Socks Protocol Version 5
INTERNET-DRAFT
Expires: In Six Months
<draft-ietf-aft-socks-protocol-v5-02.txt>

M. Leech
M. Ganis
Y. Lee
R. Kuris
D. Koblas
L. Jones

SOCKS Protocol Version 5

Status of this Memo

Acknowledgments

This memo describes a protocol that is an evolution of the previous version of the protocol, version 4 [1]. This new protocol stems from active discussions and prototype implementations. The key contributors are: Marcus Leech: Bell-Northern Research, David Koblas: Independent Consultant, Ying-Da Lee: NEC Systems Laboratory, LaMont Jones: Hewlett-Packard Company, Ron Kuris: Unify Corporation, Matt Ganis: International Business Machines.

1. **Introduction**

The use of network firewalls, systems that effectively isolate an organizations internal network structure from an exterior network, such as the INTERNET is becoming increasingly popular. These firewall systems typically act as application-layer gateways between networks, usually offering controlled TELNET, FTP, and SMTP

access. With the emergence of more sophisticated application layer protocols designed to facilitate global information discovery, there exists a need to provide a general framework for these protocols to transparently and securely traverse a firewall.

There exists, also, a need for strong authentication of such traversal in as fine-grained a manner as is practical. This requirement stems from the realization that client-server relationships emerge between the networks of various organizations, and that such relationships need to be controlled and often strongly authenticated.

The protocol described here is designed to provide a framework for client-server applications in both the TCP and UDP domains to conveniently and securely use the services of a network firewall. The protocol is conceptually a "shim-layer" between the application layer and the transport layer, and as such does not provide network-layer gateway services, such as forwarding of ICMP messages.

2. Existing practice

There currently exists a protocol, SOCKS Version 4, that provides for unsecured firewall traversal for TCP-based client-server applications, including TELNET, FTP and the popular information-discovery protocols such as HTTP, WAIS and GOPHER.

This new protocol extends the SOCKS Version 4 model to include UDP, and extends the framework to include provisions for generalized strong authentication schemes, and extends the addressing scheme to encompass domain-name and V6 IP addresses.

The implementation of the SOCKS protocol typically involves the recompilation or relinking of TCP-based client applications to use the appropriate encapsulation routines in the SOCKS library.

3. Procedure for TCP-based clients

When a TCP-based client wishes to establish a connection to an object that is reachable only via a firewall (such determination is left up to the implementation), it must open a TCP connection to the appropriate SOCKS port on the SOCKS server system. The SOCKS service is conventionally located on TCP port 1080. If the connection request succeeds, the client enters a negotiation for the authentication method to be used, authenticates with the chosen method, then sends a relay request. The SOCKS server evaluates the request, and either establishes the appropriate connection or

denies it.

The client connects to the server, and sends a version identifier/method selection message:

```
                   +----+----------+----------+
                   |VER | NMETHODS | METHODS  |
                   +----+----------+----------+
                   | 1  |    1     | 1 to 255 |
                   +----+----------+----------+
```

The VER field is set to X'05' for this version of the protocol. The NMETHODS field contains the number of method identifier octets that appear in the METHODS field.

The server selects from one of the methods given in METHODS, and sends a METHOD selection message:

```
                         +----+--------+
                         |VER | METHOD |
                         +----+--------+
                         | 1  |   1    |
                         +----+--------+
```

If the selected METHOD is X'FF', none of the methods listed by the client are acceptable, and the client MUST close the connection.

The values currently defined for METHOD are:

- o X'00' NO AUTHENTICATION REQUIRED
- o X'01' GSSAPI
- o X'02' USERNAME/PASSWORD

- o X'03' to X'7F' IANA ASSIGNED

- o X'80' to X'FE' RESERVED FOR PRIVATE METHODS

- o X'FF' NO ACCEPTABLE METHODS

The client and server then enter a method-specific sub-negotiation. Descriptions of the method-dependent sub-negotiations appear in separate drafts.

Developers of new METHOD support for this protocol

should contact IANA for a METHOD number. The ASSIGNED
NUMBERS document should be referred to for a current
list of METHOD numbers and their corresponding proto-
cols.

Compliant implementations MUST support both GSSAPI and
USERNAME/PASSWORD authentication methods.

4. Requests

Once the method-dependent subnegotiation has completed,
the client sends the request details. If the negoti-
ated method includes encapsulation for purposes of
integrity checking and/or confidentiality, these
requests MUST be encapsulated in the method-dependent
encapsulation.

The SOCKS request is formed as follows:

```
   +----+-----+-------+------+----------+----------+
   |VER | CMD |  RSV  | ATYP | DST.ADDR | DST.PORT |
   +----+-----+-------+------+----------+----------+
   | 1  |  1  | X'00' |  1   | Variable |    2     |
   +----+-----+-------+------+----------+----------+
```

Where:

```
     o  VER    protocol version: X'05'
     o  CMD
        o  CONNECT X'01'
        o  BIND X'02'
        o  UDP ASSOCIATE X'03'
        o  UDP DESTROY X'04'
     o  RSV    RESERVED
     o  ATYP   address type of following address
        o  IP V4 address: X'01'
        o  DOMAINNAME: X'03'
        o  IP V6 address: X'04'
     o  DST.ADDR       desired destination address
     o  DST.PORT       desired destination port in network octet order
```

The SOCKS server will typically evaluate the request
based on source and destination addresses, and return
one or more reply messages, as appropriate for the
request type.

5. Addressing

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 DNS-style domain name. The first octet of the address field contains the length of the domain name.

- o X'04'

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

6. Replies

The SOCKS request information is sent by the client as soon as it has established a connection to the SOCKS server, and completed the authentication negotiations. The server evaluates the request, and returns a reply formed as follows:

```
   +----+-----+-------+------+----------+----------+
   |VER | REP |  RSV  | ATYP | BND.ADDR | BND.PORT |
   +----+-----+-------+------+----------+----------+
   | 1  |  1  | X'00' |  1   | Variable |    2     |
   +----+-----+-------+------+----------+----------+
```

Where:

```
     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' to X'FF' unassigned
```

```
          o  RSV    RESERVED
          o  ATYP   address type of following address
             o  IP V4 address: X'01'
             o  DOMAINNAME: X'03'
             o  IP V6 address: X'04'
          o  BND.ADDR       server bound address
          o  BND.PORT       server bound port in network octet order
```

Fields marked RESERVED (RSV) must be set to X'00'.

If the chosen method includes encapsulation for purposes of authentication, integrity and/or confidentiality, the replies are encapsulated in the method-dependent encapsulation.

CONNECT

In the reply to a CONNECT, BND.PORT contains the port number that the server assigned to connect to the target host, while BND.ADDR contains the associated IP address. The supplied BND.ADDR is often different from the IP address that the client uses to reach the SOCKS server, since such servers are often multi-homed. It is expected that the SOCKS server will use DST.ADDR and DST.PORT, and the client-side source address and port in evaluating the CONNECT request.

BIND

The BIND request is used in protocols which require the client to accept connections from the server. FTP is a well-known example, which uses the primary client-to-server connection for commands and status reports, but may use a server-to-client connection for transferring data on demand (e.g. LS, GET, PUT).

It is expected that the client side of an application protocol will use the BIND request only to establish secondary connections after a primary connection is established using CONNECT. In is expected that a SOCKS server will use DST.ADDR and DST.PORT in evaluating the BIND request.

Two replies are sent from the SOCKS server to the client during a BIND operation. The first is sent after the server creates and binds a new socket. The BND.PORT field contains the port number that the SOCKS server assigned to listen for an incoming connection.

The BND.ADDR field contains the associated IP address. The client will typically use these pieces of information to notify (via the primary or control connection) the application server of the rendezvous address. The second reply occurs only after the anticipated incoming connection succeeds or fails.

In the second reply, the BND.PORT and BND.ADDR fields contain the address and port number of the connecting host.

UDP ASSOCIATE

The UDP ASSOCIATE request is used to establish an association within the UDP relay process to handle UDP datagrams. The DST.ADDR field is ignored in a UDP ASSOCIATE request, while the DST.PORT field contains the idle-timeout time, in minutes, that an association is allowed to exist without explicit client-side activity. If DST.PORT is zero, the SOCKS server SHOULD use an administrator-defined default.

In the reply to a UDP ASSOCIATE request, the BND.PORT and BND.ADDR fields indicate the port number/address where the client may send UDP request messages to be relayed. Once a UDP ASSOCIATE request has been processed, the SOCKS client MUST terminate the connection.

UDP DESTROY

The UDP DESTROY request is used to destroy an existing association within the UDP relay process. The DST.ADDR and DST.PORT fields contain the address and port number of the UDP relay process corresponding to the association to destroy. Once a UDP ASSOCIATE request has been processed, the SOCKS client MUST terminate the connection.

When a reply (REP value other than X'00') indicates a failure, the SOCKS server MUST terminate the TCP connection shortly after sending the reply. This must be no more than 10 seconds after detecting the condition that caused a failure.

If the reply code (REP value of X'00') indicates a success, and the request was either a BIND or a CONNECT, the client may now start passing data. If the selected authentication method supports encapsulation for the

purposes of integrity, authentication and/or confidentiality, the data are encapsulated using the method-dependent encapsulation. Similarly, when data arrives at the SOCKS server for the client, the server MUST encapsulate the data as appropriate for the authentication method in use.

7. Procedure for UDP-based clients

A UDP-based client MUST send its datagrams to the UDP relay server at the UDP port indicated by BND.PORT in the reply to the UDP ASSOCIATE request. If the selected authentication method provides encapsulation for the purposes of authenticity, integrity, and/or confidentiality, the datagram MUST be encapsulated using the appropriate encapsulation. Each UDP datagram carries a UDP request header with it:

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

The fields in the UDP request header are:

- o RSV Reserved X'0000'
- o FRAG   Current fragment number
- o ATYP   address type of following addresses:
  - o IP V4 address: X'01'
  - o DOMAINNAME: X'03'
  - o IP V6 address: X'04'
- o DST.ADDR   desired destination address
- o DST.PORT   desired destination port
- o DATA   user data

When a UDP relay server decides to relay a UDP datagram, it does so silently, without any notification to the requesting client. Similarly, it will drop datagrams it cannot or will not relay. When a UDP relay server receives a reply datagram from a remote host, it MUST encapsulate that datagram using the above UDP request header, and any authentication-method-dependent encapsulation.

The UDP relay server MUST acquire from the SOCKS server the expected IP address of the client that will send

datagrams to the BND.PORT given in the reply to UDP ASSOCIATE. It MUST drop any datagrams arriving from any source IP address other than the one recorded for the particular association.

The FRAG field indicates whether or not this datagram is one of a number of fragments. The high-order bit indicates end-of-fragment sequence, while a value of X'00' indicates that this datagram is standalone. Values between 1 and 127 indicate the fragment position within a fragment sequence. Each receiver will have a REASSEMBLY QUEUE and a REASSEMBLY TIMER associated with these fragments. The reassembly queue must be reinitialized and the associated fragments abandoned whenever the REASSEMBLY TIMER expires, or a new datagram arrives carrying a FRAG field whose value is less than the highest FRAG value processed for this fragment sequence. The reassembly timer MUST be no less than 5 seconds. It is recommended that fragmentation be avoided by applications wherever possible.

Implementation of fragmentation is optional; an implementation that does not support fragmentation MUST drop any datagram whose FRAG field is other than X'00'.

The programming interface for a SOCKS-aware UDP MUST report an available buffer space for UDP datagrams that is smaller than the actual space provided by the operating system:

- if ATYP is X'01' - 10+method_dependent octets smaller
- if ATYP is X'03' - 262+method_dependent octets smaller
- if ATYP is X'04' - 20+method_dependent octets smaller

The ASSOCIATION established with a UDP ASSOCIATE request has a specific lifetime. The UDP server SHOULD refresh the lifetime every time a valid datagram arrives from the client at the UDP relay server.

8. 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 adminis-
trator to the selection of authentication methods.

9. References

[1] Koblas, D., "SOCKS", Proceedings: 1992 Usenix Secu-
rity Symposium

Authors Address

Marcus Leech
Bell-Northern Research
P.O. Box 3511, Stn. C,
Ottawa, ON
CANADA K1Y 4H7

Email: mleech@bnr.ca
Phone: (613) 763-9145