BEHAVE WG M. Bagnulo
Internet-Draft UC3M
Intended status: Standards Track P. Matthews
Expires: May 25, 2010 Alcatel-Lucent
I. van Beijnum
IMDEA Networks
November 21, 2009

NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers
draft-ietf-behave-v6v4-xlate-stateful-03

Abstract

NAT64 is a mechanism for translating IPv6 packets to IPv4 packets and vice-versa. DNS64 is a mechanism for synthesizing AAAA records from A records. These two mechanisms together enable client-server communication between an IPv6-only client and an IPv4-only server, without requiring any changes to either the IPv6 or the IPv4 node, for the class of applications that work through NATs. They also enable peer-to-peer communication between an IPv4 and an IPv6 node, where the communication can be initiated by either end using existing, NAT-traversing, peer-to-peer communication techniques. NAT64 also support IPv4 initiated communications to a subset of the IPv6 hosts through statically configured bindings in the NAT64 device. This document specifies NAT64, and gives suggestions on how they should be deployed.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document specifies NAT64, a mechanism for IPv6-IPv4 transition and co-existence. Together with DNS64 [I-D.ietf-behave-dns64], these two mechanisms allow a IPv6-only client to initiate communications to an IPv4-only server, also allow peer-to-peer communication between IPv6-only and IPv4-only hosts. NAT64 also support IPv4 initiated communications to a subset of the IPv6 hosts through statically configured bindings in the NAT64 device.

NAT64 is a mechanism for translating IPv6 packets to IPv4 packets and vice-versa. The translation is done by translating the packet headers according to IP/ICMP Translation Algorithm [I-D.ietf-behave-v6v4-xlate], translating the IPv4 server address by adding or removing an IPv6 prefix, and translating the IPv6 client address by installing mappings in the normal NAT manner.

DNS64 is a mechanism for synthesizing AAAA resource records (RR) from A RR. The synthesis is done by adding a IPv6 prefix to the IPv4 address to create an IPv6 address, where the IPv6 prefix is assigned to a NAT64 device.

Together, these two mechanisms allow a IPv6-only client to initiate communications to an IPv4-only server.

These mechanisms are expected to play a critical role in the IPv4-IPv6 transition and co-existence. Due to IPv4 address depletion, it's likely that in the future, a lot of IPv6-only clients will want to connect to IPv4-only servers. The NAT64 and DNS64 mechanisms are easily deployable, since they require no changes to either the IPv6 client nor the IPv4 server. For basic functionality, the approach only requires the deployment of NAT64 function in the devices connecting an IPv6-only network to the IPv4-only network, along with the deployment of a few DNS64-enabled name servers in the IPv6-only network. However, some advanced features such as support for DNSSEC validating stub resolvers or support for some IPsec modes, require software updates to the IPv6-only hosts.

The NAT64 and DNS64 mechanisms are related to the NAT-PT mechanism defined in [RFC2766], but significant differences exist. First, NAT64 does not define the NATPT mechanisms used to support the general case of IPv6 only servers to be contacted by IPv4 only clients, but only defines the mechanisms for IPv6 clients to contact IPv4 servers and its potential reuse to support peer to peer communications through standard NAT traversal techniques. Second, NAT64 includes a set of features that overcomes many of the reasons the original NAT-PT specification was moved to historic status [RFC4966].

1.1. Features of NAT64

The features of NAT64 and DNS64 are:

- It enables IPv6-only nodes to initiate a client-server connection with an IPv4-only server, without needing any changes on either IPv4 or IPv6 nodes. This works for roughly the same class of applications that work through IPv4-to-IPv4 NATs.

- It supports peer-to-peer communication between IPv4 and IPv6 nodes, including the ability for IPv4 nodes to initiate communication with IPv6 nodes using peer-to-peer techniques (i.e., using a rendezvous server and ICE). To this end, NAT64 is compliant with the recommendations for how NATs should handle UDP [RFC4787], TCP [RFC4787], and ICMP [RFC5508].

- NAT64 supports IPv4 initiated communications to a subset of the IPv6 hosts through statically configured bindings in the NAT64 device.

- Compatible with ICE.

- o Supports additional features with some changes on nodes. These features include:

  - * Support for DNSSEC

  - * Some forms of IPsec support

1.2. Overview

This section provides a non-normative introduction to the mechanisms of NAT64.

NAT64 mechanism is implemented in an NAT64 box which has two interfaces, an IPv4 interface connected to the the IPv4 network, and an IPv6 interface connected to the IPv6 network. Packets generated in the IPv6 network for a receiver located in the IPv4 network will be routed within the IPv6 network towards the NAT64 box. The NAT64 box will translate them and forward them as IPv4 packets through the IPv4 network to the IPv4 receiver. The reverse takes place for packets generated in the IPv4 network for an IPv6 receiver. NAT64, however, is not symmetric. In order to be able to perform IPv6 - IPv4 translation NAT64 requires state, binding an IPv6 address and port (hereafter called an IPv6 transport address) to an IPv4 address and port (hereafter called an IPv4 transport address).

Such binding state is created when the first packet flowing from the

IPv6 network to the IPv4 network is translated. After the binding state has been created, packets flowing in either direction on that particular flow are translated. The result is that NAT64 only supports communications initiated by the IPv6-only node towards an IPv4-only node. Some additional mechanisms, like ICE, can be used in combination with NAT64 to provide support for communications initiated by the IPv4-only node to the IPv6-only node. The specification of such mechanisms, however, is out of the scope of this document.

1.2.1. NAT64 solution elements

In this section we describe the different elements involved in the NAT64 approach.

The main component of the proposed solution is the translator itself. The translator has essentially two main parts, the address translation mechanism and the protocol translation mechanism.

Protocol translation from IPv4 packet header to IPv6 packet header and vice-versa is performed according to IP/ICMP Translation Algorithm [I-D.ietf-behave-v6v4-xlate].

Address translation maps IPv6 transport addresses to IPv4 transport addresses and vice-versa. In order to create these mappings the NAT64 box has two pools of addresses i.e. an IPv6 address pool (to represent IPv4 addresses in the IPv6 network) and an IPv4 address pool (to represent IPv6 addresses in the IPv4 network). Since there is enough IPv6 address space, it is possible to map every IPv4 address into a different IPv6 address.

NAT64 creates the required mappings by using as the IPv6 address pool an IPv6 IPv6 prefix (hereafter called Pref64::/n). This allows each IPv4 address to be mapped into a different IPv6 address by simply concatenating the Pref64::/n prefix assigned as the IPv6 address pool of the NAT64, with the IPv4 address being mapped and a suffix (i.e. an IPv4 address X is mapped into the IPv6 address Pref64:X:SUFFIX). The NAT64 prefix Pref64::/n is assigned by the administrator of the NAT64 box from the global unicast IPv6 address block assigned to the site.

The IPv4 address pool is a set of IPv4 addresses, normally a small prefix assigned by the local administrator. Since IPv4 address space is a scarce resource, the IPv4 address pool is small and typically not sufficient to establish permanent one-to-one mappings with IPv6 addresses. So, mappings using the IPv4 address pool will be created and released dynamically. Moreover, because of the IPv4 address scarcity, the usual practice for NAT64 is likely to be the mapping of

IPv6 transport addresses into IPv4 transport addresses, instead of IPv6 addresses into IPv4 addresses directly, which enable a higher utilization of the limited IPv4 address pool.

Because of the dynamic nature of the IPv6 to IPv4 address mapping and the static nature of the IPv4 to IPv6 address mapping, it is easy to understand that it is far simpler to allow communication initiated from the IPv6 side toward an IPv4 node, which address is permanently mapped into an IPv6 address, than communications initiated from IPv4-only nodes to an IPv6 node in which case IPv4 address needs to be associated with it dynamically. For this reason NAT64 supports only communications initiated from the IPv6 side.

An IPv6 initiator can know or derive in advance the IPv6 address representing the IPv4 target and send packets to that address. The packets are intercepted by the NAT64 device, which associates an IPv4 transport address of its IPv4 pool to the IPv6 transport address of the initiator, creating binding state, so that reply packets can be translated and forwarded back to the initiator. The binding state is kept while packets are flowing. Once the flow stops, and based on a timer, the IPv4 transport address is returned to the IPv4 address pool so that it can be reused for other communications.

To allow an IPv6 initiator to do the standard DNS lookup to learn the address of the responder, DNS64 [I-D.ietf-behave-dns64] is used to synthesize an AAAA RR from the A RR (containing the real IPv4 address of the responder). DNS64 receives the DNS queries generated by the IPv6 initiator. If there is no AAAA record available for the target node (which is the normal case when the target node is an IPv4-only node), DNS64 performs a query for the A record. If an A record is discovered, DNS64 creates a synthetic AAAA RR that includes the IPv6 representations of the IPv4 address created by concatenating the Pref64::/n of a NAT64 to the responder's IPv4 address and a suffix (i.e. if the IPv4 node has IPv4 address X, then the synthetic AAAA RR will contain the IPv6 address formed as Pref64:X:SUFFIX). The synthetic AAAA RR is passed back to the IPv6 initiator, which will initiate an IPv6 communication with the IPv6 address associated to the IPv4 receiver. The packet will be routed to the NAT64 device, which will create the IPv6 to IPv4 address mapping as described before.

1.2.2. Walkthrough

In this example, we consider an IPv6 node located in a IPv6-only site that initiates a communication to a IPv4 node located in the IPv4 network.

The notation used is the following: upper case letters are IPv4

addresses; upper case letters with a prime(') are IPv6 addresses;

lower case letters are ports; prefixes are indicated by "P::X", which is a IPv6 address built from an IPv4 address X by adding the prefix P, mappings are indicated as "(X,x) <--> (Y',y)".

The scenario for this case is depicted in the following figure:

```
     +---------------------------------------+       +---------------+
     |IPv6 network    +-------------+        |       |               |
     |  +----+        | Name server |   +-------+    |   IPv4        |
     |  | H1 |        | with DNS64  |   | NAT64 |----| Network       |
     |  +----+        +-------------+   +-------+    |               |
     |    |IP addr: Y'     |              |  |       |  IP addr: X   |
     |    ---------------------------------  |       |  +----+       |
     +---------------------------------------+       |  | H2 |       |
                                                     |  +----+       |
                                                     +---------------+
```

The figure shows a IPv6 node H1 which has an IPv6 address Y' and an IPv4 node H2 with IPv4 address X.

A NAT64 connects the IPv6 network to the IPv4 network. This NAT64 has a /n prefix (called Pref64::/n) that it uses to represent IPv4 addresses in the IPv6 address space and an IPv4 address T assigned to its IPv4 interface. the routing is configured in such a way, that the IPv6 packets addressed to a destination address containing Pref64::/n are routed to the IPv6 interface of the NAT64 box.

Also shown is a local name server with DNS64 functionality. The local name server needs to know the /n prefix assigned to the local NAT64 (Pref64::/n). For the purpose of this example, we assume it learns this through manual configuration.

For this example, assume the typical DNS situation where IPv6 hosts have only stub resolvers and the local name server does the recursive lookups.

The steps by which H1 establishes communication with H2 are:

1. H1 performs a DNS query for FQDN(H2) and receives the synthetic AAAA RR from the local name server that implements the DNS64 functionality. The AAAA record contains an IPv6 address formed by the Pref64::/n associated to the NAT64 box and the IPv4 address of H2 and a suffix.

2. H1 sends a packet to H2. The packet is sent from a source transport address of (Y',y) to a destination transport address of

(Pref64:X:SUFFIX,x), where y and x are ports set by H1.

3. The packet is routed to the IPv6 interface of the NAT64 (since the IPv6 routing is configured that way).

4. The NAT64 receives the packet and performs the following actions:

   * The NAT64 selects an unused port t on its IPv4 address T and creates the mapping entry (Y',y) <--> (T,t)

   * The NAT64 translates the IPv6 header into an IPv4 header using IP/ICMP Translation Algorithm [I-D.ietf-behave-v6v4-xlate].

   * The NAT64 includes (T,t) as source transport address in the packet and (X,x) as destination transport address in the packet. Note that X is extracted directly from the destination IPv6 address of the received IPv6 packet that is being translated.

5. The NAT64 sends the translated packet out its IPv4 interface and the packet arrives at H2.

6. H2 node responds by sending a packet with destination transport address (T,t) and source transport address (X,x).

7. The packet is routed to the NAT64 box, which will look for an existing mapping containing (T,t). Since the mapping (Y',y) <--> (T,t) exists, the NAT64 performs the following operations:

   * The NAT64 translates the IPv4 header into an IPv6 header using IP/ICMP Translation Algorithm [I-D.ietf-behave-v6v4-xlate].

   * The NAT64 includes (Y',y) as destination transport address in the packet and (Pref64:X:SUFFIX,x) as source transport address in the packet. Note that X is extracted directly from the source IPv4 address of the received IPv4 packet that is being translated.

8. The translated packet is sent out the IPv6 interface to H1.

The packet exchange between H1 and H2 continues and packets are translated in the different directions as previously described.

It is important to note that the translation still works if the IPv6 initiator H1 learns the IPv6 representation of H2's IPv4 address (i.e. Pref64:X:SUFFIX) through some scheme other than a DNS look-up.

This is because the DNS64 processing does NOT result in any state installed in the NAT64 box and because the mapping of the IPv4

address into an IPv6 address is the result of concatenating the prefix defined within the site for this purpose (called Pref64::/n in this document) to the original IPv4 address and a suffix.

1.2.3. Filtering

A NAT64 box may do filtering, which means that it only allows a packet in through an interface if the appropriate permission exists. A NAT64 may do no filtering, or it may filter on its IPv4 interface. Filtering on the IPv6 interface is not supported, as mappings are only created by packets traveling in the IPv6 --> IPv4 direction.

If a NAT64 performs address-dependent filtering according to RFC4787 [RFC4787] on its IPv4 interface, then an incoming packet is dropped unless a packet has been recently sent out the interface with a source transport address equal to the destination transport address of the incoming packet and destination IP address equal to the source IP address of the incoming packet.

NAT64 filtering is consistent with the recommendations of RFC 4787 [RFC4787], and the ones of RFC 5382 [RFC5382]

2. Terminology

This section provides a definitive reference for all the terms used in document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The following terms are used in this document:

3-Tuple: The tuple (source IP address, destination IP address, Query Identifier). A 3-tuple uniquely identifies an ICMP Query session. When an ICMP Query session flows through a NAT64, each session has two different 3-tuples: one with IPv4 addresses and one with IPv6 addresses.

5-Tuple: The tuple (source IP address, source port, destination IP address, destination port, transport protocol). A 5-tuple uniquely identifies a UDP/TCP session. When a UDP/TCP session flows through a NAT64, each session has two different 5-tuples: one with IPv4 addresses and one with IPv6 addresses.

BIB: Binding Information Base. A table of mappings kept by a NAT64. Each NAT64 has three BIBs, one for TCP, one for UDP and one for ICMP Queries.

DNS64: A logical function that synthesizes AAAA Resource Records (containing IPv6 addresses) from A Resource Records (containing IPv4 addresses).

Endpoint-Independent Mapping: In NAT64, using the same mapping for all the sessions involving a given IPv6 transport address of an IPv6 host (irrespectively of the transport address of the IPv4 host involved in the communication). Endpoint-independent mapping is important for peer-to-peer communication. See [RFC4787] for the definition of the different types of mappings in IPv4-to-IPv4 NATs.

Hairpinning: Having a packet do a "U-turn" inside a NAT and come back out the same interface as it arrived on. Hairpinning support is important for peer-to-peer applications, as there are cases when two different hosts on the same side of a NAT can only communicate using sessions that hairpin through the NAT.

Mapping: A mapping between an IPv6 transport address and a IPv4 transport address. Used to translate the addresses and ports of packets flowing between the IPv6 host and the IPv4 host. In NAT64, the IPv4 transport address is always a transport address assigned to the NAT64 itself, while the IPv6 transport address belongs to some IPv6 host.

NAT64: A device that translates IPv6 packets to IPv4 packets and vice-versa, with the provision that the communication must be initiated from the IPv6 side. The translation involves not only the IP header, but also the transport header (TCP or UDP).

Session: A TCP, UDP or ICMP Query session. In other words, the bi-directional flow of packets between two ports on two different hosts. In NAT64, typically one host is an IPv4 host, and the other one is an IPv6 host.

Session table: A table of sessions kept by a NAT64. Each NAT64 has three session tables, one for TCP, one for UDP and one for ICMP Queries.

Synthetic RR: A DNS Resource Record (RR) that is not contained in any zone data file, but has been synthesized from other RRs. An example is a synthetic AAAA record created from an A record.

Transport Address: The combination of an IPv6 or IPv4 address and a port. Typically written as (IP address, port); e.g. (192.0.2.15, 8001).

Tuple: Refers to either a 3-Tuple or a 5-tuple as defined above.

For a detailed understanding of this document, the reader should also be familiar with DNS terminology [RFC1035] and current NAT terminology [RFC4787].

3. NAT64 Normative Specification

A NAT64 is a device with at least one IPv6 interface and at least one IPv4 interface. Each NAT64 device MUST have one unicast /n IPv6 prefix assigned to it, denoted Pref64::/n. Additional consideration about the Pref64::/n are presented in Section 3.2.5. Each NAT64 box MUST have one or more unicast IPv4 addresses assigned to it.

A NAT64 uses the following dynamic data structures:

- UDP Binding Information Base

- UDP Session Table

- TCP Binding Information Base

- TCP Session Table

- ICMP Query Binding Information Base

- ICMP Query Session Table

A NAT64 has three Binding Information Bases (BIBs): one for TCP, one for UDP and one for ICMP Queries. In the case of UDP and TCP BIBs, each BIB entry specifies a mapping between an IPv6 transport address and an IPv4 transport address:

```
   (X',x) <--> (T,t)
```

where X' is some IPv6 address, T is an IPv4 address, and x and t are ports. T will always be one of the IPv4 addresses assigned to the NAT64. A given IPv6 or IPv4 transport address can appear in at most one entry in a BIB: for example, (2001:db8::17, 4) can appear in at most one TCP and at most one UDP BIB entry. TCP and UDP have separate BIBs because the port number space for TCP and UDP are distinct.

In the case of the ICMP Query BIB, each ICMP Query BIB entry specify a mapping between an (IPv6 address, Query Identifier) pair and an (IPv4 address, Query Identifier pair).

```
   (X',I1) <--> (T,I2)
```

where X' is some IPv6 address, T is an IPv4 address, and I1 and I2 are Query Identifiers. T will always be one of the IPv4 addresses assigned to the NAT64. A given (IPv6 or IPv4 address, Query Id) pair can appear in at most one entry in the ICMP Query BIB.

Entries in any of the three BIBs can be created dynamically as the result of the flow of packets as described in the section Section 3.2 but the can also can be created manually by the system administrator. NAT64 implementations SHOULD support manually configured BIB entries for any of the three BIBs. Dynamically-created entries are deleted from the corresponding BIB when the last session associated to the BIB entry is removed from the session table. Manually-configured BIB entries are not deleted when there is no corresponding session table entry and can only be deleted by the administrator.

A NAT64 also has three session tables: one for TCP sessions, one for UDP sessions and one for ICMP Query sessions. Each entry keeps information on the state of the corresponding session. In the TCP and UDP session tables, each entry specifies a mapping between a pair of IPv6 transport address and a pair of IPv4 transport address:

```
   (X',x),(Y',y) <--> (T,t),(Z,z)
```

where X' and Y' are IPv6 addresses, T and Z are IPv4 addresses, and x, y, z and t are ports. T will always be one of the IPv4 addresses assigned to the NAT64. Y' is always the IPv6 representation of the IPv4 address Z, so Y' is obtained from Z using the algorithm applied by the NAT64 to create IPv6 representations of IPv4 addresses. y is always equal to z. In addition, each session table entry has a lifetime.

In the ICMP query session table, each entry specifies a mapping between a 3-tuple of IPv6 source address, IPv6 destination address and ICMPv6 Query Id and a 3-tuple of IPv4 source address, IPv4 destination address and ICMPv4 Query Id:

```
   (X',Y',I1) <--> (T,Z,I2)
```

where X' and Y' are IPv6 addresses, T and Z are IPv4 addresses, and I1 and I2 are ICMP query Ids. T will always be one of the IPv4 addresses assigned to the NAT64. Y' is always the IPv6 representation of the IPv4 address Z, so Y' is obtained from Z using

the algorithm applied by the NAT64 to create IPv6 representations of IPv4 addresses. In addition, each session table entry has a lifetime.

The NAT64 uses the session state information to determine when the session is completed, and also uses session information for ingress filtering. A session can be uniquely identified by either an incoming tuple or an outgoing tuple.

For each session, there is a corresponding BIB entry, uniquely specified by either the source IPv6 transport address or the source

IPv6 address and ICMPv6 Query Id (in the IPv6 --> IPv4 direction) or the destination IPv4 transport address or the destination IPv4 address and the ICMPv4 Query Id (in the IPv4 --> IPv6 direction). However, a single BIB entry can have multiple corresponding sessions. When the last corresponding session is deleted, if the BIB entry was dynamically created, the BIB entry is deleted.

The processing of an incoming IP packet takes the following steps:

1. Determining the incoming tuple

2. Filtering and updating binding and session information

3. Computing the outgoing tuple

4. Translating the packet

5. Handling hairpinning

The details of these steps are specified in the following subsections.

This breakdown of the NAT64 behavior into processing steps is done for ease of presentation. A NAT64 MAY perform the steps in a different order, or MAY perform different steps, as long as the externally visible outcome is the same.

3.1. Determining the Incoming tuple

This step associates a incoming tuple with every incoming IP packet for use in subsequent steps. In the case of TCP, UDP and ICMP error packets, the tuple is a 5-tuple consisting of source IP address, source port, destination IP address, destination port, transport protocol. In case of ICMP Queries, the tuple is a 3-tuple consisting of the source IP address, destination IP address and Query Identifier.

If the incoming IP packet contains a complete (un-fragmented) UDP or TCP protocol packet, then the 5-tuple is computed by extracting the appropriate fields from the packet.

If the incoming packet is an ICMP query message (i.e. an ICMPv4 Query message or an ICMPv6 Informational message), the 3-tuple is the source IP address. the destination IP address and the ICMP Query Identifier.

If the incoming IP packet contains a complete (un-fragmented) ICMP error message, then the 5-tuple is computed by extracting the appropriate fields from the IP packet embedded inside the ICMP error message. However, the role of source and destination is swapped when doing this: the embedded source IP address becomes the destination IP address in the 5-tuple, the embedded source port becomes the destination port in the 5-tuple, etc. If it is not possible to determine the 5-tuple (perhaps because not enough of the embedded packet is reproduced inside the ICMP message), then the incoming IP packet is silently discarded.

> NOTE: The transport protocol is always one of TCP or UDP, even if the IP packet contains an ICMP Error message.

If the incoming IP packet contains a fragment, then more processing may be needed. This specification leaves open the exact details of how a NAT64 handles incoming IP packets containing fragments, and simply requires that a NAT64 handle fragments arriving out-of-order. A NAT64 MAY elect to queue the fragments as they arrive and translate all fragments at the same time. Alternatively, a NAT64 MAY translate the fragments as they arrive, by storing information that allows it to compute the 5-tuple for fragments other than the first. In the latter case, the NAT64 will still need to handle the situation where subsequent fragments arrive before the first.

Implementors of NAT64 should be aware that there are a number of well-known attacks against IP fragmentation; see [RFC1858] and [RFC3128].

Assuming it otherwise has sufficient resources, a NAT64 MUST allow the fragments to arrive over a time interval of at least 10 seconds. A NAT64 MAY require that the UDP, TCP, or ICMP header be completely contained within the first fragment.

Except from the retrieval of 5-tuple information from the incoming packets as discussed above, all other fragmentation and PMTUD related processing performed by the NAT64 device is performed as defined in [I-D.ietf-behave-v6v4-xlate], including the translation of all related fragmentation fields in the IP header, the determination of the outgoing packet size, the fragmentation of outgoing packets and the generation and processing of ICMP Packet Too Big errors.

In particular, the NAT64 must generate ICMP Packet Too Big error messages in the case the outgoing packet does not fit in the outgoing MTU and needs to be discarded.

3.2. Filtering and Updating Binding and Session Information

This step updates binding and session information stored in the appropriate tables. This step may also filter incoming packets, if desired.

Irrespectively of the transport protocol used, the NAT64 must silently discard all incoming IPv6 packets containing a source address that contains the Pref64::/n. This is required in order to prevent hairpinning loops as described in the Security Considerations section.

The details of this step depend on the protocol (UDP TCP or ICMP Query).

3.2.1. UDP Session Handling

The state information stored for a UDP session in the UDP session table includes a timer that tracks the remaining lifetime of the UDP session. The NAT64 decrements this timer at regular intervals. When the timer expires, the UDP session is deleted. If all the UDP sessions corresponding to a UDP BIB entry are deleted, then the UDP BIB entry is also deleted (only applies to the case of dynamically created entries).

An IPv6 incoming packet is processed as follows:

> The NAT64 searches for a UDP BIB entry that matches the IPv6 source transport address. If such entry does not exists, a new entry is created. As IPv6 address, the source IPv6 transport address of the packet is included and an IPv4 transport address allocated using the rules defined in Section 3.2.3 is included as IPv4 address.
>
> The NAT64 searches for the session table entry corresponding to the incoming 5-tuple. If no such entry is found, a new entry is created. The information included in the session table is as follows: the IPv6 transport source and destination addresses contained in the received IPv6 packet, the IPv4 transport source address is extracted from the corresponding UDP BIB entry and the

IPv4 transport destination address contains the same port as the

IPv6 destination transport address and the IPv4 address that is algorithmically generated from the IPv6 destination address using the reverse algorithm as specified in Section 3.2.5.

The NAT64 sets or resets the timer in the session table entry to maximum session lifetime. By default, the maximum session lifetime is 5 minutes, but for specific destination ports in the Well-Known port range (0..1023), the NAT64 MAY use a smaller maximum lifetime. The packet is translated and forwarded as described in the following sections.

An IPv4 incoming packet is processed as follows:

The NAT64 searches for a UDP BIB entry that matches the IPv4 destination transport address. If such entry does not exists, the packet is dropped. An ICMP message MAY be sent to the original sender of the packet, unless the discarded packet is itself an ICMP message. The ICMP message, if sent, has a type of 3 (Destination Unreachable).

If the NAT64 filters on its IPv4 interface, then the NAT64 checks to see if the incoming packet is allowed according to the address-dependent filtering rule. To do this, it searches for a session table entry with a source IPv4 transport address equal to the destination IPv4 transport address in the incoming 5-tuple and destination IPv4 address (in the session table entry) equal to the source IPv4 address in the incoming 5-tuple. If such an entry is found (there may be more than one), packet processing continues. Otherwise, the packet is discarded. If the packet is discarded, then an ICMP message MAY be sent to the original sender of the packet, unless the discarded packet is itself an ICMP message. The ICMP message, if sent, has a type of 3 (Destination Unreachable) and a code of 13 (Communication Administratively Prohibited).

The NAT64 searches for the session table entry corresponding to the incoming 5-tuple. If no such entry is found, a new entry is created. The UDP session table entry contains the transport source and destination address contained in the IPv4 packet and the source IPv6 transport address (in the IPv6 --> IPv4 direction) contained in the existing UDP BIB entry. The destination IPv6 transport address contains the same port than the destination IPv4 transport address and the IPv6 representation of the IPv4 address of the destination IPv4 transport address, generated using the algorithm described in Section 3.2.5.

The NAT64 sets or resets the timer in the session table entry to maximum session lifetime. By default, the maximum session

lifetime is 5 minutes, but for specific destination ports in the Well-Known port range (0..1023), the NAT64 MAY use a smaller maximum lifetime.

3.2.2. TCP Session Handling

The state information stored for a TCP session:

Binding:(X',x),(Y',y) <--> (T,t),(Z,z)

Lifetime: is a timer that tracks the remaining lifetime of the UDP session. The NAT64 decrements this timer at regular intervals. When the timer expires, the TCP session is deleted. If all the TCP sessions corresponding to a TCP BIB entry are deleted, then the TCP BIB entry is also deleted (only applies to the case of dynamically created entries).

TCP sessions are expensive, because their inactivity lifetime is set to 2 hours and 4 min (as per [RFC5382]), so it is important that each TCP session table entry corresponds to an existent TCP session. In order to do that, the NAT64 tracks the TCP connection procedure. In this section we describe how the NAT64 does that tracking by describing the state machine.

The states are the following ones:

CLOSED

V4 SYN RCV

V6 SYN RCV

ESTABLISHED

V4 FIN RCV

V6 FIN RCV

V6 FIN + V4 FIN RCV

RST RCV

The state machine for the TCP session processing is depicted next.

```
     +----------------------------+   +-----------------------------+
     |                            |   |                             |
     |                            V   V                             |
     |                 V6       +------+      V4                    |
     |            +----SYN------|CLOSED|-----SYN------+             |
     |            |             +------+              |             |
     |            |                ^                  |             |
     |            |                |4min T.O.         |             |
     |            V                |                  V             |
     |        +-------+         +-------+          +-------+        |
     |        |V6 SYN |         |RST RCV|          |V4 SYN |        |
     |        |  RCV  |         +-------+          |  RCV  |        |
     |        +-------+          |    ^            +-------+        |
     |           |         data pkt   |               |             |
     |           |               |  V4 or V6 RST      |             |
     2Hrs     V4 SYN             V    |           V6 SYN            |
     T.O.        |          +--------------+          |             |
     |           +--------->| ESTABLISHED  |<---------+             |
     +--------------------->|              |                        |
                            +--------------+                        |
                              |           |                         |
                          V4 FIN       V6 FIN                       |
                              |           |                         |
                              V           V                         |
                      +---------+       +----------+                |
                      | V4 FIN  |       |  V6 FIN  |                |
                      +----------+      +-----------+               |
                              |           |                         |
                          V6 FIN       V4 FIN                     4 min
                              |           |                        T.O.
                              V           V                         |
                         +-------------------+                      |
                         | V4 FIN + V6 FIN   |----------------------+
                         +-------------------+
```

After bootstrapping of the NAT64 device, all TCP session are in CLOSED state. We next describe the state information and the transitions.

A TCP segment with the SYN flag set that is received through the IPv6 interface is called a V6 SYN, similarly, V4 SYN, V4 FIN, V6 FIN, V6 FIN + V4 FIN, V6 RST and V4 RST.

*** CLOSED ***

If a V6 SYN is received, the processing is as follows:

The state of the session is moved to V6 SYN RCV.

The NAT64 searches for a TCP BIB entry that matches the IPv6 source transport address.

If such entry does not exists, a new entry is created. As IPv6 address, the source IPv6 transport address of the packet is included and an IPv4 transport address allocated using the rules defined in Section 3.2.3 is included as the IPv4 transport address.

Then a new TCP session entry is created in the TCP session table. The information included in the session table is as follows:

The IPv6 transport source and destination addresses contained in the received V6 SYN packet,

The IPv4 transport source address is extracted from the corresponding TCP BIB entry and, the IPv4 transport destination address contains the same port as the IPv6 destination transport address and the IPv4 address that is algorithmically generated from the IPv6 destination address using the reverse algorithm as specified in Section 3.2.5.

The lifetime of the TCP session table entry is set to 4 min (the transitory connection idle timeout as defined in [RFC5382]).

The packet is translated and forwarded.

If a V4 SYN packet is received, the processing is as follows:

If the security policy requires silently dropping externally initiated TCP connections, then the packet is silently discarded, else,

If the destination transport address contained in the incoming V4 SYN is not in use in the TCP BIB, then the packet is discarded and an ICMP Port Unreachable error (Type 3, Code 3) is sent back to the source of the v4 SYN. The state remains unchanged in CLOSED

If the destination transport address contained in the incoming V4 SYN is in use in the TCP BIB, then

The state is moved to V4 SYN RCV.

A new session table entry is created in the TCP session table, containing the following information:

The transport source and destination address contained in the V4 SYN and,

The source IPv6 transport address (in the IPv6 --> IPv4 direction) contained in the existing TCP BIB entry.

The destination IPv6 transport address contains the same port than the destination IPv4 transport address and the IPv6 representation of the IPv4 address of the destination

IPv4 transport address, generated using the algorithm described in Section 3.2.5.

The lifetime of the entry is set to 6 seconds as per [RFC5382].

The packet is discarded.

For any other IPv6 packet, depending on the security policy other packets MAY be forwarded or MAY be silently discarded. In any case, the state remains unchanged.

For any other IPv4 packet,

If the destination transport address contained in the incoming IPv4 packet is in use in the TCP BIB depending on the security policy other packets MAY be forwarded or MAY be silently discarded. In any case, the state remains unchanged.

If the destination transport address contained in the incoming IPv4 packet is not in use in the TCP BIB the packet is silently discarded.

*** V4 SYN RCV ***

If a V6 SYN is received, then the state is moved to ESTABLISHED. The lifetime of the corresponding TCP session table entry is updated to 2 hours 4 min (the established connection idle timeout as defined in [RFC5382]). The packet is translated and forwarded.

If the lifetime expires, an ICMP Port Unreachable error (Type 3, Code 3) is sent back to the source of the v4 SYN, the session table entry is deleted and, the state is moved to CLOSED.

For any other packet, depending on the security policy other packets MAY be forwarded or MAY be silently discarded. In any case, the

state remains unchanged.

*** V6 SYN RCV ***

If a V4 SYN is received (with or without the ACK flag set), then the state is moved to ESTABLISHED. The timer is updated to 2 hours 4 min (the established connection idle timeout as defined in [RFC5382]). The packet is translated and forwarded.

If the lifetime expires, the session table entry is deleted and the state is moved to CLOSED.

For any other packet, depending on the security policy other packets MAY be forwarded or MAY be silently discarded. In any case, the state remains unchanged.

*** ESTABLISHED ***

If the lifetime expires, the session table entry is deleted and the state is moved to CLOSED.

If a V4 FIN packet is received, the packet is translated and forwarded. The state is moved to V4 FIN RCV.

If a V6 FIN packet is received, the packet is translated and forwarded. The state is moved to V6 FIN RCV.

If a packet is received, the packet is translated and forwarded. The lifetime is set to 2 hours and 4 min. The state remains unchanged as ESTABLISHED.

If the lifetime expires, the session table entry is deleted and the state is moved to CLOSED.

If a V4 RST or a V6 RST packet is received, the packet is translated and forwarded. The lifetime is set to 4 min and state is moved to RST RCV.

*** V4 FIN RCV ***

If a packet is received, the packet is translated and forwarded. The lifetime is set to 2 hours and 4 min. The state remains unchanged as V4 FIN RCV.

If a V6 FIN packet is received, the packet is translated and forwarded. The lifetime is set to 4 min. The state is moved to V6 FIN + V4 FIN RCV.

If the lifetime expires, the session table entry is deleted and the state is moved to CLOSED.

*** V6 FIN RCV ***

If a packet is received, the packet is translated and forwarded. The lifetime is set to 2 hours and 4 min. The state remains unchanged as V6 FIN RCV.

If a V4 FIN packet is received, the packet is translated and forwarded. The lifetime is set to 4 min. The state is moved to V6 FIN + V4 FIN RCV.

If the lifetime expires, the session table entry is deleted and the state is moved to CLOSED.

*** V6 FIN + V4 FIN RCV ***

All packets are translated and forwarded.

If the lifetime expires, the session table entry is deleted and the state is moved to CLOSED.

*** RST RCV ***

If a packet is received, the lifetime is set to 2 hours and 4 min and the state is moved to ESTABLISHED.

If the lifetime expires, the session table entry is deleted and the state is moved to CLOSED.

3.2.3. Rules for allocation of IPv4 transport addresses

If the rules specify that a new BIB entry is created for a source transport address of (S',s), then the NAT64 allocates an IPv4 transport address for this BIB entry as follows:

> If there exists some other BIB entry containing S' as the IPv6 address and mapping it to some IPv4 address T, then use T as the IPv4 address. Otherwise, use any IPv4 address assigned to the IPv4 interface.
>
> If the port s is in the Well-Known port range 0..1023, then allocate a port t from this same range. Otherwise, if the port s is in the range 1024..65535, then allocate a port t from this range. Furthermore, if port s is even, then t must be even, and if port s is odd, then t must be odd.

In all cases, the allocated IPv4 transport address (T,t) MUST NOT be in use in another entry in the same BIB, but MAY be in use in the other BIB.

If it is not possible to allocate an appropriate IPv4 transport address or create a BIB entry for some reason, then the packet is discarded.

3.2.4. ICMP Query Session Handling

The state information stored for an ICMP Query session in the ICMP Query session table includes a timer that tracks the remaining lifetime of the session. The NAT64 decrements this timer at regular intervals. When the timer expires, the session is deleted. If all the sessions corresponding to a ICMP Query BIB entry are deleted, then the ICMP Query BIB entry is also deleted in the case of dynamically created entries.

An incoming ICMPv6 Informational packet is processed as follows:

If the local security policy determines that ICMPv6 Informative packets are to be filtered, the packet is silently discarded. Else, the NAT64 searches for a ICMP Query BIB entry that matches the (IPv6 source address, ICMPv6 Query Id) pair. If such entry does not exist, a new entry is created. As (IPv6 address, ICMPv6 Query Id) pair, the source IPv6 address of the packet and the ICMPv6 Query Identifier are included. The IPv4 address and ICMPv4 Query Identifier values are allocated as follows:

If there exists some other BIB entry containing the same IPv6 address and mapping it to some IPv4 address T, then use T as the IPv4 address. Otherwise, use any IPv4 address assigned to the IPv4 interface.

As ICMPv4 Identifier use any available value i.e. any identifier value for which no other entry exists with the same (IPv4 address, ICMPv4 Query Id) pair.

The NAT64 searches for the session table entry corresponding to the incoming 3-tuple. If no such entry is found, a new entry is created. The information included in the session table is as follows: the IPv6 source and destination addresses contained in the received IPv6 packet, the IPv4 source address, the ICMPv4 Query Id and the ICMPv6 Query Id are extracted from the corresponding ICMP Query BIB entry and the IPv4 destination address is algorithmically generated from the IPv6 destination address using the reverse algorithm as specified in Section 3.2.5.

The NAT64 sets or resets the timer in the session table entry to maximum session lifetime. By default, the maximum session lifetime is 60 seconds. The maximum lifetime value SHOULD be configurable. The packet is translated and forwarded as described in the following sections.

An incoming ICMPv4 Query packet is processed as follows:

The NAT64 searches for a ICMP Query BIB entry that matches the IPv4 destination address and ICMPv4 query Id pair. If such entry does not exists, the packet is dropped. An ICMP message MAY be sent to the original sender of the packet, unless the discarded packet is itself an ICMP message. The ICMP message, if sent, has a type of 3 (Destination Unreachable).

If the NAT64 filters on its IPv4 interface, then the NAT64 checks to see if the incoming packet is allowed according to the address-dependent filtering rule. To do this, it searches for a session table entry with a source IPv4 address and ICMP Query Id pair equal to the destination IPv4 address and ICMP Query Id in the incoming 3-tuple and destination IPv4 address (in the session table entry) equal to the source IPv4 address in the incoming 3-tuple. If such an entry is found (there may be more than one), packet processing continues. Otherwise, the packet is discarded. If the packet is discarded, then an ICMP message MAY be sent to the original sender of the packet, unless the discarded packet is itself an ICMP message. The ICMP message, if sent, has a type of 3 (Destination Unreachable) and a code of 13 (Communication Administratively Prohibited).

The NAT64 searches for the session table entry corresponding to the incoming 3-tuple. If no such entry is found, a new entry is created. The ICMP Query session table entry contains the ICMPv4 Query Identifier, source and destination address contained in the IPv4 packet and the source IPv6 address and the ICMPv6 Query Id (in the IPv6 --> IPv4 direction) contained in the existing ICMP Query BIB entry. The destination IPv6 address contains is the IPv6 representation of the IPv4 address of the destination IPv4 address, generated using the algorithm described in Section 3.2.5.

The NAT64 sets or resets the timer in the session table entry to maximum session lifetime. By default, the maximum session lifetime is 60 seconds. The maximum lifetime value SHOULD be configurable. The packet is translated and forwarded as described in the following sections.

3.2.5. Generation of the IPv6 representations of IPv4 addresses

NAT64 support multiple algorithms for the generation of the IPv6 representation of an IPv4 address. The constraints imposed to the generation algorithms are the following:

The same algorithm to create an IPv6 address from an IPv4 address MUST be used by:

The DNS64 to create the IPv6 address to be returned in the synthetic AAAA RR from the IPv4 address contained in original A RR, and,

The NAT64 to create the IPv6 address to be included in the destination address field of the outgoing IPv6 packets from the IPv4 address included in the destination address field of the incoming IPv4 packet.

The algorithm MUST be reversible, i.e. it MUST be possible to extract the original IPv4 address from the IPv6 representation.

The input for the algorithm MUST be limited to the IPv4 address, the IPv6 prefix (denoted Pref64::/n) used in the IPv6 representations and optionally a set of stable parameters that are configured in the NAT64 (such as fixed string to be used as a suffix).

If we note n the length of the prefix Pref64::/n, then n MUST the less or equal than 96. If a Pref64::/n is configured through any means in the DNS64 (such as manually configured, or other automatic mean not specified in this document), the default algorithm MUST use this prefix. If no prefix is available, the algorithm MUST use the Well-Known prefix (include here the prefix to be assigned by IANA) defined in [I-D.ietf-behave-address-format]

NAT64 MUST support the following algorithms for generating IPv6 representations of IPv4 addresses defined in [I-D.ietf-behave-address-format]:

Zero-Pad And Embed, defined in section 3.2.3 of [I-D.ietf-behave-address-format]

Compensation-Pad And Embed, defined in section of 3.2.4 of [I-D.ietf-behave-address-format]

Embed And Zero-Pad, defined in section of 3.2.5 of [I-D.ietf-behave-address-format]

Preconfigured Mapping Table, defined in section of 3.2.6 of [I-D.ietf-behave-address-format]

The default algorithm used by NAT64 must be Embed and Zero-Pad. While the normative description of the algorithms is provided in [I-D.ietf-behave-address-format].

3.3. Computing the Outgoing Tuple

This step computes the outgoing tuple by translating the addresses and ports or ICMP Query Id in the incoming tuple.

In the text below, a reference to the the "BIB" means either the TCP BIB the UDP BIB or the ICMP Query BIB as appropriate.

NOTE: Not all addresses are translated using the BIB. BIB entries are used to translate IPv6 source transport addresses to IPv4 source transport addresses, and IPv4 destination transport addresses to IPv6 destination transport addresses. They are NOT used to translate IPv6 destination transport addresses to IPv4 destination transport addresses, nor to translate IPv4 source transport addresses to IPv6 source transport addresses. The latter cases are handled applying the algorithmic transformation described in Section 3.2.5. This distinction is important; without it, hairpinning doesn't work correctly.

3.3.1. Computing the outgoing 5-tuple for TCP, UDP and ICMP error messages

The transport protocol in the outgoing 5-tuple is always the same as that in the incoming 5-tuple.

When translating in the IPv6 --> IPv4 direction, let the incoming source and destination transport addresses in the 5-tuple be (S',s) and (D',d) respectively. The outgoing source transport address is computed as follows: the BIB contains a entry (S',s) <--> (T,t), then the outgoing source transport address is (T,t).

The outgoing destination address is computed algorithmically from D' using the address transformation described in Section 3.2.5.

When translating in the IPv4 --> IPv6 direction, let the incoming source and destination transport addresses in the 5-tuple be (S,s) and (D,d) respectively. The outgoing source transport address is computed as follows:

The outgoing source transport address is generated from S using the address transformation algorithm described in Section 3.2.5.

The outgoing destination transport address is computed as follows:

If the BIB contains an entry (X',x) <--> (D,d), then the outgoing destination transport address is (X',x).

Otherwise, discard the packet.

If the rules specify that the packet is discarded, then the NAT64 MAY send an ICMP reply to the original sender, unless the packet being translated contains an ICMP message. The type should be 3 (Destination Unreachable) and the code should be 0 (Network Unreachable in IPv4, and No Route to Destination in IPv6).

3.3.2. Computing the outgoing 3-tuple for ICMP Query messages

When translating in the IPv6 --> IPv4 direction, let the incoming source and destination addresses in the 3-tuple be S' and D' respectively and the ICMPv6 Query Identifier be I1. The outgoing source address is computed as follows: the BIB contains a entry (S',I1) <--> (T,I2), then the outgoing source address is T and the ICMPv4 Query Id is I2.

The outgoing IPv4 destination address is computed algorithmically from D' using the address transformation described in Section 3.2.5.

When translating in the IPv4 --> IPv6 direction, let the incoming source and destination addresses in the 3-tuple be S and D respectively and the ICMPv4 query Id is I2. The outgoing source address is generated from S using the address transformation algorithm described in Section 3.2.5. The outgoing destination address and ICMPv6 Query Id are computed as follows:

> If the BIB contains an entry (X',I1) <--> (D,I2), then the outgoing destination address is X' and the outgoing ICMPv6 Query Id is I1.
>
> Otherwise, discard the packet.

NOTE: Not sure if this applies to ICMP query messages....If the rules specify that the packet is discarded, then the NAT64 MAY send an ICMP reply to the original sender, unless the packet being translated contains an ICMP message. The type should be 3 (Destination Unreachable) and the code should be 0 (Network Unreachable in IPv4, and No Route to Destination in IPv6).

3.4. Translating the Packet

This step translates the packet from IPv6 to IPv4 or vice-versa.

The translation of the packet is as specified in section 3 and section 4 of IP/ICMP Translation Algorithm [I-D.ietf-behave-v6v4-xlate], with the following modifications:

- When translating an IP header (sections 3.1 and 4.1), the source and destination IP address fields are set to the source and destination IP addresses from the outgoing 5-tuple.

- When the protocol following the IP header is TCP or UDP, then the source and destination ports are modified to the source and destination ports from the outgoing 5-tuple. In addition, the TCP or UDP checksum must also be updated to reflect the translated addresses and ports; note that the TCP and UDP checksum covers the pseudo-header which contains the source and destination IP addresses. An algorithm for efficiently updating these checksums is described in [RFC3022].

- When the protocol following the IP header is ICMP and it is an ICMP Query message, the ICMP query Identifier is set to the one of the outgoing 3-tuple.

- When the protocol following the IP header is ICMP (sections 3.4 and 4.4) and it is an ICMP error message, the source and destination transport addresses in the embedded packet are set to the destination and source transport addresses from the outgoing 5-tuple (note the swap of source and destination).

The size of outgoing packets as well and the potential need for fragmentation is done according to the behavior defined in IP/ICMP Translation Algorithm [I-D.ietf-behave-v6v4-xlate]

3.5. Handling Hairpinning

This step handles hairpinning if necessary.

If the destination IP address is an address assigned to the NAT64 itself (i.e., is one of the IPv4 addresses assigned to the IPv4 interface, or is covered by the Pref64::/n prefix assigned to the IPv6 interface), then the packet is a hairpin packet. The outgoing 5-tuple becomes the incoming 5-tuple, and the packet is treated as if it was received on the outgoing interface. Processing of the packet continues at step 2 Section 3.2

4. Application scenarios

In this section, we describe how to apply NAT64/DNS64 to the suitable scenarios described in [I-D.ietf-behave-v6v4-framework] .

4.1. Scenario 1: an IPv6 network to the IPv4 Internet

An IPv6 only network basically has IPv6 hosts (those that are currently available) and because of different reasons including operational simplicity, wants to run those hosts in IPv6 only mode, while still providing access to the IPv4 Internet. The scenario is depicted in the picture below.

```
                               +-----+                 +-------------+
                               |    +------------------+IPv6 Internet+
                               |    |                  +-------------+
     IPv6 host-----------------+ GW |
                               |    |                  +-------------+
                               |    +------------------+IPv4 Internet+
                               +-----+                 +-------------+

     |-------------------------public v6-----------------------------|
     |-------public v6---------|NAT|----------public v4--------------|
```

The proposed NAT64/DNS64 is perfectly suitable for this particular scenario. The deployment of the NAT64/DNS64 would be as follows: The NAT64 function should be located in the GW device that connects the IPv6 site to the IPv4 Internet. The DNS64 functionality can be placed either in the local recursive DNS server or in the local resolver in the hosts.

The proposed NAT64/DNS64 approach satisfies the requirements of this scenario, in particular because it doesn't require any changes to current IPv6 hosts in the site to obtain basic functionality.

4.2. Scenario 3: the IPv6 Internet to an IPv4 network

The scenario of servers using private addresses and being reached from the IPv6 Internet basically includes the cases that for whatever reason the servers cannot be upgraded to IPv6 and they even may not have public IPv4 addresses and it would be useful to allow IPv6 nodes in the IPv6 Internet to reach those servers. This scenario is depicted in the figure below.

```
                                 +-----+
IPv6 Host(s)-------(Internet)-----+ GW +------Private IPv4 Servers
                                 +-----+

|---------public v6---------------|NAT|------private v4----------|
```

This scenario can again be perfectly served by the NAT64 approach. In this case the NAT64 functionality is placed in the GW device connecting the IPv6 Internet to the server's site. In this case, the DNS64 functionality is not required in general since real (i.e. non synthetic) AAAA RRs for the IPv4 servers containing the IPv6 representation of the IPv4 address of the servers can be created. See more discussion about this in [I-D.ietf-behave-dns64]

Again, this scenario is satisfied by the NAT64 since it supports the required functionality without requiring changes in the IPv4 servers nor in the IPv6 clients.

5. Security Considerations

Implications on end-to-end security.

Any protocol that protect IP header information are essentially incompatible with NAT64. So, this implies that end to end IPSec verification will fail when AH is used (both transport and tunnel mode) and when ESP is used in transport mode. This is inherent to any network layer translation mechanism. End-to-end IPsec protection can be restored, using UDP encapsulation as described in [RFC3948].

Filtering.

NAT64 creates binding state using packets flowing from the IPv6 side to the IPv4 side. In accordance with the procedures defined in this document following the guidelines defined in RFC 4787 [RFC4787] a NAT64 must offer "enpoint independent filtering". This means:

> for any IPv6 side packet with source (S'1,s1) and destination (Pref64::D1,d1) that creates an external mapping to (S1,s1), (D1,d1),
>
> for any subsequent external connection to from S'1 to (D2,d2) within a given binding timer window,

(S1,s1) = (S2,s2) for all values of D2,d2

Implementations may also provide support for "Address-Dependent Mapping" and "Address and Port-Dependent Mapping", as also defined in

this document and following the guidelines defined in RFC 4787 [RFC4787].

The security properties however are determined by which packets the NAT64 filter allows in and which it does not. The security properties are determined by the filtering behavior and filtering configuration in the filtering portions of the NAT64, not by the address mapping behavior. For example,

Without filtering - When "endpoint independent filtering" is used in NAT64, once a binding is created in the IPv6 ---> IPv4 direction, packets from any node on the IPv4 side destined to the IPv6 transport address will traverse the NAT64 gateway and be forwarded to the IPv6 transport address that created the binding. However,

With filtering - When "endpoint independent filtering" is used in NAT64, once a binding is created in the IPv6 ---> IPv4 direction, packets from any node on the IPv4 side destined to the IPv6 transport address will first be processed against the filtering rules. If the source IPv4 address is permitted, the packets will be forwarded to the IPv6 transport address. If the source IPv4 address is explicitly denied -- or the default policy is to deny all addresses not explicitly permitted -- then the packet will discarded. A dynamic filter may be employed where by the filter will only allow packets from the IPv4 address to which the original packet that created the binding was sent. This means that only the D IPv4 addresses to which the IPv6 host has initiated connections will be able to reach the IPv6 transport address, and no others. This essentially narrows the effective operation of the NAT64 device to a "Address Dependent" behavior, though not by its mapping behavior, but instead by its filtering behavior.

Attacks to NAT64.

The NAT64 device itself is a potential victim of different type of attacks. In particular, the NAT64 can be a victim of DoS attacks. The NAT64 box has a limited number of resources that can be consumed by attackers creating a DoS attack. The NAT64 has a limited number of IPv4 addresses that it uses to create the bindings. Even though the NAT64 performs address and port translation, it is possible for an attacker to consume all the IPv4 transport addresses by sending IPv6 packets with different source IPv6 transport addresses. It should be noted that this attack can only be launched from the IPv6 side, since IPv4 packets are not used to create binding state. DoS attacks can also affect other limited resources available in the NAT64 such as memory or link capacity. For instance, it is possible

for an attacker to launch a DoS attack to the memory of the NAT64 device by sending fragments that the NAT64 will store for a given period. If the number of fragments is high enough, the memory of the NAT64 could be exhausted. NAT64 devices should implement proper protection against such attacks, for instance allocating a limited amount of memory for fragmented packet storage.

Avoiding hairpinning loops

If the IPv6-only client can guess the IPv4 binding address that will be created, it can use the IPv6 representation of it as source address for creating this binding. Then any packet sent to the binding's IPv4 address will loop in the NAT64.

Consider the following example:

Suppose that the IPv4 pool is 192.0.2.0/24

Then the IPv6-only client sends this to NAT64:

Source: [Pref64::192.0.2.1]:500

Destination: whatever

The NAT64 allocates 192.0.2.1:500 as IPv4 binding address. Now anything sent to 192.0.2.1:500, be it a hairpinned IPv6 packet or an IPv4 packet, will loop.

It should be noted that it is not hard to guess the IPv4 address that will be allocated. First the attacker creates a binding and use e.g. STUN to know your external IPv4. New bindings will always have this address. Then it uses a source port in the range 1-1023. This will increase your chances to 1/512 (since range and parity must be preserved).

In order to address this vulnerability, the NAT64 drops IPv6 packets whose source address is in Pref64::/n.

6. IANA Considerations

This document contains no IANA considerations.

7. Contributors

George Tsirtsis

Qualcomm

tsirtsis@googlemail.com

Greg Lebovitz

Juniper

gregory.ietf@gmail.com

Simon Parreault

Viagenie

simon.perreault@viagenie.ca

8. Acknowledgements

Dave Thaler, Dan Wing, Alberto Garcia-Martinez, Reinaldo Penno, Ranjana Rao and Joao Damas reviewed the document and provided useful comments to improve it.

The content of the draft was improved thanks to discussions with Fred Baker and Jari Arkko.

Marcelo Bagnulo and Iljitsch van Beijnum are partly funded by Trilogy, a research project supported by the European Commission under its Seventh Framework Program.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, January 2005.

[RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.

[RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, April 2009.

[I-D.ietf-behave-dns64]
Bagnulo, M., Sullivan, A., Matthews, P., and I. Beijnum, "DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-dns64-02 (work in progress), October 2009.

[I-D.ietf-behave-v6v4-xlate]
Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", draft-ietf-behave-v6v4-xlate-04 (work in progress), November 2009.

[I-D.ietf-behave-address-format]
Huitema, C., Bao, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", draft-ietf-behave-address-format-01 (work in progress), October 2009.

9.2. Informative References

[RFC2765] Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC2766] Tsirtsis, G. and P. Srisuresh, "Network Address

Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[RFC1858] Ziemba, G., Reed, D., and P. Traina, "Security Considerations for IP Fragment Filtering", RFC 1858, October 1995.

[RFC3128] Miller, I., "Protection Against a Variant of the Tiny

Fragment Attack (RFC 1858)", RFC 3128, June 2001.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC4966] Aoun, C. and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", RFC 4966, July 2007.

[I-D.ietf-mmusic-ice]
Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-19 (work in progress), October 2007.

[RFC3498] Kuhfeld, J., Johnson, J., and M. Thatcher, "Definitions of Managed Objects for Synchronous Optical Network (SONET) Linear Automatic Protection Switching (APS) Architectures", RFC 3498, March 2003.

[I-D.ietf-behave-v6v4-framework]
Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", draft-ietf-behave-v6v4-framework-03 (work in progress), October 2009.

Authors' Addresses

Marcelo Bagnulo
UC3M
Av. Universidad 30
Leganes, Madrid 28911
Spain

Phone: +34-91-6249500
Fax:
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es/marcelo

Philip Matthews
Alcatel-Lucent
600 March Road
Ottawa, Ontario
Canada

Phone: +1 613-592-4343 x224
Fax:
Email: philip_matthews@magma.ca
URI:

Iljitsch van Beijnum
IMDEA Networks
Avda. del Mar Mediterraneo, 22
Leganes, Madrid 28918
Spain

Email: iljitsch@muada.com