                        SOCKS Protocol Version 5

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.

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.

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.

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 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 GSSAPI and SHOULD support
USERNAME/PASSWORD authentication methods.

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 |  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 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 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 |  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' Command not supported
             o X'08' Address type not supported
             o X'09' to X'FF' unassigned
          o RSV   RESERVED
          o ATYP   address type of following address

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.