

SOCKS Protocol Version 5

Status of this Memo

This document is a submission to the IETF Authenticated Firewall Traversal (AFT) Working Group. Comments are solicited and should be addressed to the working group mailing list (aft@socks.nec.com) or to the editor.

This document is an Internet-Draft. 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 draft documents are 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."

To learn the current status of any Internet-Draft, please check the "lid-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on [ftp.is.co.za](ftp://ftp.is.co.za) (Africa), [nic.nordu.net](ftp://nic.nordu.net) (Europe), [munni.oz.au](ftp://munni.oz.au) (Pacific Rim), [ds.internic.net](ftp://ds.internic.net) (US East Coast), or [ftp.isi.edu](ftp://ftp.isi.edu) (US West Coast).

Distribution of this memo is unlimited

Acknowledgments

This memo describes a protocol that is an evolution of the previous version of the protocol, version 4[SOCKS]. This new protocol stems from active discussions and prototype implementations. The key contributors are:

- o Marcus Leech: Bell-Northern Research
- o David Koblas: Independent Consultant
- o Ying-Da Lee: NEC Systems Laboratory
- o LaMont Jones: Hewlett-Packard Company
- o Ron Kuris: Unify Corporation
- o Matt Ganis: International Business Machines
- o David Blob: NEC USA
- o Wei Lu: NEC USA.
- o William Perry: Aventail
- o Dave Chouinard: Intel

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.

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.

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' CHAP

- o X'04' 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 memos.

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 protocols.

Compliant implementations MUST support CHAP, SHOULD support USERNAME/PASSWORD and MAY support GSSAPI authentication methods.

As with other TCP application data, out of band data is normally proxied to the SOCKS server as out of band data; note that implementations may be limited to handling a single byte of such data at a time. Authentication methods which define some content encapsulation SHOULD define a method-specific mechanism for proxying out of band data.

4. Requests

Once the method-dependent subnegotiation has completed, the client sends the request details. If the negotiated 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 | FLAG | ATYP | DST.ADDR | DST.PORT |
+---+---+---+---+---+---+
| 1  |  1  |  1  |  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 X'04' to X'7F' IANA ASSIGNED

- o X'80' to X'FF' RESERVED FOR PRIVATE METHODS
- o FLAG command dependent flag (defaults to 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'
- 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 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.

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 | FLAG | ATYP | BND.ADDR | BND.PORT |
+---+---+---+---+---+---+
| 1  |  1  |  1  |  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' 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'
- o BND.ADDR server bound address
- o BND.PORT server bound port in network octet order

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. DST.ADDR must be the address of the primary connection's destination. DST.PORT should be the requested port (or 0 for a random, unused port). It is

VanHeyningen

Expires September 1998

[Page 6]

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.

7. UDP procedure

UDP ASSOCIATE requests

The UDP ASSOCIATE request is used to establish an association within the UDP relay process to handle UDP datagrams. The DST.ADDR and DST.PORT fields contain the address and port that the client expects to use to send UDP datagrams on for the association. The server MAY use this information to limit access to the association. If the client is not in possession of the information at the time of the UDP ASSOCIATE, the client MUST use address type X'01' with a port number and address of all zeros.

A UDP association terminates when the TCP connection that the UDP ASSOCIATE request arrived on terminates.

Flag bits in the request and reply are defined as follows:

```
INTERFACE REQUEST X'01'  
USECLIENTSPORT   X'04'
```

If the USECLIENTSPORT bit is set in the flag field of the request, the server SHOULD use interact with the application server using the same port the client used in the request, and set the USECLIENTSPORT bit in the flag field of the reply to acknowledge having done so.

If the INTERFACE REQUEST bit is set in the flag field of the request, the server may indicate its support for this extension by setting this bit in the reply. If both client and server support this feature, the client MAY send interface-request subcommands, described below, during the UDP association.

In the reply to a UDP ASSOCIATE request, the BND.PORT and BND.ADDR fields indicate the port number/address where the client MUST send UDP request messages to be relayed.

Reply Processing

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.

UDP Control Channel

A UDP association terminates when the TCP connection that the UDP ASSOCIATE request arrived on terminates. If the flag negotiation indicated mutual support for it, the client may send INTERFACE-REQUEST commands to learn the external address information for the UDP association with respect to a particular destination.

Such requests are formatted as follows:

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| RSV | SUB | FLAG | ATYP | ADDR      | PORT | SIZE | DATA      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1   | 1   | 1   | 1   | Variable | 2    | 4    | Variable   |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

The fields in the CONTROL CHANNEL packet are:

- 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 addresses:
 - o IP V4 address: X'01'
 - o DOMAINNAME: X'03'
 - o IP V6 address: X'04'
- o ADDR any address information
- o PORT any port information

- o SIZE the size (in octets) of data in network order
- o DATA user data

Replies to INTERFACE DATA commands are structured the same way as ordinary SOCKS replies, as per [section 6](#).

UDP packet structure

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:

```

+-----+-----+-----+-----+-----+-----+
| 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 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

FRAG is currently unused, and reserved for future work to deal with fragmentation.

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 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:

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

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 administrator to the selection of authentication methods.

9. References

- [CHAP] VanHeyningen, M., "Challenge-Handshake Authentication Protocol for SOCKS V5," work in progress.
- [RFC 1928] Leech, M., Ganis, M., Lee, Y., Kuris, R. Koblas, D., & Jones, L., "SOCKS Protocol V5," April 1996.
- [RFC 1929] Leech, M., "Username/Password Authentication for SOCKS V5," March 1996.
- [RFC 1961] McMahon, P., "GSS-API Authentication Method for SOCKS Version 5," June 1996.
- [SOCKS] Koblas, D., "SOCKS", Proceedings: 1992 Usenix Security Symposium.

Author's Address

Marc VanHeyningen
Aventail Corporation
117 South Main Street, Suite 400
Seattle, WA 98104

Phone: +1 (206) 215-1111
Email: marcvh@aventail.com

