INTERNET-DRAFT H. Kitamura
<draft-kitamura-socks-ipv6-01.txt NEC Corporation
Expires in six months 8 February 1999

SOCKSv5 Protocol Extensions for IPv6/IPv4 Communication Environment
<draft-kitamura-socks-ipv6-01.txt

Status of this Memo

Abstract

This document describes three types of extensions of SOCKS Version 5 protocol [RFC1928]. A new address type and a new command for Requests and Replies are introduced. These extensions supplement the insufficient generic functions of the SOCKSv5 protocol.

These extensions enable a SOCKS server to be used as a translator for IPv4 and IPv6 mixed heteregenous communications with ease. In addition, they make each homogeneous IPv4 and IPv6 communication efficient.

Introduction

The SOCKS Version 5 protocol [RFC1928] can deal with IPv6 address type. Only with this specification, however, it is insufficient to support IPv6 based efficient communication environment. Especially, it is difficult to support IPv4 and IPv6 mixed heterogeneous communication environment and to use a SOCKS server as a translator for mixed heterogeneous communication environment.

In this document, three types of extensions of SOCKSv5 protocol are described. A new address type notion and a new command for Requests and Replies are introduced as the extensions.

These extensions enable a SOCKS server to be used as a translator for IPv4 and IPv6 mixed heteregenous communications with ease. In addition, they make each homogeneous IPv4 and IPv6 communication efficient. Because they supplement the insufficient generic functions of the SOCKSv5 protocol.

Extension

* Extension 1 (New address type notion)

As a new address type (ATYP) notion, "ADDRESS ID" is introduced. This extension is closely related with Extension 2 (New command), which is described below.

An ADDRESS ID is an identifier that represents an entry of the address association mapping table between a SOCKS client and a SOCKS server. Typically, the ADDRESS ID represents the servers' internal address association table identifier.

```
 Internal ADDRESS ID Format
     +-----+-----------+
     |CLASS|REALID(key)|
     +-----+-----------+
     |  1  |     3     |
     +-----+-----------+
```

An ADDRESS ID occupies 4 octets. Internally, the first octet is used for a CLASS. The CLASS indicates categories and characteristics of the ADDRESS ID. For the time being, it is reserved and filled with zero. The rest octets are used for the real identifier (REALID) of the ADDRESS ID.

Since the ADDRESS ID dose not include explicit address information, there are potential vulnerabilities. If some SOCKS clients use the same ADDRESS ID that is already used, the SOCKS server may cause

confusion. (It depends on implementation methods of the SOCKS.) Most of the vulnerabilities can be avoided by the difference of the SOCKS clients' IP addresses, but they still exist for the processes at the same host.

In order to avoid such potential problems and to enhance the security of the system, a simple key exchange mechanism is introduced. The SOCKS server provides a REALID as a key to the SOCKS client. It means that the REALID works as both a identifier and a key. A series of the provided REALIDs by the SOCKS server are not simple sequential numbers. The randomness of the REALIDs avoids the vulnerabilities. 3 octets are long enough to realize this mechanism, and long enough to support the number of the addresses to be dealt between the client and the server.

The length of the ADDRESS ID is fixed and shorter than other address types that are specified in the current SOCKSv5 protocol, and the ADDRESS ID shows essential information between the client and the server. So, the introduction of the ADDRESS ID address type makes efficient communications via the SOCKS server. Especially, in case of UDP communications via the SOCKS server, the utilization of the ADDRESS ID address type contributes to shorten the length of the UDP Packet Structure header and to make the filtering procedure efficient.

Since the introduction of the ADDRESS ID conceals the address family types, it becomes easy to enable to relay different address based connections (e.g., between IPv4 and IPv6) at the SOCKS server.

* Extension 2 (New command for Requests and Replies)

As a new command (CMD), "ADDRESS ASSOCIATE" is introduced. This extension is closely related with Extension 1 (New address type notion).

This command is prepared for a SOCKS client to get a ADDRESS ID from a SOCKS server. Followings are the procedure to get the ADDRESS ID.
1. A SOCKS client sends a Request filled with the ADDRESS ASSOCIATE command to a SOCKS server.
2. The server sends a Reply filled with the ADDRESS ID information to the BND.ADDR field.

After the procedure is finished successfully, the client can use the received ADDRESS ID to any ADDR fields instead of other address types (IP V4 address, DOMAINNAME, or IP V6 address) for the Requests and the UDP Packet Structure header.

With this extension, the client's DNS query delegation to the server

can be accomplished explicitly in the SOCKS protocol.

An ADDRESS ASSOCIATE command has different characteristics from other commands. It can be executed as a dedicated command, but also, it is possible to execute the ADDRESS ASSOCIATE command with other commands (CONNECT, BIND, or UDP ASSOCIATE) simultaneously under the special conditions that the client does not require explicit BND.ADDR information from the Replies. With the simultaneous commands execution can reduce the handshake times between the SOCKS client and the server. In order to realize this function, the ADDRESS ASSOCIATE command needs to be assigned to an appropriate bit of the CMD field of the Requests.

In case an ADDRESS ASSOCIATE command is executed with other command simultaneously, the meanings of the REP field of the Replies may become unclear, and the client can not get explicit BND.ADDR information from the Replies, because BND.ADDR field is filled with ADDRESS ID information. (BND.PORT field is filled with normal information.) In case confusion may happen, an orthodox method that the each command is executed as one dedicated function must be taken. Only when the client does not need BND.ADDR information and the meanings of the REP field is clear, this simultaneous commands execution can be taken.

In case of the dedicated ADDRESS ASSOCIATE command Requests, DST.PORT field does not make sense. The meanings of this field is changed and reused. The name of it is changed to ADR.PREF. It shows the preference of the reply address type of the client. In the Extension 3 (Show the preference of the reply address type), the details of this specification are described.

* Extension 3 (Show the preference of the reply address type)

In case the SOCKS server relays different address based connections (e.g., between IPv4 and IPv6), the address type (ATYP) and the bound address (BND.ADDR) of the Replies are important. If the client can not deal with the replied address type, it causes confusion in the client.

In order to avoid this confusion, the client needs to show the preference of the reply address type to the server. There are three methods to realize this feature.

1. Do nothing special

   The client shows nothing special to the server and expects a default reply address type that can be associated naturally. It means to expect the same address (family) type that is used for

the connection between the client and the server. In this method, the client dose not care which address family type is used in the connection between the server and the desired destination.

2. Use FLAG field of the Requests

The client shows the reply address type preference by setting the flag field FLAG of the Requests. An appropriate bit of the FLAG field shows off or on of the preference of the client.

If the appropriate bit is off (0), it is the same case of the "Do nothing special." Default address type is replied.

If the appropriate bit is on (1), the client asks the server to reply the address as the same address (family) type that is used in the connection between the server and the desired destination. In this case, the client must deal with all of the expected address family types.

3. Use DST.PORT field of the dedicated ADDRESS ASSOCIATE Requests

In case of the dedicated ADDRESS ASSOCIATE Requests, the name of the DST.PORT field is changed to ADR.PREF, and it shows the preference of the reply address type.

Since the ADR.PREF has 2 octets, it has a possibility to show complex preference. For the time being, the upper octet of the ADR.PREF is reserved. The lower octet is used to show the show the preference. The format is the same to the ATYP field.

(As an additional function, this mechanism enables the client to realize the deligation of the reverse DNS query, also.)

The appropriate bit of the FLAG has a high priority and can overwrite the preference that is shown by the ADR.PREF field.

Formats

In the following sections, the formats that include the described extensions are shown. Most parts are quoted from the [RFC1928] and the current SOCKS version 5 specification [SOCKSv5]. Since [RFC1928] and [SOCKSv5] explain the meanings of the fields except extensions that are described in this document, they are omitted here.

x marks (instead of o marks) indicate extensions.

Note:

Unless otherwise noted, the decimal numbers appearing in packet-format diagrams represent the length of the corresponding field, in octets. Where a given octet must take on a specific value, the syntax X'hh' is used to denote the value of the single octet in that

field. When the word 'Variable' is used, it indicates that the corresponding field has a variable length defined either by an associated (one or two octet) length field, or by a data type field.

Requests Format

```
        +----+-----+------+------+----------+----------+
        |VER | CMD | FLAG | ATYP | DST.ADDR | DST.PORT |
        +----+-----+------+------+----------+----------+
        | 1  |  1  |  1   |  1   | Variable |    2     |
        +----+-----+------+------+----------+----------+

          o  VER    protocol version:  X'05'
          o  CMD
             o  CONNECT                X'01'
             o  BIND                   X'02'
             o  UDP ASSOCIATE          X'03'
             x  ADDRESS ASSOCIATE      X'08' (bit set)
             x  CONNECT
                   +ADDRESS ASSOCIATE  X'09'
             x  BIND
                   +ADDRESS ASSOCIATE  X'0A'
             x  UDP ASSOCIATE
                   +ADDRESS ASSOCIATE  X'0B'
             o  X'10' to X'7F' IANA ASSIGNED
             o  X'80' to X'FF' RESERVED FOR PRIVATE METHODS
          o FLAG   command dependent flag (defaults to X'00')
             x  Prefer Default address family type
                                       X'00' (off)
             x  Prefer address family type of the Destination
                                       X'10' (on)
          o  ATYP   address type of following address
             o  IP V4 address:         X'01'
             o  DOMAINNAME:            X'03'
             o  IP V6 address:         X'04'
             x  ADDRESS ID:            X'08'
          o  DST.ADDR       desired destination address
          o  DST.PORT desired destination port in network octet
             order

          In case of the dedicated ADDRESS ASSOCIATE Requests:
          x  DST.PORT = ADR.PREF
                show the preference of the reply address type.
                      X'00'(reserved)+ ATYP
             x  IP V4 address:         X'01'
             x  DOMAINNAME:            X'03'
             x  IP V6 address:         X'04'
```

Addressing Format

In an address field (DST.ADDR, BND.ADDR), the ATYP field specifies the type of address contained within the field:

o X'01'

the address is a version-4 IP address, with a length of 4 octets

o X'03'

the address field contains a fully-qualified domain name. The first octet of the address field contains the number of octets of name that follow, there is no terminating NUL octet.

o X'04'

the address is a version-6 IP address, with a length of 16 octets.

x X'08'

the address is a identifier of the servers' internal address association table, with a length of 1 octet.

Replies Format

VER	REP	FLAG	ATYP	BND.ADDR	BND.PORT
1	1	1	1	Variable	2

o VER   protocol version: X'05'

o REP   Reply field:

o X'00' succeeded

o X'01' general SOCKS server failure

o X'02' connection not allowed by ruleset

o X'03' Network unreachable

o X'04' Host unreachable

o X'05' Connection refused

o X'06' TTL expired

o X'07' Command not supported

o X'08' Address type not supported

o X'09' Invalid address

o X'0A' to X'FF' unassigned

o FLAG   command dependent flag

o ATYP   address type of following address

o IP V4 address: X'01'

```
             o  DOMAINNAME:            X'03'
             o  IP V6 address:         X'04'
             x  ADDRESS ID:            X'08'
          o  BND.ADDR       server bound address
          o  BND.PORT       server bound port in network octet order
```

UDP Control Channel Requests Format

RSV	SUB	FLAG	ATYP	ADDR	PORT
1	1	1	1	Variable	2

```
          o  RSV  Reserved X'00'
          o  SUB  Subcommand
               o  INTERFACE DATA: X'01'
          o  FLAG  A subcommand dependent flag (normally X'00')
          o  ATYP   address type of following address
             o  IP V4 address:         X'01'
             o  DOMAINNAME:            X'03'
             o  IP V6 address:         X'04'
             x  ADDRESS ID:            X'08'
          o  ADDR  destination address information
          o  PORT  destination port information
```

UDP Packet Structure Format

FLAG	FRAG	ATYP	DST.ADDR	DST.PORT	DATA
2	1	1	Variable	2	Variable

The fields in the UDP request header are:

```
          o  FLAG   Reserved X'0000'
          o  FRAG   Current fragment number
          o  ATYP   address type of following address
             o  IP V4 address:         X'01'
             o  DOMAINNAME:            X'03'
             o  IP V6 address:         X'04'
             x  ADDRESS ID:            X'08'
          o  DST.ADDR       desired destination address
          o  DST.PORT       desired destination port
          o  DATA     user data
```

Security Considerations

This document describes a protocol for the application-layer traversal of IP network firewalls. The security of such traversal is highly dependent on the particular authentication and encapsulation methods provided in a particular implementation, and selected during negotiation between SOCKS client and SOCKS server.

Careful consideration should be given by the administrator to the selection of authentication methods.

References

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R. Koblas, D., & Jones, L., "SOCKS Protocol V5," RFC1928, April 1996.

[SOCKSv5] VanHeyningen, M, "SOCKS Protocol Version 5," June 1998 currently draft-ietf-aft-socks-pro-v5-03.txt

[IPv6]   S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC2460, December 1998.

Author's Address

Hiroshi Kitamura
NEC Corporation
C&C Media Research Laboratories
1-1, Miyazaki, 4-Chome, Miyamae-ku,
Kawasaki, Kanagawa, 216-8555, JAPAN

Phone: +81 (44) 856-2123
Fax: +81 (44) 856-2230
EMail: kitamura@ccm.cl.nec.co.jp