
Tn2.irisa.fr 1.201.254.131.in-addr.arpa	TN2: 131.254.201.1 tn2.irisa.fr

5.3.9.1 DNS Query & DNS Response (*)

Purpose:

Check the correct translation of DNS query and response packet.

References:

- **[RFC2766]** ***Pages 9, 10***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- A DNS-ALG MUST be available on the NUT

Test Requirement:

- Create a State in the NAT-PT Box for TN1

Topology:

In this test case, TN1 will be considered as the DNS server from the IPv6 Side.

Discussion:

If TN2 do a name look-up A for TN1bis, a DNS-ALG available on the NUT MUST change the query from A to AAAA before to forward it to TN1. Moreover, in the response *the DNS-ALG once again intercepts the DNS packet and MUST:*

- Translate DNS responses for "AAAA"records into "A" records
- Replace the V6 address resolved by the V6 DNS with the V4 address internally assigned by the NAT-PT router.

Packets:

- IPv6/UDP/DNS Query (length: 86 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 46 NextHeader: 17 HopLimit : (63) **[one less than in the IPv4/UDP/DNS Query Packet]** SourceAddress: **TN2 IPv6 Global NAT-PT Address** DestinationAddress: **TN1 IPv6 Global Address**
UDP (length: 46) SourcePort: (1000) DestinationPort: **53** Length: 46 Checksum: To calculate **DNS (length: 38)** Identifier: (614) QR: 0 Opcode: 0 AA: 0 TC: 0 RD: 0 RA: 0 Reserved: 0 Rcode: 0 QDCount: 1 ANCount: 0 NSCount: 0 ARCount: 0 **Question Entry (length: 26)** Name: **TN1Bis DNS** Type: 28 **[AAAA]** Class: 1

- IPv6/UDP/DNS Response (length: 136 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 94 NextHeader: 17 HopLimit : (64) SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
UDP (length: 94) SourcePort: (1000) DestinationPort: **53** Length: 94 Checksum: To calculate **DNS (length: 86)**

Identifier: (615)
QR: 1
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 1
ANCount: 1
NSCount: 0
ARCount: 0

DNS Question Entry (length: 26)
Name: **TN1Bis DNS**
Type: 28 **[AAAA]**
Class: 1

DNS Answer (length: 48)
Name: **TN1Bis DNS**
Type: 28 **[AAAA]**
Class: 1
TTL: (0)
Length: 16
Address: **TN2 IPv6 Global NAT-PT Address**

- IPv4/UDP/DNS Query (length: 66 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 66
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To calculate
SourceAddress: **TN2 IPv4 Address**
DestinationAddress: **TN1 IPv4 Address given by NAT-PT Box**

UDP (length: 46)

SourcePort: (1000)
DestinationPort: **53**
Length: 46
Checksum: To Calculate

DNS (length: 38)

Identifier: (614)
QR: 0
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 1
ANCount: 0
NSCount: 0

ARCount: 0

DNS Question Entry (length: 26)
Name: **TN1Bis DNS**
Type: 1 **[A]**
Class: 1

- IPv4/UDP/DNS Response (length: 125 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 125
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: **[one less than in the IPv6/UDP/DNS Response Packet]**
Protocol: 17
HeaderChecksum: To calculate
SourceAddress: **TN1 IPv4 Address given by NAT-PT Box**
DestinationAddress: **TN2 IPv4 Address**

UDP (length: 105)

SourcePort: (1000)
DestinationPort: **53**
Length: 105
Checksum: To Calculate

DNS (length: 97)

Identifier: (615)
QR: 1
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 2
ANCount: 1
NSCount: 0
ARCount: 0

Question Entry (length: 26)
Name: **TN1Bis DNS**
Type: 1 **[A]**
Class: 1

Question Entry (length: 26)
Name: **TN1Bis DNS**
Type: 28 **[AAAA]**
Class: 1

Answer (length: 33)
Name: **TN1Bis DNS**
Type: 1 **[A]**
Class: 1
TTL: (0)
Length: 4
Address: **TN2 IPv4 Address**

Procedure:

0. Create a State in the NAT-PT Box for TN1
1. TN2 sends “IPv4/UDP/DNS Query” to TN1 to get IP address of TN1Bis. (Type is A and Name is TN1Bis DNS).
2. TN1 sends “IPv6/UDP/DNS Response” to TN2 to give IPv6 address of TN1Bis.

Observable Results:

- **Step 1:** The DNS-ALG on the NAT-PT Box Device forwards the A Query to TN1 in an “IPv6/UDP/DNS Query” Packet.
- **Step 2:** The DNS-ALG on the NAT-PT Box Device forwards the Answer to TN2 in an “IPv4/UDP/DNS Response” Packet.

Test Sequence:

Tester	Link1 [IPv6]	RUT DNS-ALG	Link2 [IPv4]	Tester(IPv6)
			IPv4/UDP/DNS Query	
TN1		1	<----------	TN2
	IPv6/UDP/DNS Query			
TN1	<----------	Step1		
	IPv6/UDP/DNS Response			
TN1	---------->	2		TN2
			IPv4/UDP/DNS Response	
		Step2	---------->	TN2

5.3.9.2 Inverse DNS Query & DNS Response(*)

Purpose:

Check the correct translation of Inverse DNS query and response packet.

References:

- **[RFC2766]** ***Pages 9, 10***

Resource Requirement:

- Packet generator
- Monitor To capture Packets
- A DNS-ALG MUST be available on the NUT

Test Requirement:

- Create a State in the NAT-PT Box for TN1

Topology:

In this test case, TN1 will be considered as the DNS server from the IPv6 Side.

Discussion:

Since **[RFC3596]** deprecates references to IP6.INT in **[RFC2766]** section 4.1, we have to take into account the use of IP6.ARPA.

If TN2 do a name look-up PTR in order to get DNS Name of TN1bis, a DNS-ALG available on the NUT MUST replace the string "IN-ADDR.ARPA" with "IP6.ARPA" and the v4 address octet (in reverse order) with the corresponding v6 address octets in reverse order. The same has to be done for the response.

Packets:

- IPv6/UDP/DNS Inverse Query (length: 101 bytes)

IPv6 Header (length: 40) Version: 6 TrafficClass: (0) FlowLabel: (0) PayloadLength: 61 NextHeader: 17 HopLimit : (63) **[one less than in the IPv4/UDP/DNS Inverse Query Packet]** SourceAddress: **TN1 IPv6 Global Address** DestinationAddress: **TN2 IPv6 Global NAT-PT Address**
UDP (length: 61) SourcePort: (1000) DestinationPort: **53** Length: 61 Checksum: To calculate **DNS (length: 53)** Identifier: (616) QR: 0 Opcode: 0 AA: 0 TC: 0 RD: 0 RA: 0 Reserved: 0 Rcode: 0 QDCount: 1

ANCount: 0
NSCount: 0
ARCount: 0

Question Entry (length: 41)
Name: **TN1Bis IPv6 Reverse DNS**
Type: 12 **[PTR]**
Class: 1

- IPv6/UDP/DNS Inverse Response (length: 194 bytes)

IPv6 Header (length: 40)

Version: 6
TrafficClass: (0)
FlowLabel: (0)
PayloadLength: 154
NextHeader: 17
HopLimit : (64)
SourceAddress: **TN1 IPv6 Global Address**
DestinationAddress: **TN2 IPv6 Global NAT-PT Address**

UDP (length: 154)

SourcePort: (1000)
DestinationPort: **53**
Length: 154
Checksum: To calculate

DNS (length: 146)

Identifier: (617)
QR: 1
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 1
ANCount: 1
NSCount: 0
ARCount: 0

Question Entry (length: 53)
Name: **TN1Bis IPv6 Reverse DNS**
Type: 12 **[PTR]**
Class: 1

DNS Answer (length: 81)
Name: **TN1Bis IPv6 Reverse DNS**
Type: 12 **[PTR]**
Class: 1
TTL: (0)
Length: 22
PTRDName: **TN1Bis DNS**

- IPv4/UDP/DNS Inverse Query (length: 73 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5

TypeOfService: (0)
TotalLength: 73
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (64)
Protocol: 17
HeaderChecksum: To calculate
SourceAddress: **TN1 IPv4 Address given by NAT-PT Box**
DestinationAddress: **TN2 IPv4 Address**

UDP (length: 53)

SourcePort: (1000)
DestinationPort: **53**
Length: 53
Checksum: To Calculate

DNS (length: 45)

Identifier: (616)
QR: 0
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 2
ANCount: 0
NSCount: 0
ARCount: 0

Question Entry (length: 33)
Name: **TN1Bis IPv4 Reverse DNS**
Type: 12 **[PTR]**
Class: 1

- IPv4/UDP/DNS Inverse Response (length: 139 bytes)

IPv4 Header (length: 20)

Version: 4
IHL: 5
TypeOfService: (0)
TotalLength: 139
Identifier: 0
Reserved: 0
DF: 1
MF: 0
FragmentOffset: 0
TTL: (63) **[one less than in the IPv6/UDP/DNS Inverse Response Packet]**
Protocol: 17
HeaderChecksum: To Calculate
SourceAddress: **TN1 IPv4 Address given by NAT-PT Box**
DestinationAddress: **TN2 IPv4 Address**

UDP (length: 119)

SourcePort: (1000)
DestinationPort: **53**
Length: 119

Checksum: To Calculate

DNS (length: 111)

Identifier: (617)
QR: 1
Opcode: 0
AA: 0
TC: 0
RD: 0
RA: 0
Reserved: 0
Rcode: 0
QDCount: 1
ANCount: 1
NSCount: 0
ARCount: 0

Question Entry (length: 33)
Name: **TN1Bis IPv4 Reverse DNS**
Type: 12 **[PTR]**
Class: 1

DNS Answer (length: 66)
Name: **TN1Bis IPv4 Reverse DNS**
Type: 12 **[PTR]**
Class: 1
TTL: (0)
Length: 19
PTRDName: **TN1Bis DNS**

Procedure:

0. Create a State in the NAT-PT Box for TN1
1. TN2 sends “IPv4/UDP/DNS Reverse Query” to TN2 to get DNS Name of TN1Bis. (Type is PTR and Name is TN1Bis IPv6 Reverse DNS).
2. TN1 sends “IPv6/UDP/DNS Reverse Response” to TN2 to give DNS Name of TN1Bis.

Observable Results:

- **Step 1:** The DNS-ALG on the NAT-PT Box Device translates the Query to TN1 in an “IPv6/UDP/DNS Reverse Query” Packet.
- **Step 2:** The DNS-ALG on the NAT-PT Box Device forwards the Answer to TN2 in an “IPv4/UDP/DNS Reverse Response” Packet.

Test Sequence:

Tester	Link1 [IPv6]	RUT DNS-ALG	Link2 [IPv4]	Tester(IPv6)
			IPv4/UDP/DNS Reverse Query	
TN1		1	<----------	TN2
	IPv6/UDP/DNS Reverse Query			
TN1	<----------	Step1		
	IPv6/UDP/DNS Reverse Response			
TN1	---------->	2		TN2
			IPv4/UDP/DNS Reverse Response	
		Step2	---------->	TN2

6 Annexes

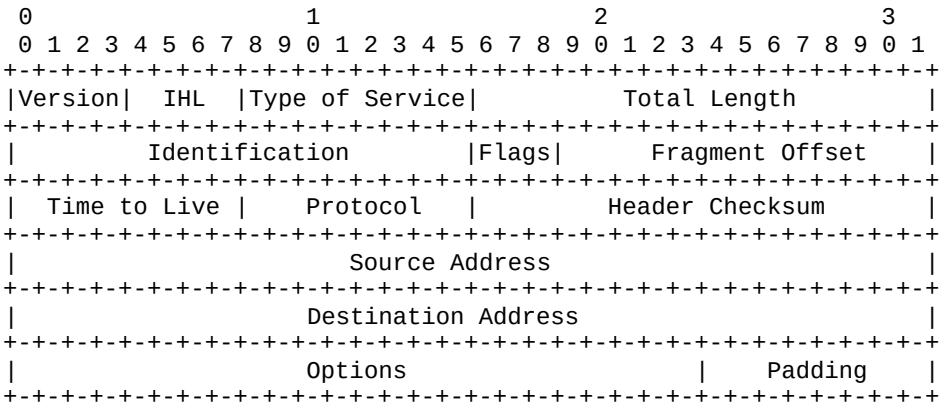
In this annexe we present how to calculate checksums of this test suite. It will be helpful for packets with the key-word "To Calculate".

6.1 IPv4 packets Checksum computation

6.1.1 IPv4 Checksum

The IPv4 Internet Header Format and the IPv4 Header Checksum calculation is defined in **[RFC791].**

The IPv4 Header Format is the following:

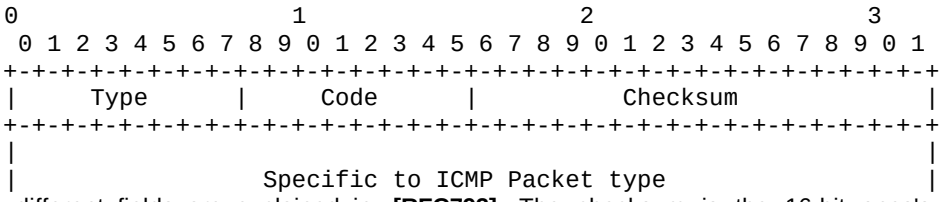


The different fields are explained in **[RFC791].** The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

6.1.2 ICMPv4 Checksum

The ICMPv4 Header Format and the ICMPv4 Header Checksum calculation is defined in **[RFC792].**

The ICMPv4 Header Format is the following:

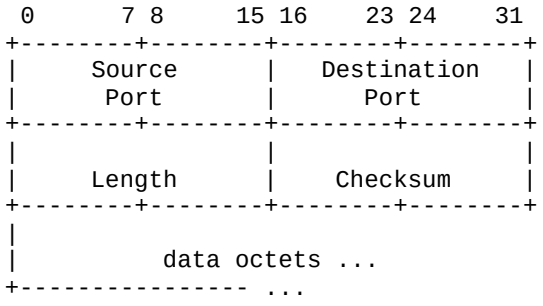


The different fields are explained in **[RFC792].** The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type.for computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

6.1.3 UDP Checksum

The UDP Header Format for IPv4 and the UDP Header Checksum calculation is defined in **[RFC768].**

The UDP Header Format is the following:

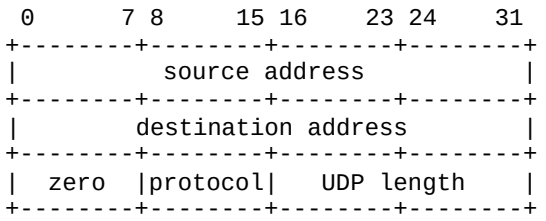


The different fields are explained in **[RFC768].**

Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

The pseudo header conceptually prefixed to the UDP header contains the source address, the destination address, the protocol, and the UDP length. This information gives protection against misrouted datagrams.

This checksum procedure is the same as is used in TCP.

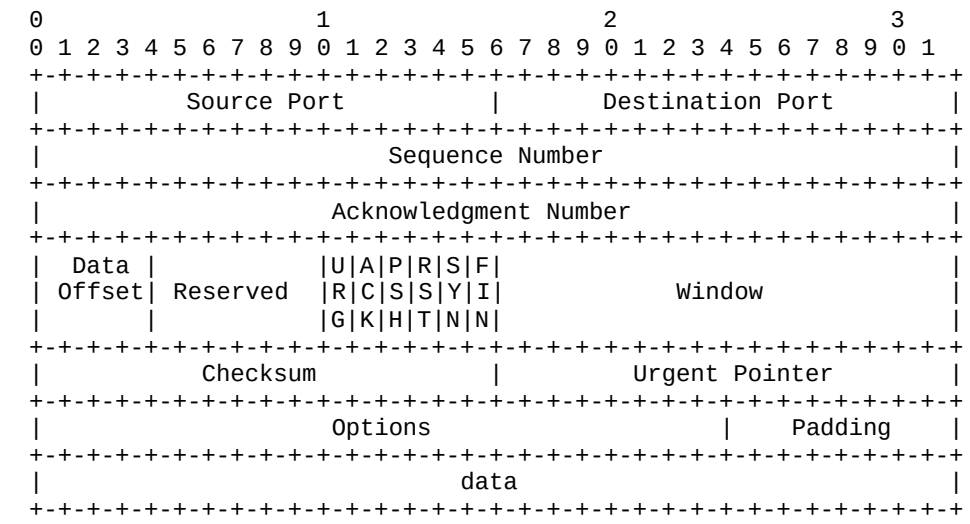


If the computed checksum is zero, it is transmitted as all ones (the equivalent in one's complement arithmetic). An all zero transmitted checksum value means that the transmitter generated no checksum for debugging or for higher level protocols that don't care).

6.1.4 TCP Checksum

The TCP Header Format for IPv4 and the TCP Header Checksum calculation is defined in **[RFC793].**

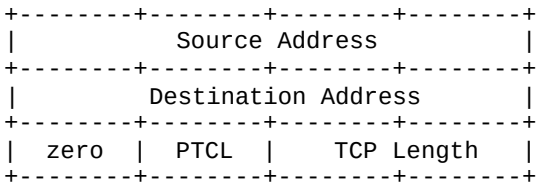
The TCP Header Format is the following:



The different fields are explained in **[RFC793].**

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros toform a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

The checksum also covers a 96 bit pseudo header conceptually prefixed to the TCP header. This pseudo header contains the Source Address, the Destination Address, the Protocol, and TCP length. This gives the TCP protection against misrouted segments. This information is carried in the Internet Protocol and is transferred across the TCP/Network interface in the arguments or results of calls by the TCP on the IP.

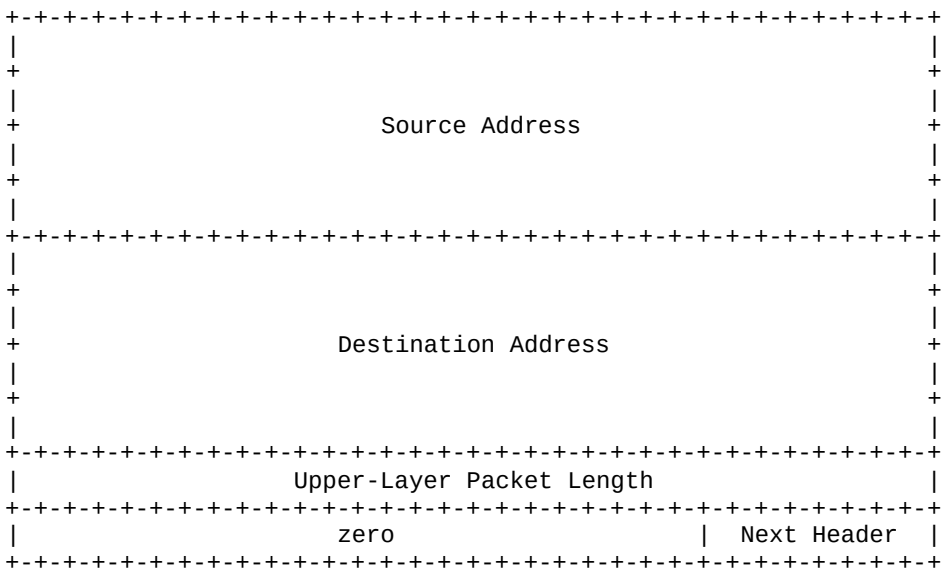


The TCP Length is the TCP header length plus the data length in octets (this is not an explicitly transmitted quantity, but is computed), and it does not count the 12 octets of the pseudo header.

6.2 IPv6 packets Checksum computation

6.2.1 Pseudo-header

Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. In particular, the following illustration **[RFC2460]** shows the TCP and UDP "pseudo-header" for IPv6:



If the IPv6 packet contains a Routing header, the Destination Address used in the pseudo-header is that of the final destination. At the originating node, that address will be in the last element of the Routing header; at the recipient(s), that address will be in the Destination Address field of the IPv6 header.

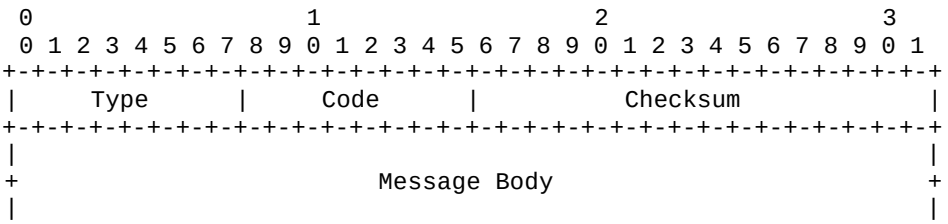
The Next Header value in the pseudo-header identifies the upper-layer protocol (e.g., 6 for TCP, or 17 for UDP). It will differ from the Next Header value in the IPv6 header if there are extension headers between the IPv6 header and the upper-layer header.

The Upper-Layer Packet Length in the pseudo-header is the length of the upper-layer header and data (e.g., TCP header plus TCP data). Some upper-layer protocols carry their own length information (e.g., the Length field in the UDP header); for such protocols, that is the length used in the pseudo-header. Other protocols (such as TCP) do not carry their own length information, in which case the length used in the pseudo-header is the Payload Length from the IPv6 header, minus the length of any extension headers present between the IPv6 header and the upper-layer header.

6.2.2 ICMPv6 Checksum

The ICMPv6 Header Format and the ICMPv6 Header Checksum calculation is defined in **[RFC2463].**

The ICMPv6 Header Format is the following:



The different fields are explained in **[RFC768].**

The checksum is the 16-bit one's complement of the one's complement sum of the entire ICMPv6 message starting with the ICMPv6 message type field, prepended with the previous "pseudo-header". The Next Header value used in the pseudo-header is 58.

6.2.3 UDP and TCP Checksums

The UDP/TCP Header Formats and checksums calculation are identical in IPv4 and IPv6 but IPv6 uses the previous pseudo-header in its checksum calculation

Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header.

7 References

[MILLER]

G. Miller, Email to the ngtrans mailing list on 26 March 1999.

[RFC768]

RFC 768, STD0006, User Datagram Protocol, J. Postel, August 1980, STANDARD

[RFC791]

RFC 791, STD0005, Internet Protocol, J. Postel, Sep-01-1981, STANDARD

[RFC792]

STD0005, RFC 792, Internet Control Message Protoco, lJ. Postel, September1981, STANDARD

[RFC793]

RFC 793, STD0007, Transmission Control Protocol, J. Postel, September 1981, STANDARD

[RFC0959]

RFC 959, STD 0009, File Transfer Protocol, J. Postel, J.K. Reynolds, Oct-01-1985, STD

[RFC1191]

RFC 1191, Path MTU discovery, J.C. Mogul, S.E. Deering, November 1990, DRAFT STANDARD

[RFC1631]

RFC 1631, The IP Network Address Translator (NAT), K. Egevang, P. Francis May, INFORMATIONAL

[RFC1918]

RFC 1918, BCP0005, Address Allocation for Private Internets, Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, February 1996, BEST CURRENT PRACTICE

[RFC2428]

RFC 2428, FTP Extensions for IPv6 and NATs, M. Allman, S. Ostermann, C. Metz, September 1998, PROPOSED STANDARD

[RFC2460]

RFC 2460, "Internet Protocol, Version 6 (IPv6) Specification", Deering, S., and R. Hinden, December 1998, DRAFT STANDARD

[RFC2461]

RFC 2461, "Neighbor Discovery for IP Version 6 (IPv6)", Narten, T., Nordmark, E. and W. Simpson, December 1998, DRAFT STANDARD

[RFC2462]

RFC 2462, Thomson, S., and T. Narten, "IPv6 Stateless Address Autoconfiguration", December 1998, DRAFT STANDARD

[RFC2463]

RFC 2463, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", Conta, A., and S. Deering, December 1998, DRAFT STANDARD

[RFC2464]

RFC2464, Transmission of IPv6 Packets over Ethernet Networks, M. Crawford, December 1998, PROPOSED STANDARD

[RFC2474]

RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, K. Nichols, S. Blake, F. Baker, D. Black, December 1998, PROPOSED STANDARD

[RFC2710]

RFC2710, Multicast Listener Discovery (MLD) for IPv6, S. Deering, W. Fenner, B. Haberman, October 1999, PROPOSED STANDARD

[RFC2765]

RFC 2765, Stateless IP/ICMP Translation Algorithm (SIIT), E. Nordmark, February 2000, PROPOSED STANDARD

[RFC2766]

RFC 2766, Network Address Translation - Protocol Translation (NAT-PT), G. Tsirtsis, P. Srisuresh, February 2000, PROPOSED STANDARD

[RFC2874]

RFC 2874, DNS Extensions to Support IPv6 Address Aggregation and Renumbering, M. Crawford, C. Huitema, July 2000, EXPERIMENTAL [pub as:PROPOSED STANDARD]

[RFC2893]

RFC 2893, Transition Mechanisms for IPv6 Hosts and Routers, R. Gilligan, E. Nordmark, August 2000, PROPOSED STANDARD

[RFC3022]

RFC 3022, Traditional IP Network Address Translator (Traditional NAT), P. Srisuresh, K. Egevang, January 2001, INFORMATIONAL

[RFC3513]

RFC 3513, Internet Protocol Version 6 (IPv6) Addressing Architecture, R. Hinden, S. Deering, April 2003, PROPOSED STANDARD

[RFC3596]

RFC 3596, DNS Extensions to Support IP Version 6, S. Thomson, C. Huitema, V. Ksinant, M. Souissi, October 2003, DRAFT STANDARD