Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 25, 2007

L. Eggert
P. Sarolahti
R. Denis-Courmont
V. Stirbu
Nokia

June 23, 2007

A Survey of Protocols to Control Network Address Translators and Firewalls
draft-eggert-middlebox-control-survey-00

Status of this Memo

This Internet-Draft will expire on December 25, 2007.

Copyright Notice

Abstract

This document surveys existing protocols for the control of network address translators and firewalls. It includes standards-level

protocols developed by the IETF and other standards organizations as well as protocols designed by individuals.

Table of Contents

1. Introduction

Network address translators (NATs) and firewalls - frequently referred to as "middleboxes" - are a subject of active discussion in the IETF and related development and standardization communities. These devices are not part of the traditional end-to-end Internet architecture, because they usually operate on transport- and higher-layer information inside the network, which is a layering violation in the original architecture, because operating on that information was the exclusive providence of the end-hosts.

However, in practice, NATs have turned out to be necessary to be able to mitigate the IPv4 address space limitations of many network domains. Similarly, because the networking software in many systems has turned out to contain security defects of different kinds, use of firewalls is common to protect the systems from attacks coming from the network. Another motivation for firewalls is to protect against the abuse of possibly scarce of expensive bandwidth, for example, unwanted traffic on wireless links.

A shared disadvantage of NATs and firewalls is that they often encode knowledge about a particular set of higher-layer protocols and applications in order to operate. This practice prevents new types of network protocols or applications from being deployed end-to-end, unless the middleboxes are upgraded or reconfigured as well. For example, a firewall is usually configured with a list of ports for a set of common network applications, preventing introduction of new applications. Furthermore, they commonly only support traditional transport protocols, such as TCP and UDP, preventing the use of other protocols, such as IPsec, IP tunneling, or other transport protocols. In addition, many NATs and firewalls maintain some state for each active transport-layer session that typically needs to be refreshed in constant intervals, and can be initiated only by certain hosts.

To overcome the above-mentioned limitations, the different protocols and applications have needed to adapt to the behavior of NATs and

firewalls. Commonly, new protocols must be encapsulated into UDP packets in order to pass through these devices. Although the UDP header and protocol logic are minimal and do not consume much network capacity, the use of UDP causes other problems, because it is not connection-oriented. Because there are no messages for connection establishment or connection tear-down in UDP, a stateful NAT or firewall needs to monitor ongoing UDP traffic between a source and destination, and in the absence of such traffic assumes that the session has ended, removing the related session state. In practice, the timers for state clean-up have turned out to be short (on the order of seconds), requiring the end hosts to transmit frequent and resource-consuming keep-alive messages to refresh the session state

maintained at middleboxes.

A more advanced method to overcome the limitations caused by NATs or firewalls would be to allow end-hosts to signal their communication characteristics and profiles explicitly to the NATs and firewalls, or alternatively to allow these devices to signal their configuration information to the end-hosts. With such schemes, the number of state-refreshing keep-alives could be significantly reduced, and NATs and firewalls could be made directly aware of the communication characteristics of the end-hosts.

A number of existing protocols have been proposed for this purpose, and new proposals are being prepared. This document aims to support such efforts by surveying existing proposals and by discussing the the benefits and shortcomings of these schemes.

2. Standards-Track Protocols from the IETF

2.1. SOCKS

2.1.1. Protocol Overview

The SOCKS Protocol Version 5 [RFC1928] defines a method for nodes located on an IP network (such as an Intranet with no routing to the Internet) to establish TCP sessions and exchange UDP datagrams with another IP network (usually the global Internet). To that end, a SOCKS server must be located on the boundary of both networks, and SOCKS clients must explicitly request the server to relay their

communication sessions.

When a SOCKS client establishes a TCP session to the remote network, it first connects to the SOCKS server on a well-known TCP port, sending a connection request with optional authentication credentials. The request specifies in which direction the TCP session is to be established, i.e., whether the SOCKS server will act as the active or passive endpoint. The SOCKS server, if it accepts the request, informs the client of the external IP address and TCP port number that it will use. If the SOCKS server acts as the passive endpoint, it sends an additional response once the TCP three-way handshake is completed. The SOCKS server then forwards traffic between the internal and external TCP sessions, until either of them is terminated.

UDP sessions are also initially negotiated via a TCP session to the SOCKS server, in a similar manner. If successful, the client obtains the IP address and UDP port number of a UDP relay server. The relay forwards UDP datagrams between the local and remote networks. When

sending a datagram, the client adds an additional, SOCKS-specific header, which carries the IP address or DNS name of the remote peer and the intended destination UDP port number of the datagram. The relay adds the same header when forwarding datagrams from the remote network back to the client.

2.1.2. Protocol Analysis

The SOCKS protocol makes few assumptions about the network environment it operates in. In particular, there can be any number NAT/firewall devices between the client and the SOCKS server, and there need not be a router between the local and remote networks. It is assumed that the SOCKS server itself can freely exchange TCP and UDP packets with the remote network.

SOCKS supports both IPv4 and IPv6 as well as translation from one to the other. SOCKS only supports UDP and TCP as transport protocols. Conveyance of IP header parameters other than the IP addresses (such as IP options, hop limit, TOS field, etc.) are not defined. SOCKS cannot be used for generic server applications: only one passive TCP session per request is allowed.

The protocol is mature and implementations for clients and servers are widely available. SOCKS is supported in many FTP and HTTP clients. Applications must usually be modified to support SOCKS, but it is also possible to implement SOCKS transparently as a shim layer above the BSD socket API.

SOCKS requires manual configuration on the client. SOCKS server address and optional credentials must be explicitly provisioned.

2.2. NSIS - NAT/Firewall Signaling Layer Protocol

To be completed [I-D.ietf-nsis-nslp-natfw].

2.3. MIDCOM - Managed Objects for Middlebox Communication

To be completed [RFC3303].

2.4. SIMCO - NEC's Simple Middlebox Configuration Protocol

To be completed [RFC4540].

3. Standards-Level Protocols from other Organizations

3.1. UPnP - Internet Gateway Device Standardized Device Control Protocol

3.1.1. Protocol Overview

The Internet Gateway Device (IGD) is an "edge" interconnection device between a residential Local Area Network (LAN) and a Wide Area Network (WAN), providing connectivity to the Internet for the networked devices in the residential network. The IGD could be physically implemented as a dedicated, standalone device or modeled as a set of Universal Plug-and-Play (UPnP) devices and services on a personal computer.

As an UPnP-based protocol, the IGD Standardized Device Control Protocol [UPNP] inherits the features that the UPnP Device

Architecture provides for support zero-configuration, "invisible" networking, and automatic discovery for a breadth of device categories. Any device can dynamically join a network, obtain an IP address, announce its name, convey its capabilities upon request, and learn about the presence and capabilities of other devices. Devices can disconnect from the network automatically without leaving any unwanted state information behind.

The IGD Standardized Device Control Protocol contains a set of devices and services that allow clients (in the UPnP context also called "Control Points") to control the initiation and termination of connections, monitor status and events of connections, or manage host configuration services, e.g., DHCP or Dynamic DNS. Among these services, the "WANIPConnection" service provides the functionality that allows the Control Points to manage the network address translation on the IGD device.

The IGD Standardized Device Control Protocol preserves the ability of non-UPnP devices to initiate and/or share Internet access.

3.1.2. Protocol Analysis

The IGD Standardized Device Control Protocol is intended to be used in unmanaged network environments such as those found typically in residential networks. The residential network can have up to four segments, a limitation inherited from the UPnP Device Architecture, because the TTL value for the link-local multicast discovery messages is four.

By design, the protocol does permit the presence of several residential gateways in the same network and also permits residential gateways to have multiple connections to the Internet. In practice, a lack of routing mechanisms across multiple, simultaneously active

WAN connections on multiple WAN interfaces and related issues caused by multiple, simultaneously active Internet Gateway devices (e.g., default gateway conflict resolution, load balancing or fail over) make scenarios other than that of a single Gateway Device with a single Internet connection difficult to support.

A residential gateway supporting the IGD Standardized Device Control Protocol is able to operate in two modes. In "routed mode", the

gateway shares the routable WAN IP address with the control points on the LAN using NAT. In "bridged" mode, all Ethernet packets from devices on the LAN are bridged to the WAN. In scenarios where the IP address on the WAN interface is not routable, the device can be switched from routed to bridged mode, allowing both the discovery of IGD Standardized Device Control Protocol devices further along the path as well as the use of other NAT hole-punching protocols.

Applications that intend to create port mappings on a residential gateway supporting the IGD Standardized Device Control Protocol need to have embedded control point functionality, enabling them to create port mappings from TCP or UDP port on the external IPv4 address to the corresponding ports on the internal IPv4 addresses.

The protocol is implemented in more than 90% of the consumer routers, although the functionality might not be enabled by default.

4. Other Protocols

4.1. NAT-PMP - NAT Port Mapping Protocol

4.1.1. Protocol Overview

The NAT Port Mapping Protocol [I-D.cheshire-nat-pmp] is a light-weight binary protocol running on top of UDP between client hosts and their IPv4 gateway. Clients can send requests to their first-hop gateway on a well-known UDP port, in order to determine whether NAT-PMP is supported. If that is the case, they will also learn the external IPv4 address of the gateway device.

If the gateway supports NAT-PMP, a host can assume that it is behind a NAT and start sending request for mapping of external TCP or UDP ports on the external IPv4 address. Mappings can be destroyed explicitly. They also automatically expire after a timeout that can be negotiated per mapping, unless refreshed.

Through the use of link-local multicast, the gateway can notify hosts if its external IP changes, and/or if it has rebooted. In the latter case, hosts are expected to recreate any mappings. This procedure

attempts to protect against loss of state at the gateway.

NAT-PMP assumes that there is one and only one NAT along the path, which also has to be the first-hop gateway. A gateway must not enable NAT-PMP if it is does not perform address/port translation.

4.1.2. Protocol Analysis

NAT-PMP is applicable to small, unmanaged, non-routed networks, connecting multiple hosts to the IPv4 Internet through a single public IPv4 address. It does not require any configuration. Support from the gateway can be auto-detected by clients, and the trust model is solely based on network attachment. There is no support for IPv6, nor is there support for transport protocols other than TCP or UDP. Nested NATs and non-NAT'ing firewalls are not supported.

NAT-PMP covers the same use cases as [UPNP], although it is not as widely deployed today. (NAT-PMP is currently mostly implemented by equipment from Apple Computer, Inc.). The specification is recent, but nevertheless mature.

Applications will normally need to be modified to explicitly request port mappings from the operating system, which would then operate the NAT-PMP state machine and message handling. By design, the protocol allows applications to expose services to the outside, so hole-punching could conceivably be done automatically whenever an application listens to a local TCP port (although this would probably have unwelcome security implications).

4.2. STUN - Controlling NAT Bindings using STUN

To be completed [I-D.wing-behave-nat-control-stun-usage].

4.3. RSIP - Realm-Specific IP

To be completed [RFC3103].

4.4. ALD - Application Listener Discovery for IPv6

4.4.1. Protocol Overview

Application Listener Discovery for IPv6 [I-D.woodyatt-ald] is a light-weight, binary protocol to explicitly punch holes through stateful IPv6 firewalls. It uses ICMPv6 for signaling and is auto-configured through an explicit ICMPv6 router advertisement option.

If the gateway supports ALD, a host can request the opening of holes for any incoming packet toward its own IPv6 address, based on one of

multiple possible criteria:

- always
- an IP protocol (excluding IPv6 extension headers)
- for any standard transport protocol, a destination port number
- for IPsec ESP, an SPI value

Holes automatically expire if not refreshed before a negotiated timeout. The gateway can additionally notify hosts through unsolicited advertisements if it has rebooted or lost state.

4.4.2. Protocol Analysis

ALD is aimed at unmanaged IPv6 networks, where it might not be acceptable to pass any unsolicited packets coming from the outside toward any hosts in the internal network. ALD needs support from all IPv6 routers within the network, because clients learn the ALD middlebox address through IPv6 Neighbor Discovery auto-configuration. It is expected that ALD will operate on the router itself in most cases. As with IPv6 Neighbor Discovery, there is no authentication.

The specification is currently a work-in-progress; there is no known deployment to date. Applications would supposedly need slight modifications, similar as with [UPNP] or [I-D.cheshire-nat-pmp].

ALD can handle "pinholes" for any transport protocol, although it is limited to IPv6 networks, and is meant to restore the ability to operate general-purpose servers behind stateful firewalls. It currently does not explicitly support nesting, though ALD middleboxes could probably forward pinholes request in a hierarchical manner (from innermost to outermost device).

4.5. NLS - Network Layer Signaling Transport Layer

To be completed [I-D.shore-nls-fw].

4.6. AFWC - Authorized IP Firewall Control Application

To be completed [I-D.shore-afwc].

5. Security Considerations

This document is a survey of existing protocols and does not raise any new security considerations. The security considerations of the

surveyed protocols are discussed in their respective specifications, at least for protocols developed within the IETF.

6. IANA Considerations

This document raises no IANA considerations.

7. Acknowledgments

The authors would like to thank: Pasi Eronen.

8. Informative References

[I-D.cheshire-nat-pmp]
Cheshire, S., "NAT Port Mapping Protocol (NAT-PMP)", draft-cheshire-nat-pmp-02 (work in progress), October 2006.

[I-D.ietf-nsis-nslp-natfw]
Stiemerling, M., "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", draft-ietf-nsis-nslp-natfw-14 (work in progress), March 2007.

[I-D.shore-afwc]
Shore, M. and D. McGrew, "An Authorized IP Firewall Control Application", draft-shore-afwc-00 (work in progress), September 2006.

[I-D.shore-nls-fw]
Shore, M., "The NLS Firewall Application", draft-shore-nls-fw-00 (work in progress), February 2006.

[I-D.wing-behave-nat-control-stun-usage]
Wing, D. and J. Rosenberg, "Discovering, Querying, and Controlling Firewalls and NATs using STUN", draft-wing-behave-nat-control-stun-usage-02 (work in

progress), June 2007.

[I-D.woodyatt-ald]
Woodyatt, J., "Application Listener Discovery (ALD) for IPv6", draft-woodyatt-ald-01 (work in progress), June 2007.

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928,

March 1996.

[RFC3103] Borella, M., Grabelsky, D., Lo, J., and K. Taniguchi, "Realm Specific IP: Protocol Specification", RFC 3103, October 2001.

[RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.

[RFC4540] Stiemerling, M., Quittek, J., and C. Cadar, "NEC's Simple Middlebox Configuration (SIMCO) Protocol Version 3.0", RFC 4540, May 2006.

[UPNP] UPnP Forum, "Internet Gateway Device (IGD) Standardized Device Control Protocol V 1.0", November 2001.

Authors' Addresses

Lars Eggert
Nokia Research Center
P.O. Box 407
Nokia Group FIN-00045
Finland

Phone: +358 50 4824461
Email: lars.eggert@nokia.com
URI: http://research.nokia.com/people/lars_eggert/

Pasi Sarolahti

Nokia Research Center
P.O. Box 407
Nokia Group FIN-00045
Finland

Phone: +358 50 4876607
Email: pasi.sarolahti@nokia.com
URI: http://www.iki.fi/pasi.sarolahti/

Remi Denis-Courmont
Nokia Technology Platforms
P.O. Box 407
Nokia Group FIN-00045
Finland

Phone: +358 50 4876315
Email: remi.denis-courmont@nokia.com
URI: http://www.remlab.net/

Vlad Stirbu
Nokia Technology Platforms
P.O. Box 407
Nokia Group FIN-00045
Finland

Phone: +358 50 3860572
Email: vlad.stirbu@nokia.com

Full Copyright Statement

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).