Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 6, 2019

K. Dupont

April 4, 2019

NAT64 Handoff
draft-dupont-6man-nat64handoff-00

Abstract

This document describes a NAT64 extension which allows IPv4 hosts to learn the real IP address of hosts which they are communicating with and assume responsibility to maintain the authoritative connection tracking table.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 6, 2019.

Table of Contents

1. Introduction

Due to shortage of IPv4 addresses various models of NAT are seeing widespread usage as a way to reach IPv4-only services. The use of NAT has some drawbacks. This document aims to address two of those drawbacks by providing an extension for NAT64.

Connection tracking entries on the NAT can be expired due to idleness. When doing such expiry the NAT has no way of knowing if the entry was still in use and may cause connections between client and server to be interrupted.

The server doesn't know the real IP address of the client. For the purpose of debugging software bugs and investigating abuse all clients using the same NAT device will appear indistinguishable to the server.

This document provides a NAT64 extension which aims to solve these two drawbacks by allowing the server to assume responsibility to maintain the authoritative version of the connection tracking table and allowing the NAT64 to only maintain a cache in which entries can be purged without causing connections to be interrupted.

2. Terminology

NAT64: A dual-stack host translating TCP and UDP according to [RFC6146] and implementing the NAT64 handoff protocol as specified in this document.

Server: An IPv4 host implementing the server component of

Client: An IPv6 host which initiates communication through a NAT64 capable NAT64.

NAT64 handoff: The UDP based protocol specified in this document to exchange connection tracking between NAT64 and server.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Communication model

The communication model involves a many-to-many relation between clients and NAT64 devices as well as a many-to-many relation between NAT64 devices and servers.

Application data is transmitted and translated according to [RFC6146]. This document adds a control channel between NAT64 and server. There is a separate control channel per NAT64 and server pair. The control channel is uniquely identified by the pair of IPv4 addresses of the NAT64 and server.

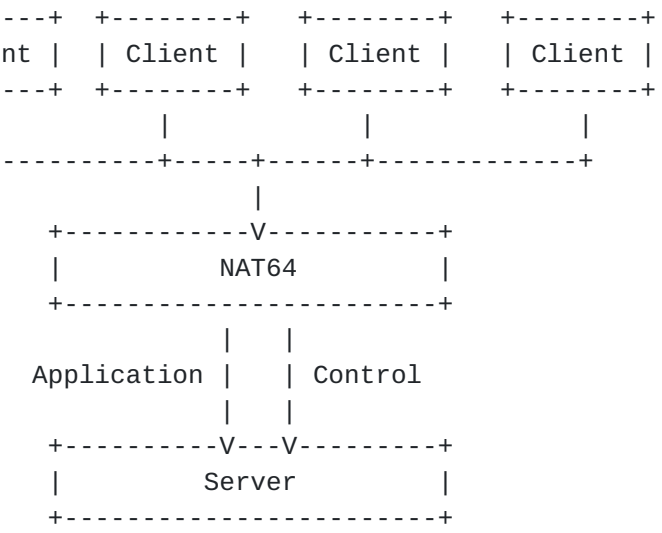


Figure 1: The communication model

This protocol aims to move as much state as possible from the NAT64 to the server and change the remaining state still needed on NAT64 to serve as a cache which can be purged as needed and repopulated from the server.

Protection against IP spoofing is done through the use of tokens in the control channel messages. These tokens are chosen by the NAT64

in order to allow the NAT64 to use a stateless algorithm to generate tokens.

4. Packet formats

The NAT64 handoff packets are UDPv4 packets exchanged between a port on the IPv4 address of the NAT64 and the NAT64 handoff server port (TBD) on the IPv4 address of the server. The UDP payload consist of a sequence of messages.

Message formats are determined by the first nibble of the message. Following message formats are defined:

0 - Reserved to avoid ambiguity in message parsing if NAT64 handoff and Teredo are ever accidentally or deliberately mixed on the same port.

1 - Reserved for future extensions.

2 - Reserved for future extensions.

3 - NAT64 handoff messages.

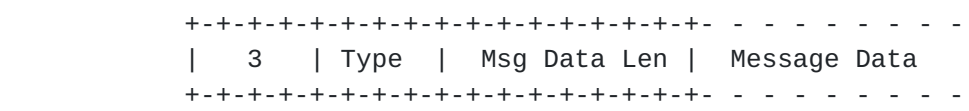
4 - IPv4 packet.

5 - Reserved for future extensions.

6 - IPv6 packet.

7-15 - Reserved for future extensions.

An implementation of NAT64 handoff MUST support packet formats 3, 4, and 6. A packet containing a message of an unknown format MUST be silently dropped.



Type 4-bit identifier of the type of message.

Msg Data Len 8-bit unsigned integer. Length of the Message Data field of this message, in 16 bit units.

Message Data Variable-length field. Message-Type-specific data.

0 - Reserved for future extensions

1 - Protocol identifier

2 - No-op

3 - Invalid

4 - Token

5 - Token rotation

6 - Remote IPv6 address authenticator

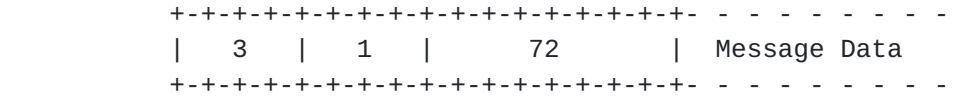
7 - Connection tracking entry

8 - Mapped port hint

9-15 - Reserved for future extensions

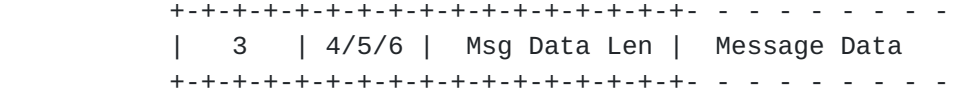
An implementation of the NAT64 handoff server component MUST understand messsage types 4, 5, 6, and 7 and SHOULD understand message types 1 and 8. An implementation of NAT64 with handoff extension MUST understand message types 4, 6, and 7.

An implementation of either component MUST ignore messages of unknown types.



Message data must be the 144 octet string

```
"[[ This is the NAT64 handoff protocol. The client is using "
"IPv6. To know their real IP address you need to use this "
"protocol or support IPv6. ]]"
```

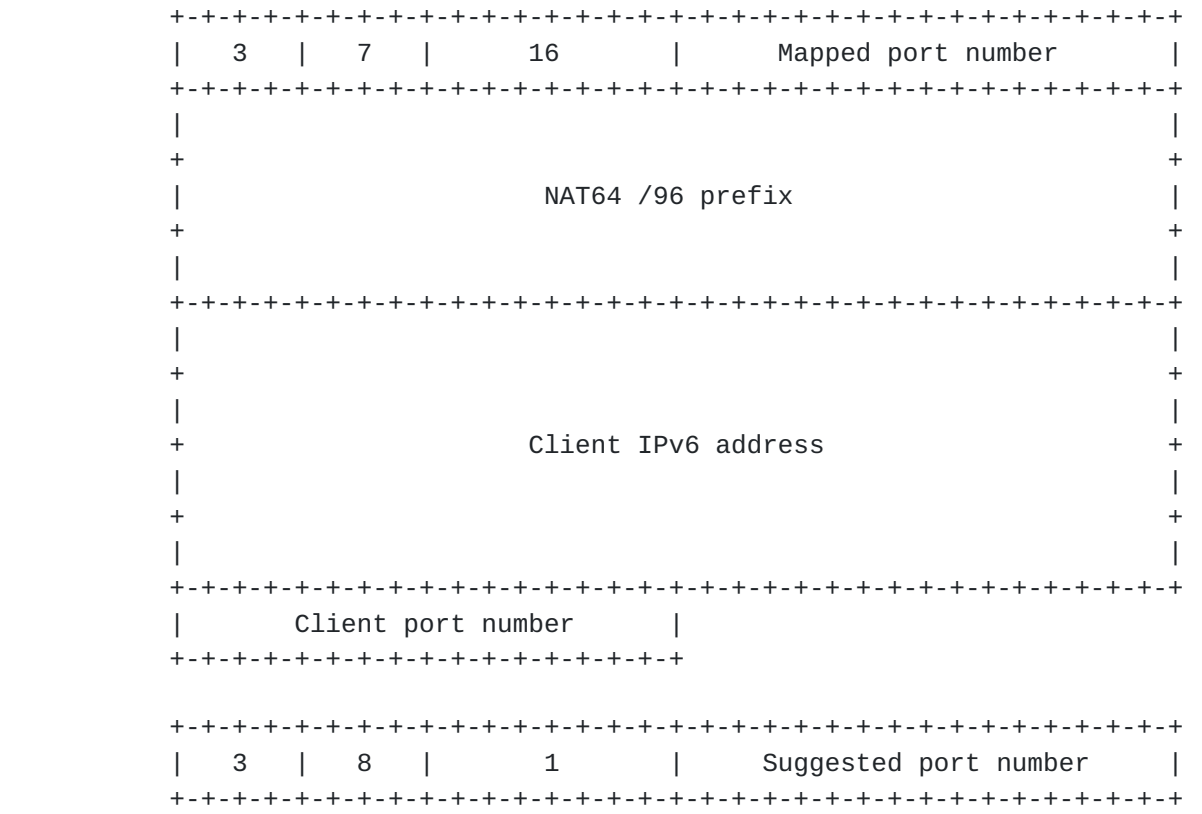


For message types 4, 5, and 6 the NAT64 choose the contents of message data and its length. The length of message data MUST be an even number of octets and at most 510 octets. The message data SHOULD be generated as a Message Authentication Code of at least 16 octets using a secret key known only to the NAT64 itself. The server MUST accept message data of any length from 0 to 510 octets.

For message type 4 message data SHOULD be a MAC of the server IPv4 address computed using a key known only to the NAT64.

For message type 5 message data MUST be computed the same way as message type 4 but using the most recently expired key.

For message type 6 message data should be a MAC of the server IPv4 address and client IPv6 address computed using a key known only to the NAT64.



Not every concatenation of messages form a valid packet. With the exception of the multi-connection-tracking-entry format explained below a packet MUST NOT contain more than one message of each type. When a receiver parses a message containing multiple messages of same type it MAY pick one message of each type to process and ignore the rest. The sender MAY put the messages in a packet in any order.

Any IPv4 or IPv6 packet included in a control channel packet MUST be the last message in the packet. The receiver MAY stop parsing once it sees an IPv4 or IPv6 packet. This also means that a sender is not allowed to include both an IPv4 and an IPv6 packet in the same control channel message.

5. Packet formats from NAT64 to server

All control channel packets from NAT64 to server MUST include a token message (message type 4). And they MUST include at least one of connection tracking entry (message type 7), IPv4 packet, IPv6 packet. Any IPv4 or IPv6 packet sent from NAT64 to server must be the before translation version of the packet.

A packet with a connection tracking entry (messsage type 7), must include a protocol identifier (message type 1) and remote IPv6 address authenticator (message type 6).

A packet with an IPv6 packet must include a remote IPv6 address authenticator (message type 6). It MAY include a mapped port hint (message type 8).

6. Packet formats from server to NAT64

All control channel packets from server to NAT64 MUST satisfy at least one of the following three requirements.

Include a token message (message type 4) and an IPv4 packet before or after translation.

Include a remote IPv6 address authenticator (message type 6) and a post-translation IPv6 packet.

Include a remote IPv6 address authenticator (message type 6) and a connection tracking entry (messsage type 7). Packets of this format MAY include any IPv4 or IPv6 packet before or after translation.

7. Multiple connection tracking entry format

Packets sent from NAT64 to server may follow the format in this section. This is the only packet format which permits multiple messages of the same type.

The packet must contain a protocol identifier (message type 1) and a token (message type 4).

The packet must contain one or more groups of messages in which each group consists of exactly one remote IPv6 address authenticator (message type 6) followed by one or more connection tracking entries (messsage type 7). The remote IPv6 address authenticator must be valid for all connection tracking entries in the group.

The UDP payload MUST NOT exceed 1232 octets.

The packet must not contain an IPv4 or IPv6 packet.

8. NAT64 operation

Upon receiving a packet the NAT64 must first classify the packet as one of the following:

Packet subject to NAT

Control plane packet

ICMP packet

Packet with invalid destination IP

IPv4 packets are classified as follows:

Incoming packets with destination address different from the NAT64 public IPv4 address are considered as invalid destination.

Packets with protocol number 1 are ICMP.

Packets sent from UDP port (TBD) to the client side port (chosen by NAT64) are control packets.

Packets with destination port 80 may be treated as HTTP.

All other packets are subject to NAT.

IPv6 packets are classified as follows:

If destination address is within one of the NAT64 prefixes configured on this NAT64 and the first octet of the embedded IPv4 address is not 0 or 127 the packet is subject to NAT.

Any other destination address is considered as invalid destination.

IPv4 packets subject to NAT are handled as follows:

If it is neither TCP nor UDP forward it over the control connection to the server.

If it matches a cached connection tracking entry from the server perform translation.

If it matches a NAT64 generated connection which has previously been used with this IPv4 address perform translation.

If it matches a NAT64 generated connection which has not been used with this IPv4 address and no valid control packets have ever been received from that server IPv4 address the NAT64 can choose between performing translation and sending the packet over the control channel to the server.

Otherwise send the packet over the control channel to the server.

IPv6 packets subject to NAT are handled as follows:

If it is neither TCP nor UDP forward it over the control connection to the server.

If it matches a cached connection tracking entry from the server perform translation.

If it matches a NAT64 generated connection entry which has previously been used with this IPv4 address perform translation.

If a valid control packet has previously been received from this server IPv4 address forward it over the control connection to the server.

If no NAT64 generated connection tracking entry exists for this source IPv6/port create an entry. Record that the entry (new or existing) has been used with this server IPv4 address. Perform translation.

ICMP error messages containing a (truncated) IPv4 error are handled as follows:

If the ICMP packet has invalid checksum it is silently dropped.

If the embedded packet source address does not match the NAT64 public IPv4 address, the ICMP error is silently dropped.

If the embedded IP payload is an ICMP packet, handle the packet according to [RFC0792] considering the NAT64 itself to be the final destination of the packet.

If the embedded packet is a UDP packet from the client side port (chosen by NAT64) to the server side port (TBD) it's considered to be an undelivered control packet and is silently dropped. If the origin IP of the ICMP error matches the destination IP of the inner IP packet, and the ICMP error has type 3 and code 3, and the payload contains a valid token, the NAT64 must consider the handoff server to be down. The NAT64 must switch back to generating connection entries as if that server IPv4 address never

supported handoff in the first place. Any previously cached connection entries from that server must be kept in cache and expired as if the server was still responding. Once the server is confirmed to be responding again the still cached connection entries must be sent to the server.

If the embedded IP payload is not TCP or UDP forward the packet over the control connection to the source IP of the embedded IP payload.

If the embedded IP payload matches a cached connection from the server perform translation.

If the embedded IP payload matches a NAT64 generated connection previously used with the destination IP of the embedded IP packet perform translation.

Otherwise forward the packet over the control connection to the source IP of the embedded IP payload.

9. Token validation

When the NAT64 server component receives a packet with an unknown combination of token, NAT64 IPv4 address and port number, the server MUST validate it according to the following algorithm which requires no per-NAT64 state until validation succeeds.

Compute a Message Authentication Code calculated over token, client IPv4 address, and client port. The server may rotate the MAC key 30 seconds after it was last used to send a validation packet. If the control channel packet contains an IPv4 packet and the MAC is found inside of that IPv4 packet the client is authenticated and the IPv4 address, port number, and token are stored by the server. Then proceed with processing the packet as an authenticated packet.

Apply attenuation which is recommended to be implemented as follows: At server start time initialize a counter of packets to process to as two. On recepit of a packet if the counter is zero silently drop the packet and set the counter to 1 or 2 with a 50% probability each. Otherwise decrease the counter and continue processing. The server MAY use a per NAT64 counter if it has previously communicated with that NAT64 but the token is unknown.

Generate an IPv4 validation packet to the NAT64 which will not match any connection tracking entry. That packet MUST include a Message Authentication Code calculated by server over token, client IPv4 address, and client port. The server may rotate the MAC key 30 seconds after it was last used to send a validation packet.

The IPv4 validation packet MAY be formatted as a UDP packet from the NAT64 handoff server port (TBD) to UDP port 9 (discard) formatted as a control channel message containing remote IPv6 address authenticator and a no-next-header IPv6 packet using the previously computed MAC as IPv6 payload.

The NAT64 must treat this as a packet not matching a known connection tracking entry and encapsulate the entire packet in a UDP packet sent back to the server. If the NAT64 implements attenuation against reflection attacks it must parse the received packet as a control channel packet and look for a valid token or remote IPv6 address authenticator, if either of those is found it must not drop the packet.

10. Server operation

When the the server receives a packet with a valid token it is handled according to this section.

When the server receives an IPv6 packet over the control connection it must look for a matching connection tracking entry and if none exists it must create one. When using the public IPv4 address of the NAT64 for the connection entry it must use a port number in the range 1024-65535 avoiding the following port numbers:

3544

NAT64 handoff server port (TBD)

The source port of the current control channel packet.

The server must then either:

Translate the packet to IPv4 and deliver it directly

Translate the packet to IPv4 and return it to the NAT64

Return the connection tracking entry and IPv6 packet to the NAT64

When the server receives an IPv4 packet over the control connection it must look for a matching connection tracking entry. If no matching connection tracking entry is found the server should return the packet to the NAT64 if it is a TCP or UDP packet and otherwise construct an appropriate ICMP error message which it can either deliver directly or send back over the control connection to the NAT64.

If a match is found the server must then either:

Translate the packet to IPv6 and deliver it directly

Translate the packet to IPv6 and return it to the NAT64

Return the connection tracking entry and IPv4 packet to the NAT64

11. Server side NAT

When the server creates new connection tracking entries it can choose between using the public IPv4 address of the NAT64 or an address from the ranges specified in [RFC1918] or [RFC6598].

A minimal server side implementation will always use the public IPv4 address of the NAT64 and never perform NAT itself. This will be limited to TCP and UDP support and will cost an extra roundtrip each time the NAT64 cache needs to be populated. It will be subject to the limitations in choice of port number for connection tracking entries outlined in this protocol. But it still allows a larger number of connections than relying entirely on the NAT64 as it won't be competing against servers on other IPv4 addresses for the same pool of port numbers.

Using a local IP range on the server side has several advantages. But it requires server side NAT which also requires the server to run with the additional privileges needed to create a virtual network interface for this purpose.

Advantages of server side NAT is that there is access to more IPv4 addresses thus a larger pool of available ports. And it is not subject to the requirements that the same port number be used for mappings towards all TCP and UDP ports server side. It also allows translation of all protocols needed by the server, not just TCP and UDP. It also completely avoids the use of cache entries on the NAT64 and the roundtrips needed to populate the cache.

A mixed mode operation is also possible where the public IPv4 address of the NAT64 is used even with protocols not supported by the NAT64 and connections absent from the NAT64 cache. In this mode the NAT64 will tunnel IPv6 packets to the server. The server performs NAT and can either deliver the IPv4 packets directly or tunnel them back to the NAT64.

IPv4 packets from the server are sent to the NAT64 which tunnels them to the server for translation. Once packets have been translated to IPv6 the server can either deliver them directly or tunnel them back to the NAT64.

The extra roundtrip due to all the IPv4 packets needing to go from server to NAT64 and back to the server makes this mode less desirable. The extra roundtrip can be avoided at the cost of very complicated routing rules on the server. Whether to use this operation mode is choice made server side and the complexity of supporting it lies server side. The NAT64 is required to support this operation mode in case any server it communicates with makes use of it. The NAT64 side just needs to support the tunneling required for this.

12. Security Considerations

This protocol addresses one security concern around NAT64 by making it harder for attacks to go through a NAT64 as a way to hide their IP address. Other security considerations regarding NAT64 still apply.

The control channel introudced by this document operates over UDP and as such it needs to protect against reflection attacks. Control channel packets received by the NAT64 all contain either a token or a remote IPv6 address authenticator which can validate their authenticity and spoofed packets can be silently dropped. Control channel packets received by the server are required to be attenuated before processing if they do not contain a known valid token. If attenuation is implemented using the algorithm suggested by this document the attenuation factor will be 60%. That means if packets with spoofed source IP are sent to the server component only 40% of them will generate a response.

The tokens and remote IPv6 address authenticators specified in this document not only serves as protection against reflections but also protect against:

Connection cache poisoning

Using control channel for injecting spoofed packets

IPv4 hosts using the NAT64 to send traffic to IPv6 hosts which did not themselves initiate communication through the NAT64.

Application layer security mechanisms such as those implemented by SSH and TLS will work the same with and without NAT64 handoff.

13. IANA Considerations

One UDP port number for the NAT64 handoff server component needs to be allocated by IANA.

```
Service Name: NAT64 handoff
Transport Protocol(s): UDP
Description: NAT64 handoff
Reference: This document.
Port Number: TBD -- To be assigned by IANA.
```

14. Normative References

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/info/rfc1918>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, DOI 10.17487/RFC6598, April 2012, <https://www.rfc-editor.org/info/rfc6598>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Author's Address

Kasper Dupont
Ellemosevaenget 31
Tranbjerg J. 8310
Denmark

Email: kasperd@ntstm.30.mar.2019.kasperd.dk