IPv4/v6 NAT With Explicit Control (NAT-XC)
draft-moore-nat-xc-02

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79. This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on September 9, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of
Abstract

This document describes a mechanism called NAT-XC (for NAT with Explicit Control) for translating between IPv4 and IPv6. NAT-XC is distinguished from other IPv4/IPv6 translations schemes in that it separates the translation between IPv4 and IPv6 from the management of address bindings for such a translation; and is designed to allow applications to be explicitly aware of, and control, their address bindings. NAT-XC can be used by both IPv4 clients wishing to communicate via IPv6, and IPv6 clients wishing to communicate via IPv4. NAT-XC appears to be usable in a wide variety of scenarios requiring communication across IPv4/IPv6 boundaries.
Table of Contents

1. Introduction .................................................. 5
2. Functional Description ........................................ 8
   2.1. Terminology ............................................. 8
   2.2. Translator .............................................. 10
   2.3. Control Point .......................................... 12
3. Control Protocol .............................................. 14
   3.1. Protocol Design Goals .................................. 14
   3.2. Bindings and Translation ................................ 14
   3.3. Leases .................................................. 17
   3.4. Sending Requests ....................................... 17
   3.5. Security ............................................... 18
   3.6. Framing of requests and responses sent using TCP and TLS over TCP .............. 18
3.7. Protocol Messages .......................................... 18
   3.7.1. Create Binding ..................................... 18
   3.7.2. Renew Lease .......................................... 21
   3.7.3. Delete Binding ...................................... 21
   3.7.4. Change Binding ...................................... 22
   3.7.5. Get Binding List ...................................... 23
   3.7.6. Binding Notification messages ....................... 23
   3.7.7. Keepalives .......................................... 23
3.8. Attributes ................................................. 24
   3.8.1. XOR-"ADDRESS ....................................... 24
   3.8.2. XOR-PIGGYBACK-PACKET ............................... 24
   3.8.3. REQUESTED-TTL ...................................... 24
   3.8.4. CREATE-BINDING-OPTIONS ............................ 24
   3.8.5. CREATE-BINDING-RESPONSE-FLAGS ..................... 25
   3.8.6. BINDING-ID ......................................... 25
   3.8.7. LEASE-ID ............................................ 25
   3.8.8. LEASE-TTL ........................................... 25
   3.8.9. DELETE-BINDING-DELAY ............................... 25
4. Control Point Implementation ................................ 26
   4.1. Application Control Over Bindings ....................... 26
   4.2. Control Point implemented in a Network Library ............ 27
   4.3. Control Router ........................................ 28
   4.4. Use of ALGs - and avoiding unnecessary ALGs ............ 29
5. Inspiration and Related Work ................................ 31
6. Using NAT-XC .................................................. 32
   6.1. Use cases ............................................... 32
7. Security Considerations ...................................... 35
8. IANA Considerations .......................................... 36
9. Informative References ...................................... 37
Author’s Address ................................................. 38
1. Introduction

This document describes a mechanism called NAT-XC (or NAT with Explicit Control) for translating between IPv4 and IPv6, with the following characteristics:

- The translation is explicitly controlled (e.g. its address bindings are created, maintained, and discarded) from a remote location using a well-defined protocol, rather than having the bindings maintained at the point at which translation occurs using traffic analysis and heuristics.

- Such control may be accomplished from any of several points, including from one of the endpoints participating in a conversation, or even (when necessary or desirable) by an application.

- Any party wishing to conduct a conversation across address realm boundaries, may arrange for the translation without knowledge of, or cooperation by, other parties.

- While the most common deployment of NAT-XC is assumed to locate the translation at the periphery of an enterprise network or within the enterprise’s ISP, such translation may occur, via explicit arrangement, at any location on the network which has both public IPv4 and public IPv6 network access.

- An application using the translator to accept inbound traffic from a remote address realm, is able to be informed of its endpoint addresses in that realm, as well as the endpoint addresses of its peers.

This mechanism is believed to have the following benefits:

- Deployability.

This single mechanism is adaptable to suit a wide variety of application configurations, network constraints, and operator requirements. Because the translation can be accomplished at a variety of network locations, operators have a great deal of flexibility as to how they arrange for such translation. The mechanism can accommodate IPv4-only networks, IPv6-only networks, legacy hosts without IPv4 capability, applications written to a dual-stack model, and applications written to an IPv4-only model. The mechanism can also function when the client host is behind a legacy IPv4 NAT.

Because NAT-XC is able to translate packets for either IPv4-only
or IPv6-only clients, it allows either of two hosts wishing to communicate (one using IPv4, the other using IPv6) to make itself accessible to the other.

NAT-XC also avoids the need to upgrade multiple components of the network before any application can communicate across the IPv4/IPv6 boundary. Any of several mechanisms can be used to adapt an application to use a NAT-XC translator; and the NAT-XC translator itself can either be provided locally, or by the network’s ISP, or outsourced to a third party.

- Minimizes the need for ALGs (including DNS ALG).

In many cases, NAT-XC permits applications written to a dual-stack programming model to function as if they had direct access to both native IPv4 and native IPv6 networks. Applications written to a dual-stack model are presumed to be aware of both IPv4 and IPv6, and therefore, to also be aware of details of using higher-level protocols with IPv4 and IPv6 (e.g. address literals, DNS AAAA records, FTP EPRT vs. PORT, and so forth).

However, some applications, such as those written to an IPv4-only model, will not be aware of these differences. Those applications will need protocol translation to be implemented by ALGs in order to effectively interoperate with peers speaking only IPv6. Unfortunately, ALGs, when applied indiscriminately to all traffic, can interfere with the interoperation of applications that do not need ALGs.

The NAT-XC architecture minimizes unnecessary use of ALGs as follows: First, by separating the management of address bindings from the address translation, it becomes possible for the address binding management to be implemented closer to the application. So a network library or network stack that supports NAT-XC can provide a means to enable or disable ALGs on a per-application basis. Second, the NAT-XC architecture allows the bindings to be managed by the application itself, pre-empting any binding management that might otherwise be provided by lower layers. Third, for those cases when the binding control is implemented in a Control Router, the NAT-XC Control Protocol has the provision to explicitly disable ALGs. This makes it possible for an application to disable ALGs that might have otherwise interfered with its interoperation.

- Provides a predictable programming environment for applications.

With NAT-XC it is possible for application vendors and authors to ship a single application binary that will work correctly across
IPv4/IPv6 realm boundaries in a wide range of customer environments, ranging anywhere from a dual-stack host with access to both public IPv4 and public IPv6, to a IPv4-only host running on an IPv4-only network located behind a legacy NAT. Such an application would be able to use a NAT-XC translator provided by the customer’s network if one existed, or be explicitly configured to use a remote NAT-XC translator with which the operator had arranged to use.

- Separates network security policy from address translation.

An application request for an address binding can be refused with an error indicating that such communication is a violation of local policy. This provides a means for applications to be aware of the difference between security policy, and the limitations imposed by a traditional NAT. Such an application can then report failures due to security policy to its operator or user (and the NAT-XC translator can also report failed requests), while continuing to work around other network limitations or problems that are not policy related.

- Encourages a desirable end-state for the Internet.

NAT-XC is designed to ease the transition to an Internet in which IPv6 network access is sufficient for a host in order to reach all other hosts of interest (when not prohibited by network policy). In the near term, NAT-XC allows applications to communicate across the IPv4/IPv6 boundary without requiring major changes to the hosts and networks on which they reside. In the long term, NAT-XC allows applications to operate on an IPv6-only network even if they still need to occasionally communicate with hosts using IPv4. NAT-XC thus reduces the near-term burden of transition, while still permitting cross-realm operation, and allows changes to a network’s infrastructure to be decoupled from IPv6 transition and deferred until a later date.

It appears that, in a future Internet where NAT-XC were widely supported by software, the greatest functionality with the least overhead would be achieved by a configuration where most applications were written to a dual-stack model, and where most enterprise networks were IPv6-only. Other configurations could be similarly functional, but have greater overhead. It is assumed that networks would eventually migrate to the configuration with lower operational cost.
2. Functional Description

2.1. Terminology

NAT-XC defines a Translator, which mediates between a Client Application and one or more other Address Realms to which the Client Application lacks direct access. The translation allows the Client Application to communicate with a Peer Application. The Translator is controlled by a Control Protocol. Control Protocol messages are exchanged between the Translator and the Control Point. The Control Point is responsible for establishing and maintaining the bindings used by an application. The Control Point for a particular binding may be located at any one of several locations along the signal path between the Client Application and the Translator, or at the Client Application itself.

(Note that the use of the term Client Application does not imply that the application has a client role in the sense of the client-server model. The Client Application may originate outbound connections or accept inbound connections, or both.)

Control Protocol messages sent by a Control Point are addressed to a Control Address and Port (CAP) assigned to the Translator. Such packets are not translated, but are used to control Translator operation.
The signal path between the Client Application to Peer Application is shown in Figure 1. The path taken by a packet sent by the Client Application to the Peer Application is described as follows:

- The Client Application emits packets from the Client Host with a Private Client Address (PrCA) as the IP source address and a Private Client Port (PrCP) as the TCP or UDP source port. These packets are sent to an IP destination address called the Local Translator Address (LTA) and a TCP or UDP destination port called the Local Translator Port (LTP). Note that the triple consisting of (PrCA, LTA, LTP) is different for each Peer Address and Peer Port with which the Client Application communicates.

- Since the NAT-XC Protocol is designed to permit use of a legacy IPv4 NAT between the Client Host and the Translator, an IP packet sent to a Translator by a Client Host may arrive at the Translator with a different source address and port than the ones originally specified by the Client Host. The source address and port appearing in the packet as it arrives at the translator are known as the Public Client Address (PuCA) and Public Client Port (PuCP).
(The destination address and port of IP packets sent to the Translator from the Client Host must be the same as originated by the Client Host.)

- When a packet from a Client Host is received by the Translator, it determines whether there is a Binding established for that particular PuCA and PuCP and LTA and LTP. If so, it will translate the incoming IPvX packet into an IPvY packet that is originated from a Remote Translator Address (RTA) and Remote Translator Port (RTP) and sent to a Peer Address (PA) and Peer Port (PP).

The path taken by a packet sent by the Peer Application to the Client Application is similar. It originates with source address PA and source port PP, is sent to the Translator at destination address RTA and destination port RTP. The Translator then looks for a Binding associated with RTA and RTP. If it finds one, it translates the packet into one with source address LTA and source port LTP, with destination address PuCA and destination port PuCP. (In the case where the Client Application is listening for incoming traffic from Peers for which there is no prior Binding, a new LTA and LTP will be assigned for use with a new Peer, and a new Binding will be created specifically for use with that Peer.) The translated packet will then be sent to the Client Host with destination address PrCA and destination port PrCP.

The description above is not intended to forbid the use of administrative controls on communication between endpoints. If so configured, the Translator may refuse to forward traffic between particular endpoint addresses and ports, even when a Binding exists.

Note: the author’s intention is to rewrite this document using the concept of a "transport address" to avoid the need, in most cases, to refer to an address and port.

2.2. Translator

A Translator may be located at any point which has both public IPv4 and public IPv6 network access. One or more public IPv4 addresses and one or more public IPv6 addresses will be routed to the Translator.

A Translator translates between IPv4 and IPv6 packets, in both directions, according to Bindings which have been established. A Binding associates the following with one another:

- Transport Protocol (e.g. UDP, TCP)
o Public Client Address and Port (PuCA, PuCP)

o Local Translator Address and Port (LTA, LTP)

o Remote Translator Address and Port (RTA, RTP)

o Peer Address and Port (PA, PP)

Note that an Address may either be an IPv4 address or an IPv6 address. The Public Client Address and the Local Translator Address associated with a Binding must be in the same address realm. Likewise, the Remote Translator Address and the Peer Address must be in the same address realm. It is generally expected that the Local Translator Address and the Remote Translator Address associated with a Binding will be in different address realms.

In discussions of NAT-XC, "Client" refers to some party for whom access to a remote address realm is needed. A Translator may serve IPv4 clients (providing them with access to the public IPv6 network), IPv6 clients (providing them with access to the public IPv4 network), or both. It is possible for both ends of a conversation to be Clients of the same Translator.

For each realm for which it provides services to clients, a Translator has a Control Address and Port (CAP) to which Control Protocol messages may be sent. Note that a Translator may have more than one CAP. It is anticipated that a well-known address and port will be requested from IANA for use with the NAT-XC Control Protocol as the default CAP, as this will allow the use of NAT-XC without site-specific configuration. However, a Translator may accept Control Protocol messages at any address and port at which it can receive packets, and a Control Point may be explicitly configured to use a particular CAP.

Unlike traditional NAT devices, the Translator does not act as the default router for any address realm. The Translator MAY appear to the network as a router in either or both of the public IPv4 and public IPv6 address realms, but packets sent to the Translator from the Client Host or a Peer Host are sent directly to an IP address assigned to the Translator. Similarly, there is no "inside" or "outside" to a NAT-XC Translator, nor even the notion of "sides" in the definition of the Translator. Client-originated traffic is distinguished from Peer-originated traffic via the destination address and port. i.e. the Translator designates certain address and port combinations to be used as the destination of Client-originated traffic. Packets arriving at these address and port combinations which were not originated by a Client will not be translated or forwarded.
2.3. Control Point

A Control Point is the point from which Bindings are requested and managed. Depending on circumstances, the Control Point may exist at a variety of locations between the Client Application and the Translator. Bindings are created and managed via a Control Protocol. Binding requests are sent by the Control Point to a Control Address and Port (CAP) on the Translator.

The following examples illustrate different locations (i.e. Control Points) from which Translator Bindings may be managed:

- An Application may be explicitly coded to generate NAT-XC Control Messages, or it may be statically linked to a library which generates NAT-XC Control Messages as part of its implementation of network access. In either case the Application is the Control Point.

- An Application may be bound at run time to a library which generates NAT-XC Control Messages as part of its implementation of network access, in which case the library may serve as the Control Point for any application that calls it, and which does not manage its own bindings. (also known as "Bump in the API" or "BitA").

- Support for NAT-XC may be included in the network stack of the host platform. In this case the network stack is the Control Point for any application that uses it, and whose bindings are not managed either by the application itself or by a library called by the application. (also known as "Bump in the Stack" or "BitS").

- If there is no Control Point closer to the Client Application, a Control Router located in the signal path between the Client Host and the Translator may serve as the Control Point. Unlike other kinds of Control Points, the Control Router appears to the network as a router. Such a Control Router infers bindings based on traffic analysis and heuristics, in a manner similar to legacy NAT devices. (Such a Control Router may also implement a DNS ALG or other ALGs to accommodate IPv4-only hosts or applications not written to a dual stack model. The Control Router configuration is thus considered a "last resort" mechanism, and it should be used sparingly.) (NB: I’m looking for a better name than Control Router for this.)

- For legacy "server" Applications in the "client-server" sense (that is, Applications that accept inbound traffic) it is possible for a separate process to manage one or more Bindings in the Translator so that traffic sent to a particular Remote Translator Address and Port will be forwarded to a Private Client Address and
Port. This allows such "server" Applications to accept traffic from other address realms.

It is possible for more than one of these mechanisms to be in place. For instance, an Application which is NAT-XC aware may run on a network stack which is also NAT-XC aware, and there may also be a Control Router between that Host and the Translator. In this case the Control Point that is closest to the Client Application is the one that controls the Bindings for that Application.

There are tradeoffs associated with different locations for Control Points. In particular, a Control Router arrangement requires explicit configuration to establish a binding that listens for traffic from a remote realm. Also, a Control Router cannot easily distinguish between traffic from dual-stack applications and IPv4-only applications, and so it does not reliably know when to intercept traffic using ALGs that compensate for such legacy applications. On the other hand, the other mechanisms all require that some change be made on each host supporting an application that wishes to communicate across realm boundaries. And a Control Router can be very useful for accommodating an occasional legacy application, host, or network appliance, as long as it is configured so as not to adversely affect other network traffic.
3. Control Protocol

This is an ROUGH sketch of what the Control Protocol for use between a Control Point and a Translator might look like. Many details have not been worked out yet.

3.1. Protocol Design Goals

- Permit operation of most dual-stack applications by intercepting existing API calls.
- Permit applications to explicitly control translation bindings when necessary.
- Permit use of NAT-XC with unmodified dual stack or legacy IPv4-only applications using any of BitA, BitS, or a Control Router.
- Defer control of NAT-XC to the most upstream Control Point in the signal path.
- Allow enterprise networks to avoid per-host configuration, but allow individual host or application operators to use NAT-XC even without local network support.
- Facilitate operation from hosts with IPv4 only (or IPv6 only) stacks.
- Permit operation through legacy IPv4 NAT.
- Support all of the Control Point configurations listed above, including Control Routers, while still permitting applications to disable ALGs implemented by Control Routers.
- Facilitate recovery from loss of state at the Translator or Control Router.

3.2. Bindings and Translation

A Translator has one or more public IPv4 addresses routed to it, and one or more public IPv6 addresses routed to it. Each of those addresses has potentially 2**16 TCP and 2**16 UDP ports which can be used. A Translator MAY be a host which performs other functions and/or provides other services in addition to being a Translator. If so, some of the TCP and/or UDP ports may be reserved for other purposes and not be available to the Translator.

Of the (transport protocol, address, port) combinations available to the Translator, the Translator will mark some of them as for use by
Clients, and others for use by Peers. (Other combinations may be reserved and unavailable for either use). Any (transport protocol, address, port) combination currently used in a Binding must be marked in such a way. The designation of a (transport protocol, address, port) combination as Client or Peer may not be changed while the combination appears in any Binding.

The Translator maintains a set of Bindings which it uses to translate packets from one realm to another. A Binding is a 9-tuple consisting of Transport Protocol (e.g. UDP or TCP), Public Client Address and Port (PuCA, PuCP), Local Translator Address and Port (LTA, LTP), Remote Translator Address and Port (RTA, RTP), and Peer Address and Port (PA, PP). The PA, PP, and LTP parameters of a Binding may be "wildcards" which can match any address or port. A Binding consisting of PA, PP, and LTP which are "wildcards" is used to permit new inbound conversations from potential Peers to a Client. Normally when such a binding exists, the Client Host will be "listening" for traffic at the PrCA and PrCP corresponding to the PuCA and PuCP associated with that binding.

Other information (e.g. lease timeout, binding ID, client ID, access permissions) may also be associated with the Binding, but the details of these are implementation-specific.

For any Transport Protocol, there is at most one unique, one-to-one, bidirectional mapping between a combination of client-side binding parameters (PuCA, PuCP, LTA, LTP) and a combination of peer-side binding parameters (RTA, RTP, PA, PP).

The Client Address and Peer Address SHOULD be from different realms - e.g. either the Client Address IPv4 and the Peer Address is IPv6, or vice versa. The Public Client Address and the Local Translator Address MUST be from the same realm. Similarly, the Remote Translator Address and Peer Address MUST be from the same realm.

NOTE: Translation between public IPv6 addresses is strongly discouraged. Use of this protocol to translate between public IPv4 and private IPv4, or between different private IPv4 realms, is for further study.

Translation between IPv4 and IPv6 is generally as defined in SIIT [RFC2765], except that address mapping is as follows:

- When a packet arrives at the Translator, its transport protocol, IP destination address, and transport protocol destination port are inspected.
  - If the transport protocol is not supported, an appropriate ICMP (v4 or v6) Destination Unreachable message SHOULD be generated in
response.

- If this (transport protocol, address, port) combination is marked for use by Clients, the Translator searches for a Binding matching the transport protocol and (PuCA = source address, PuCP = source port, LTA = IP destination address, LTP = destination port) for the incoming packet.

- If such a Binding is found, the inbound packet is translated to a new packet with (source address = RTA, source port = RTP, destination address = PA, destination port = RTP).

  If no such Binding is found, an ICMP Destination Unreachable message SHOULD be generated.

- If this (transport protocol, address, port) combination is marked for use by Peers, the Translator searches for a Binding matching the transport protocol and (RTA = destination address, RTP = destination port, PA = source address, PP = source port).

  If such a Binding is found, the inbound packet is translated to a new packet with (source address = LTA, source port = LTP, destination address = PuCA, destination port = PuCP).

  If no such Binding is found, the Translator searches for a Binding matching (RTA = destination address, RTP = destination port, PA = "wildcard", PP = "wildcard"). If such a Binding is found, a new Binding is created with the same PuCA, PuCP, LTA, RTA, and RTP as the one matching the inbound packet. The PA and PP of the new binding are the source address and source port, respectively, from the inbound packet. The LTP of the new binding is chosen by the translator from the set of available ports, subject to the constraint that the (transport protocol, LTA, port) are marked for Client use. Finally, the inbound packet is translated according to the newly created binding.

  Note that whenever a new Binding is created, a Binding Information message is sent to the Control Point.

It is possible for both endpoints of a conversation to use the same Translator at the same time, and thus, for the packet to need to be translated twice. It is therefore necessary for the Translator to detect this case. It is assumed that the right thing to do here is to avoid translating the packet between IPv6 and IPv4 (and back again) and instead, just translate the addresses without changing the packet format. This case needs further study.
3.3. Leases

3.4. Sending Requests

Communications between a Control Point and a Translator are accomplished using different mechanisms depending on the nature of the request.

- A Control Channel may be established between a Control Point and a Translator’s Control Address and Port (CAP). The Control Channel uses TLS [RFC5246] over TCP. This channel is used to establish credentials for the authentication of client requests sent over UDP, for Binding Information messages sent from the Translator to the Control Point, and other purposes. The Control Channel is not required to be maintained at all times, and Bindings MUST be maintained for the duration of their leases even if the Control Channel fails for some reason.

- However, due to the requirement that NAT-XC work when a legacy IPv4 NAT exists between the Client Host and the Translator, Bind Request messages for UDP MUST be sent to a Translator’s CAP via UDP from the PuCA and PuCP. This is because the Translator must be able to establish the Binding in terms of the PrCA and PrCP, and these are only known by sending traffic through the legacy IPv4 NAT from the same transport protocol, Client source address, and port that will be used by later traffic between the Client and the Peer.

- In addition, when a Binding Request is issued for a new client-originated conversation by a Control Router, it is necessary for the new Binding to be established before the initial packet is translated. For this reason, the Control Point MAY include a "piggybacked" packet to be translated onto a Binding Request or Renew Binding Request. This facility SHOULD NOT be used by other kinds of Control Points.

Discussion: There is a possibility that some kinds of middleware boxes (e.g. traffic filters) may block TCP connections unless they first see a SYN packet from the host initiating the SYN. If, say, a NAT-XC aware TCP stack were to use piggybacking to send an initial SYN packet while establishing a Binding in the Translator, and the middleware box were placed between the host and the Translator, the middleware box would not see a SYN packet, and might disrupt subsequent traffic from the host.
3.5. Security

As details have obviously not been worked out, the main purpose of this section is to explicitly acknowledge the necessity of designing security into this protocol from day one.

There are many cases (perhaps all of them) where communications between a Control Point and a Translator will need to be authenticated, and perhaps encrypted. At the moment, it is naively assumed that TLS can be profiled to provide adequate Translator-to-Control Point authentication and encryption for the Control Channel. Authentication by the Control Point to the Translator, when needed, can be accomplished either using TLS client certificates or a username/password like mechanism similar to that used with TLS by several application protocols (e.g. POP, IMAP). However, there is a conflict between the goal of providing zero configuration for Control Points and providing the authentication needed to avoid man-in-the-middle attacks over TLS.

For other communications between the Control Point and the Translator, it is (again, naively) assumed that a symmetric encryption key obtained via the Control Channel (and subject to renewal at intervals) can be used to both authenticate and encrypt those communications, in a manner similar to that used by Kerberos.

3.6. Framing of requests and responses sent using TCP and TLS over TCP

Protocol messages sent via UDP have an obvious framing - one request or response per UDP datagram. Protocol messages sent via TCP require framing in order to separate one protocol message from another. For now it is assumed that, when sent over TCP, each request or response message can be prefixed by a 16-bit request or response length in network byte order.

3.7. Protocol Messages

The intention is to define NAT-XC control protocol as a usage of STUN [RFC5389]. It seems appropriate to leverage STUN because there are usage scenarios in which there will be a legacy v4 NAT between the Control Point and the Translator, and STUN is designed to work around some of the payload damage that some NATs are known to cause. The NAT-XC Control Protocol therefore consists of some new STUN methods and some new attributes to be used with those methods.

3.7.1. Create Binding

The CreateBindingRequest message requests the Translator to establish a Binding between the Client Host’s Public Address and Port (PuCA,
PuCP) and a Remote Address and Port available to the Translator.

CreateBindingRequest messages for TCP MUST be sent over the Control Channel to the CAP. The source address and port used by the Control Point to request a TCP Binding MUST be the same as the Private Client Address and Port (PrCA, PrCP) which will be used with that Binding.

CreateBindingRequest messages for UDP MUST be sent over UDP to the CAP. The source address and port used by the Control Port to request a UDP Binding MUST be the same as the Private Client Address and Port (PrCA, PrCP) which will be used with that Binding.

The following attributes apply to CreateBindingRequest messages:

- **XOR-PRIVATE-CLIENT-ADDRESS.** This is used to convey the Private Client Address and Port (PrCA, PrCP). This attribute is REQUIRED for CreateBindingRequest messages.

- **XOR-REMOTE-ADDRESS.** This is used to convey the Remote Translator Address and Port. This attribute is REQUIRED. However if either of the X-Address or X-Port fields of that attribute are set to zero, the Translator may assign any available address or port to that binding.

- **XOR-PEER-ADDRESS.** This is used to convey the Peer Address and Port to the Translator, for the case where the client wishes to communicate with a specific peer. If the XOR-PEER-ADDRESS attribute is included in the CreateBinding request, the Translator assigns a particular Local Translator Address and Port for use when sending to the Peer Address and Port provided by the Control Point, and the Translator will include that Local Translator Address and Port in the XOR-LOCAL-ADDRESS of the response.

  If the XOR-PEER-ADDRESS attribute is omitted, it implies that the Control Point is requesting a binding to permit any peer to send unsolicited traffic to the Remote Translator Address and Port assigned by the Translator. If such a CreateBinding request is honored, the Translator will effectively create a new Binding any time it receives traffic at the Remote Translator Address and Port from a new Peer Address and/or Port. The Binding will associate a Local Translator Address and Port to be assigned by the Translator, and the Public Client Address and Port from which the CreateBinding request was received, with the Peer Address and Port from which the new traffic was received. Whenever such a Binding is created, a Binding Notification message is sent to the Control Point.
A XOR-PIGGYBACK-PACKET attribute MAY be included with a Bind Request. If the Bind Request is successful, this packet will be translated and sent by the Translator just as if it had been sent by the Client Host. The source IP address and source port of the PiggyBack Packet MUST be the same as the PrCA and PrCP, and the destination IP address and port of the PiggyBack Packet. (The Translator MAY accept the Bind Request while refusing to forward the PiggyBack packet, in which case it will return a Status of {TBD} in the CreateBinding Response message).

- REQUESTED-TTL. This attribute is OPTIONAL. If omitted, the Translator will supply a default.

- A CREATE-BINDING-OPTIONS attribute MAY be included to request specific binding options:
  * The DisableALG option is actually for use by Control Routers. A Control Router intercepts traffic sent to the default CAP, processes it, and then forwards the requests to the Translator. This allows the Control Router to act as an authentication proxy between the Client and the Translator, but it also provides a means by which applications can disable ALG behavior provided by the Control Router.
  
  * The DoubleNAT option affects the Bind Response and Binding Information messages in the case where (a) two endpoints are clients of the same Translator, and (b) both endpoints are connected via a translated address. If the DoubleNAT option is TRUE, the Bind Response and Binding Information messages will reflect both layers of address translation, even if the translation is v6 to v6 or v4 to v4. If the DoubleNAT option is FALSE, the Bind Response and Binding Information messages for that Binding will only reflect a single layer of translation. This is believed to be useful when implementing IPv4 socket APIs – so that for instance a getpeername() call on a socket that is connected via the translator will always produce an address of the expected address family.

The CreateBindingResponse message contains the following attributes:

- XOR-PRIVATE-CLIENT-ADDRESS.
- XOR-LOCAL-ADDRESS.
- XOR-REMOTE-ADDRESS.
o XOR-PEER-ADDRESS.

o CREATE-BINDING-RESPONSE-FLAGS.

* The LegacyNATisPresent flag is set to 1 if the Translator detects the presence of a Legacy NAT (i.e. if the Public Client Address and/or Port are different from the values in the XOR-PRIVATE-CLIENT-ADDRESS attribute of the request).

* The ALGisPresent flag is set to 1 by a Control Router (when forwarding a response to a client) if the Control Router is imposing some ALG on the traffic associated with this binding.

o BINDING-ID. The BindingID assigned to this Binding, for use in a subsequent CancelBinding request

o LEASE-ID. The LeaseID assigned to this Binding, for use in a subsequent RenewLease request. If the Control Point is authenticated to the Translator this LeaseID may be the same ID as used for other bindings requested by the same Control Point.

o LEASE-TTL. The new TTL associated with this LeaseID. Note that if any other Bindings are associated with this LeaseID, the new TTL applies to all of them. Each new CreateBindingRequest message from an authenticated Control Point thus serves as an implicit RenewLease request.

3.7.2. Renew Lease

The RenewLeaseRequest message is to be used to renew the lease on one or more Bindings that are already established.

Attributes included with the RenewLeaseRequest message are:

LEASE-ID. This attribute is REQUIRED.

REQUESTED-TTL. This attribute is OPTIONAL.

Attributes included with the RenewLeaseResponse message are:

LEASE-TTL. This specifies the new TTL associated with that lease ID.

3.7.3. Delete Binding

The DeleteBindingRequest message is to be used to cancel a Binding that is already established. The following attributes may be used:
BINDING-ID. This attribute is REQUIRED

DELETE-BINDING-DELAY. This attribute MAY be used to specify the amount of time (in milliseconds) during which the binding will be maintained for existing connections. This is, for example, to allow for TCP FINs and FIN ACKs to continue to traverse the Translator so that both ends will be aware that the connection was cleanly closed. If the Client has round-trip time estimates available for a particular connection, that information can be used to specify an appropriate delay. This attribute is OPTIONAL. If omitted, a default value will be used. Translators SHOULD limit the amount of delay which a client can request, so that the client cannot use this mechanism to specify a longer lease than might otherwise be available.

The DeleteBindingResponse message contains the following attribute:

DELETE-BINDING-DELAY. This is the actual delay assigned by the Translator after which it will delete this binding.

3.7.4. Change Binding

The Change Binding Request is to be used when, for whatever reason, the Client Host has changed its PuCA. (For instance, if its IPv4 DHCP server has changed its address.)

Note that this is of limited applicability for many kinds of Control Points, because a TCP stack that has open TCP connections in terms of the host’s old IP address will not change the local address associated with those connections. However this request may be useful for Control Points implemented within a host’s TCP stack.

Attributes applicable to a ChangeBindingRequest message are:

BINDING-ID. The ID of the existing Binding. REQUIRED.

XOR-PRIVATE-CLIENT-ADDRESS. The new Private Client Address and Port. REQUIRED.

REQUESTED-TTL. The requested TTL for the new Binding. OPTIONAL.

Attributes included in a ChangeBindingResponse message are the same as defined for a CreateBindingResponse message.
3.7.5. Get Binding List

The purpose of this function is to allow a Control Point to request a list of all of the Bindings which it currently has established. This request may be useful, for instance, when the Control Channel has been broken, or in general, to synchronize views between the Control Point and the Translator.

(not yet specified, though it is expected that a parameter will be LEASE-ID, and the request will list all bindings associated with that LEASE-ID.)

3.7.6. Binding Notification messages

Any time a new Binding is established, or a Binding expires, or a Binding is changed, a Binding Notification message is sent to the Control Point by the Translator over the Control Channel. This message is not a response to an explicit request, but is sent asynchronously.

(not yet specified)

3.7.7. Keepalives

When communicating using UDP via a legacy IPv4 NAT, it may be necessary to occasionally send traffic that will maintain the legacy IPv4 NAT’s binding, in a manner similar to that employed by Teredo [RFC4380]. So in order to maintain the part of the communications path of a UDP conversation between the Control Point, through the legacy IPv4 NAT, to the Translator, it may be necessary to send UDP messages between the PuCA,PuCP and LTA,LTP (in either direction) which are NOT translated or forwarded to the Peer. Similarly it may be necessary to send UDP messages from the Translator through the legacy IPv4 NAT to the PuCA, PuCP which are discarded before they reach the Client Application. There needs to be some way to construct a UDP packet which will appear normal to the legacy NAT and be passed through it, but which the Translator can recognize as a packet that should not be forwarded. It is assumed that IP options will not work for this purpose.

One way to do this might be for the Control Point and Translator to choose a random number of sufficient length to be very unlikely to appear in a conversation. Any UDP packet of exactly that length, containing exactly that random number, would be discarded.

This needs further study.

(not yet specified)
3.8. Attributes

3.8.1. XOR-*-ADDRESS

The attributes XOR-PRIVATE-CLIENT-ADDRESS, XOR-PUBLIC-CLIENT-ADDRESS, XOR-LOCAL-ADDRESS, XOR-REMOTE-ADDRESS, and XOR-PEER-ADDRESS are defined as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| x x x x x x x x |    Protocol   |         X-Port                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                X-Address (Variable) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Protocol is an Internet Protocol Number as defined by the IANA Internet Protocol Numbers registry. This is the same value as used in the IPv4 "protocol" field or the IPv6 "xxx" (whatever it’s called) field. Examples are TCP (6), UDP (17), and SCTP (132).

Note that not all Protocol values are useful or appropriate for bindings in a NAT. Also note that this field’s definition differs from the Family field in STUN’s MAPPED-ADDRESS and XOR-MAPPED-ADDRESS attributes. This was done to make NAT-XC more generally applicable.

X-Port and X-Address are as defined in STUN.

3.8.2. XOR-PIGGYBACK-PACKET

This attribute consists of an IP packet which is obscured in such a way as to deter overzealous legacy NATs from modifying patterns within the packet that happen to look like addresses. Details of such obscuration are TBD.

3.8.3. REQUESTED-TTL

This is a single 32 bit unsigned integer in network byte order, specifying the requested TTL in seconds.

3.8.4. CREATE-BINDING-OPTIONS

This attribute consists of one or more 32-bit integers in network byte order. The first integer is used to encode Boolean values as follows:

- Bit Mask 0x01: DisableALG. Directs any Control Router acting as a proxy between the Control Point and a Translator to disable any
ALGs that apply to the endpoint addresses associated with the Binding.

Bit Mask 0x02: DoubleNAT: In case two endpoints are clients of the same Translator, and both endpoints are connected via a translated address, setting this bit to 1 will cause the response to reflect the complete translation being done. If the bit is set to 0, the response will only reflect a single layer of translation.

Other bits of the first integer, and the contents of additional words of this attribute, are TBD. Clients implementing this version of the specification MUST set the additional bits of the first word to zero, and MUST NOT include additional words in this attribute.

3.8.5. CREATE-BINDING-RESPONSE-FLAGS

Like CREATE-BINDING-OPTIONS, this consists of one or more 32-bit integers in network byte order. Bits defined so far include: LegacyNATisPresent (word 0, mask 0x01), and ALGisPresent (word 0, mask 0x02).

3.8.6. BINDING-ID

This consists of a 32-bit unsigned integer in network byte order.

3.8.7. LEASE-ID

This consists of a 32-bit unsigned integer in network byte order.

3.8.8. LEASE-TTL

This consists of a 32-bit unsigned integer in network byte order, defining the TTL of the lease in seconds.

3.8.9. DELETE-BINDING-DELAY

This consists of a 32-bit unsigned integer in network byte order, defining the time in milliseconds during which a binding is requested to be maintained after a DeleteBindingRequest, or the time in milliseconds during which the Translator agrees to maintain the binding when this attribute appears in a DeleteBindingResponse.
4. Control Point Implementation

As stated above, the Control Point may be located at any of several locations in the signal path between the Client Host and the Translator. This section discusses some details of Control Point implementation which are understood at this time.

4.1. Application Control Over Bindings

A NAT-XC aware application may explicitly control its own bindings. This is done by generating NAT-XC protocol messages and sending them to the NAT-XC Translator. For example, an application running on an IPv4-only network (or on an IPv4-only host) may still access IPv6 via a NAT-XC Translator. To establish a TCP connection to an IPv6 host, the application would:

- Establish a TCP over IPv4 connection to the Translator’s Control Address and Port (CAP). Any available source IP address and port can be used, but the application needs to know what they are.
- Authenticate to the Translator (if required). If Authentication is required the application will need to have been supplied with authentication credentials.
- Request a Binding between the source IPv4 address and TCP port used by the application, and the desired Peer IPv6 address and TCP port.
- Read the Bind Response message from the Translator to determine whether the Bind Request was honored.
- If the Bind Request was honored, note the BindingID and LeaseTTL associated with the binding, and arrange for the lease to be renewed as necessary when the TTL expires.
- Establish a new TCP over IPv4 connection from the same source IPv4 address and Port as before, but with a destination IPv4 address and port as returned in the Bind Response issued by the Translator. At this point there is a connection between the local host (using IPv4) and the remote peer (using IPv6).
- Once the binding is no longer needed, free it up by issuing a Delete Binding request.

Similarly, to listen for inbound IPv6 connections, the application would:
Arrange to listen for incoming connections on a TCP socket with a known IPv4 address and TCP port.

Establish a TCP over IPv4 connection to the CAP, using the same IPv4 source address and port as is being listened to.

Request a Binding between that source address and port, and a Translator IPv6 address and port of the application’s choosing. Either the address or port can be specified to be a wildcard, in which case the Translator is free to choose them.

Read the Bind Response message to determine whether the Bind Request was successful.

Arrange for the Binding to be renewed when its TTL expires, for as long as necessary.

At this point the application will be able to receive inbound IPv6 traffic that is sent to the Translator address and port assigned to it. This traffic will be translated and forwarded to the IPv4 address and port on which the application is listening.

When the Binding is no longer needed, free it up by issuing a Delete Binding request.

While few applications are expected to need to control their own bindings, this technique does have some interesting advantages. For instance, it allows an application to communicate with IPv6 peers even if the local host or network do not support IPv6.

4.2. Control Point implemented in a Network Library

One way to manage NAT-XC bindings is to provide a Library which implements the platform’s usual network API. The library would issue NAT-XC requests as necessary to provide the application with the appearance of having access to both the native IPv4 and native IPv6 networks.

There are two ways to implement such a library. One is to provide a "dual stack" API which makes both IPv4 and IPv6 visible to the application as separate networks, even if the underlying host stack or network only supported one of those. With such an library, the application would be able to lookup IPv4 or IPv6 addresses in DNS, request IPv4 or IPv6 connections, etc. using normal API calls. The library would make "real" system calls and invoke the translator as necessary to provide the application with the appearance of having access to both networks.
The other way to implement such a library is to map all IPv6 traffic onto IPv4. This would be used to allow an app written to an IPv4-only programming model to exchange traffic with IPv6 peers. In this case several of the usual "tricks" would be needed to provide the application with the illusion that IPv6-only hosts had IPv4 addresses - DNS ALG to return fake IPv4 addresses for hosts with only AAAA records. An additional layer of address mapping within the library (in addition to that provided by the Translator) would be needed to make this work. All of the usual caveats associated with NAT-PT and similar schemes apply here.

A single library could implement both programming models, but would need external configuration (say via an environment variable) to let it know whether to provide a dual stack programming model or an IPv4-only programming model. The dual stack model should be the default.

It is possible that a NAT-XC aware application might be used with a NAT-XC aware library. There are two cases for this. One is when the application wishes to completely manage all of its Bindings by itself and to not have the library get in the way. It is useful if such an application has a way to "turn off" the library so that networking API calls are handled transparently. One "natural" way to do this might be for the library to recognize if the application establishes a TCP connection with the Translator’s CAP. This works as long as the library’s idea of the Translator’s address is the same as the application’s idea of the address. A less natural way would be for the application to be able to set an environment variable which could then be recognized by the library to mean "don’t intercept networking system calls and let the application manage its own bindings".

The other case is when the application is willing to let the library do the work of maintaining the bindings, but it wants to have visibility into the bindings, lease times, etc. that are being maintained by the library. Such an application might also want to adjust lease times, disable ALGs, etc. For this case it is useful if the library provides additional API calls to give it that visibility. For example, on UNIX, the ioctl(), setsockopt() or getsockopt() functions might be overloaded to allow the application to find out binding information for network sockets. Overloading an existing API function would allow applications to link to that function without the potential of an unresolved symbol error. The application could determine at runtime whether the additional functionality were supported.

4.3. Control Router
4.4. Use of ALGs - and avoiding unnecessary ALGs

It is hoped that in most cases Application Layer Gateways (ALGs) will not be needed. In particular, since NAT-XC can often provide an application with the appearance of direct access to both public IPv4 and public IPv6 networks, the application can know its public addresses (RTA, RTP) and its peer addresses (PA, PP) via the normal API calls (e.g. getlocalname, getpeername). Address referrals, IP address logging, etc., should work fine. In addition, source IP addresses may still be used for access control, but this requires that trust be extended to the Translator and to the entire communications path between the Control Point and the Translator.

However, ALGs will still be needed for some applications, especially those written for an IPv4-only programming model. ALGs MAY therefore be provided at the Control Point. But since ALGs can actually interfere with the operation of applications that don’t need them, it is necessary to provide means to explicitly enable and disable them. For Control Points which implement ALGs, a default setting of "ALGs disabled" is strongly encouraged. In addition, an application may disable ALGs implemented downstream by issuing a Bind request with the disableALGs option set to TRUE.

- For the case of apps that are explicitly aware of NAT-XC and interact directly with the Translator, ALGs should not be an issue. However, an ALG in a downstream Control Router might still interfere with traffic. For instance an FTP ALG would change the addresses in PORT commands, whereas a DNS ALG would alter the nature of DNS queries and return results different than provided by the queried server. While Control Routers SHOULD NOT enable ALGs except for hosts known to need them, the only way that an application can be sure that a Control Router will not apply an ALG is to issue a Bind request with the disableALGs flag set to TRUE.

- For the case of Control Points implemented on the Client Host (BitA, BitS), it SHOULD be possible to explicitly configure whether any particular ALGs can be enabled. Ideally this would be done on a per-application basis. This could be done, say, by setting an environment variable when launching the application, or by marking the application in a particular way that could be recognized by the Control Point.

- There is potential for an ALG implemented in a Control Router to interfere with an application. For instance, a DNS ALG implemented in a Control Router can provide incorrect and misleading results for a dual-stack app. Furthermore a Control Router cannot reliably distinguish between different applications’
traffic nor determine which applications need ALGs and which do not.

A Control Router that implements ALGs SHOULD have them disabled by default, and SHOULD be configurable to enable them on a per-host basis. For instance, it should be possible to enable ALGs for an IPv4-only host and have them be disabled for dual-stack hosts. (It is possible to imagine a small Control Router, designed for use only with a single host, with a switch to turn ALGs needed by v4-only apps either "on" or "off". That would at least allow control of ALGs on a per-host basis.)

However, because per-host configuration can be wrong, and because different applications on the same host may be affected differently by ALGs, it seems necessary to provide a mechanism by which upstream Control Points can disable or bypass ALGs implemented in Control Routers on a per-application basis.
5. Inspiration and Related Work

Ideas for NAT-XC came from various places.

- SOCKS [RFC1928] is a mechanism that was originally designed to permit IPv4 access over a serial line to hosts lacking a network connection. It was later adapted as a means to establish communications through a firewall.

- The author designed a general purpose NAT traversal solution for the NetSolve distributed computing project, which used connection forwarding and was similar to TRT. Like NAT-XC, the NetSolve mechanism was designed to be usable with minimal changes to existing code.

- RSIP [RFC3103] is a mechanism for providing access to the public IPv4 realm from within a private IPv4 realm. NAT-XC is similar, but because IPv4 and IPv6 use different packet formats with different sized addresses, because the two kinds of addresses are separate spaces which do not overlap, and because many applications nowadays are written to handle the two different kinds of addresses explicitly, many of the limitations associated with RSIP do not appear to impact NAT-XC.
6. Using NAT-XC

6.1. Use cases

NAT-XC can be used to facilitate access across IPv4/IPv6 realm boundaries in a variety of cases. Note that the inability of an application to communicate with both IPv4 and IPv6 peers can be due to any of several different factors:

- The application may be written only to an IPv4 programming model,
- The host may lack either an IPv4 or an IPv6 stack,
- The local network may lack support for either IPv4 or IPv6,
- The local network may not provide routing to both the public IPv4 and IPv6 networks, or
- The local network may be behind a legacy IPv4 NAT and use private IPv4 addressing.

NAT-XC was designed to permit cross-realm communications in all of the cases above. e.g.:

- Public IPv4 access from an IPv6-only network.
- Public IPv6 access from an IPv4-only network. (6to4 and Teredo address this problem in a different way, via tunneling rather than address/packet translation.)
- Dual Stack application operating on IPv4-only host, needing access to IPv6.
- Dual Stack application operating on IPv6-only host, needing access to IPv4. (assumed to be rare)
- IPv4-only application talking with public IPv6 hosts. (DNS ALG and perhaps other ALGs required.)
- Applications explicitly aware of NAT-XC.

6.2. "How do I get my applications working across IPv4/IPv6 boundaries?"

This section is intended to illustrate the ways in which ANY of various parties can act to use NAT-XC to ease IPv4/IPv6 transition, independently of one another. This is a contrast to the traditional IPv6 transition model where multiple parties (user, server operator,
network operator, ISPs) ALL have to act to provide IPv6 access.

- If you are the developer of an application, you can:
  
  * modify your application to explicitly support NAT-XC (if provided by the customer), or
  
  * relink your application with a library that intercepts network API calls and makes use of NAT-XC (if provided by the customer), or
  
  * if your application is dynamically linked (i.e. it makes use of a separate library that is loaded at run time to implement network access), you can ship a library that is compatible with NAT-XC as a replacement for the previous one.

  You can even (if you wish) provide a NAT-XC Translator for use by your customers.

- If you are a server operator, you can:
  
  * update your servers’ operating systems to support NAT-XC, or
  
  * for dynamically linked applications, install a NAT-XC aware networked library on your servers.

  You have the option of providing your own NAT-XC Translator or making arrangements with a third party to provide that service.

- If you are a network operator, you can
  
  * make arrangements for your network to have a NAT-XC Translator (either by installing one, or by arrangement with your ISP, or by making arrangement with a third-party Translator and tunneling Control Protocol traffic to that Translator.), and
  
  * optionally, install a Control Router

- If you are an ordinary personal computer user, you can
  
  * upgrade your operating system to support NAT-XC, or
  
  * install a NAT-XC aware dynamic library, or
  
  * upgrade your software to versions that explicitly support NAT-XC, or
* install a NAT-XC Control Router between your computer and the network, and configure it to establish connections with a third-party Translator.
7. Security Considerations

Security considerations are still being determined. The following issues have been identified.

- There is a need for the Translator to be able to require authentication, and to impose access controls on Bindings, especially when the Translator is not provided by the enterprise network or that network’s ISP.

- There is a need to provide encryption for the Control Protocol.

- NAT-XC provides the capability of individual hosts and applications to source traffic from addresses outside the enterprise network and receive traffic sent to addresses to outside the enterprise network. In some cases network operators will want to prevent, or control, such traffic - and in some cases they have a legitimate interest in doing so. This tussle needs to be addressed explicitly in the document.

- More generally, NAT-XC impacts any network that analyzes or filters traffic based on IP address. Locally-provided Translators may log, analyze, or filter traffic based on local policies, and networks MAY attempt to block connections to external Translators. However the Control Channel is encrypted, and nothing prevents an application and an external Translator from agreeing to use a different port for Control Channels. Also, there is no reliable mechanism for distinguishing between Control Channel traffic and other traffic that might be sent over TLS.

- Applications using IP source addresses as authentication tokens, will be extending trust to the Translator and to the entire signal path between the Application and the Translator. Especially when the Translator is located on an external network, this may introduce new opportunities for source address spoofing.
8. IANA Considerations

This document is a long way from being a formal protocol specification, much less a published one. However in the event that this protocol were ever standardized or approved on an experimental basis, IANA would be requested to assign a well-known port for use with NAT-XC, and to assign an IP address which could be used as a default address for use with NAT-XC.
9. Informative References


Author’s Address

Keith Moore
Network Heretics
PO Box 1934
Knoxville, TN 37901
US

Email: moore@network-heretics.com