Internet Area Working Group
Internet-Draft
Intended status: Experimental
Expires: December 30, 2017

V. Olteanu
D. Niculescu
University Politehnica of Bucharest

June 28, 2017

SOCKS Protocol Version 6
draft-olteanu-intarea-socks-6-00

Abstract

The SOCKS protocol is used primarily to proxy TCP connections to arbitrary destinations via the use of a proxy server. Under the latest version of the protocol (version 5), it takes 2 RTTs (or 3, if authentication is used) before data can flow between the client and the server.

This memo proposes SOCKS version 6, which reduces the number of RTTs used, takes full advantage of TCP Fast Open, and adds support for 0-RTT authentication.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Versions 4 and 5 [RFC1928] of the SOCKS protocol were developed two decades ago and are in widespread use for circuit level gateways or as circumvention tools, and enjoy wide support and usage from various software, such as web browsers, SSH clients, and proxifiers. However, their design needs an update in order to take advantage of the new features of transport protocols, such as TCP Fast Open [RFC7413], or to better assist newer transport protocols, such as MPTCP [RFC6824].

One of the main issues faced by SOCKS version 5 is that, when taking into account the TCP handshake, method negotiation, authentication, connection request and grant, it may take up to 5 RTTs for a data exchange to take place at the application layer. This is especially costly in networks with a large delay at the access layer, such as 3G, 4G, or satelite.

The desire to reduce the number of RTTs manifests itself in the design of newer security protocols. TLS version 1.3 [I-D.ietf-tls-tls13] defines a zero round trip (0-RTT) handshake mode for connections if the client and server had previously communicated.

TCP Fast Open [RFC7413] is a TCP option that allows TCP to send data in the SYN and receive a response in the first ACK, and aims at obtaining a data response in one RTT. The SOCKS protocol needs to concern itself with at least two TFO deployment scenarios: First, when TFO is available end-to-end (at the client, at the proxy, and at the server); second, when TFO is active between the client and the proxy, but not at the server.

This document describes the SOCKS protocol version 6. The key improvements over SOCKS version 5 are:

- The client sends as much information upfront as possible, and does not wait for the authentication process to conclude before requesting the creation of a socket.

- The connection request also mimics the semantics of TCP Fast Open [RFC7413]. As part of the connection request, the client can supply the payload for the initial SYN that is sent out to the server.

- The protocol can be extended via options without breaking backward-compatibility.

- The protocol can leverage the aforementioned options to support 0-RTT authentication schemes.

2. Requirements language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Mode of operation

```
    CLIENT                                                        PROXY

            +------------------------+
            | Authentication methods | Request
    --------> Command code           +------------------------------->
            | TFO                    |
            | Address                |
            | Port                   |
            | Options                |
            | Initial data           |
            +------------------------+

                                        +-----------------------+
                   Authentication reply | Type                  |
     <----------------------------------+ Method                <-----
                                        | Options               |
                                        +-----------------------+

     <-------------------(Authentication protocol)------------------->

                          +-----------------------+
        Operation reply   | Reply code            |
     <--------------------+ Bind address          <-------------------
                          | Bind port             |
                          | Options               |
                          | Initial data offset   |
                          +-----------------------+
```

Figure 1: The SOCKS version 6 protocol message exchange

When a TCP-based client wishes to establish a connection to a server, it must open a TCP connection to the appropriate SOCKS port on the SOCKS proxy. The client then enters a negotiation phase, by sending the request in figure Figure 1, that contains, in addition to fields present in SOCKS 5 [RFC1928], fields that facilitate low RTT usage and faster authentication negotiation.

Next, the server sends an authentication reply. If the request did not contain the necessary authentication information, the proxy indicates an authentication method that must proceed. This may trigger a longer authentication sequence that could include tokens for ulterior faster authentications. The part labeled "Authentication protocol" is specific to the authentication method employed and is not expected to be employed for every connection between a client and its proxy server. The authentication protocol typically takes up 1 RTT or more.

If the authentication is successful, an operation reply is generated by the proxy. It indicates whether the proxy was successful in creating the requested socket or not.

In the fast case, when authentication is properly set up, the proxy attempts to create the socket immediately after the receipt of the request, thus achieving an operational conection in one RTT (provided TFO functionality is available at the client, proxy, and server).

4. Connection Requests

The client starts by sending a request to the proxy.

```
+---------------+-----------+----------+
|    Version    | Number of | Methods  |
| Major | Minor |  Methods  |          |
+-------+-------+-----------+----------+
|   1   |   1   |     1     | Variable |
+-------+-------+-----------+----------+
+---------+-----+---------+----------+------+
| Command | TFO | Address | Address  | Port |
|  Code   |     |  Type   |          |      |
+---------+-----+---------+----------+------+
|    1    |  1  |    1    | Variable |  2   |
+---------+-----+---------+----------+------+
+-----------+----------+--------------+---------------+
| Number of | Options  | Initial Data | Initial Data  |
|  Options  |          |     Size     |               |
+-----------+----------+--------------+---------------+
|     1     | Variable |      2       |   Variable    |
+-----------+----------+--------------+---------------+
```

Figure 2: SOCKS 6 Request

- Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.

- Number of Methods: The number of supported authentication methods that the client wishes to advertise.

- Methods: One byte per advertised method. Method numbers are assigned by IANA.

- Command Code:

  - 0x00 AUTH: authenticate the client and do nothing.

  - 0x01 CONNECT: requests the establishment of a TCP connection.

  - 0x02 BIND: requests the establishment of a TCP port binding.

  - 0x03 UDP ASSOCIATE: requests a UDP port association.

- TFO:

  - 0x00 indicates that the proxy MUST NOT attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command. In case of an AUTH or UDP ASSOCIATE command, this field MUST be set to 0x00.

  - 0x01 indicates that the proxy SHOULD attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command.

- Address Type:

  - 0x01: IPv4

  - 0x03: Domain Name

  - 0x04: IPv6

- Address: this field's format depends on the address type:

  - IPv4: a 4-byte IPv4 address

  - Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated.

  - IPv6: a 16-byte IPv6 address

- Port: the port in network byte order.

- Number of Options: the number of SOCKS options that appear in the Options field.

- Options: see section Section 5.

- Initial Data Size: A two-byte number in network byte order. In case of AUTH, BIND or UDP ASSOCIATE, this field MUST be set to 0. In case of CONNECT, this is the number of bytes of initial data that are supplied in the following field.

- Initial Data: The first octets of the data stream.

Clients MUST support the "No authentication required" method. Clients MAY omit advertising the "No authentication required" option.

Clients SHOULD NOT issue AUTH commands unless they advertise authentication methods with support for 0-RTT authentication.

The server MAY truncate the initial data to an arbitrary size and disregard the rest.

5. SOCKS Options

SOCKS options have the following format:

```
+---------------+-------------+
| Kind | Length | Option Data |
+------+--------+-------------+
|  1   |   1    |   Variable  |
+------+--------+-------------+
```

Figure 3: SOCKS 6 Option

- Kind: MUST be allocated by IANA. (See section Section 9.)
- Length: The length of the option data.
- Option Data: the contents are specific to each option kind.

5.1. Authentication options

Authentication options have the following format:

```
+---------------+--------+---------------------+
| Kind | Length | Method | Authentication Data |
+------+--------+--------+---------------------+
|  1   |   1    |   1    |       Variable      |
+------+--------+--------+---------------------+
```

Figure 4: Authentication Option

- Kind: MUST be allocated by IANA. (See section Section 9.)
- Length: the length of the option data.
- Method: the number of the authentication method. These numbers are assigned by IANA.

- Authentication Data: the contents are specific to each method.

All proxy implementations MUST support authentication method options. Clients MAY omit advertising authentication methods for which they have included at least an authentication option.

6. Authentication Replies

Upon receipt of a request, the proxy sends an Authentication Reply:

```
+---------------+------+--------+-----------+----------+
|    Version    | Type | Method | Number of | Options  |
| Major | Minor |      |        |  Options  |          |
+-------+-------+------+--------+-----------+----------+
|   1   |   1   |  1   |   1    |     1     | Variable |
+-------+-------+------+--------+-----------+----------+
```

Figure 5: SOCKS 6 Authentication Reply

- Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.

- Type:

  * 0x00: authentication successful.

  * 0x01: further authentication needed.

- Method: The chosen authentication method.

- Number of Options: the number of SOCKS options that appear in the Options field.

- Options: see section Section 5.

Multihomed clients SHOULD cache the chosen method on a per-interface basis and SHOULD NOT include authentication options related to any other methods in further requests originating from the same interface.

If the server signals that further authentication is needed and selects "No Acceptable Methods", the client MUST close the connection.

The client and proxy begin a method-specific negotiation. During such negotiations, the proxy MAY supply information that allows the client to authenticate a future request using an authentication

option. Descriptions of such negotiations are beyond the scope of this memo.

If the cliend issued an AUTH command, the client MUST close the connection after the negociation is complete.

7. Operation Replies

After the authentication negotiations are complete, the server sends an Operation Reply:

```
+---------------+-------+---------+----------+------+
|    Version    | Reply | Address |   Bind   | Bind |
| Major | Minor | Code  |  Type   | Address  | Port |
+-------+-------+-------+---------+----------+------+
|   1   |   1   |   1   |    1    | Variable |  2   |
+-------+-------+-------+---------+----------+------+
+-----------+----------+--------------+
| Number of | Options  | Initial Data |
|  Options  |          |    Offset    |
+-----------+----------+--------------+
|     1     | Variable |      2       |
+-----------+----------+--------------+
```

Figure 6: SOCKS 6 Operation Reply

- Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.

- Reply Code:

  - 0x00: Succes

  - 0x01: General SOCKS server failure

  - 0x02: Connection not allowed by ruleset

  - 0x03: Network unreachable

  - 0x04: Host unreachable

  - 0x05: Connection refused

  - 0x06: TTL expired

  - 0x07: Command not supported

   - 0x08: Address type not supported

- Address Type:

   - 0x01: IPv4

   - 0x03: Domain Name

   - 0x04: IPv6

- Bind Address: the proxy bound address in the following format:

   - IPv4: a 4-byte IPv4 address

   - Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated.

   - IPv6: a 16-byte IPv6 address

- Bind Port: the proxy bound port in network byte order.

- Number of Options: the number of SOCKS options that appear in the Options field.

- Options: see section Section 5

- Initial Data Offset: A two-byte number in network byte order. In case of BIND or UDP ASSOCIATE, this field MUST be set to 0. In case of CONNECT, it represents the offset in the plain data stream from which the client is expected to continue sending data.

If the proxy returns a reply code other than "Success", the client MUST close the connection.

7.1. Handling CONNECT

In case the client has issued a CONNECT request, data can now pass. The client MUST resume the data stream at the offset indicated by the Initial Data Offset field.

7.2. Handling BIND

In case the client has issued a BIND request, it must wait for a second Operation reply from the proxy, which signifies that a host has connected to the bound port. The Bind Address and Bind Port fields contain the address and port of the connecting host. Afterwards, application data may pass.

7.3. Handling UDP ASSOCIATE

The relay of UDP packets is handled exactly as in SOCKS 5 [RFC1928].

8. Security Considerations

Given the format of the request message, a malicious client could craft a request that is in excess of 100 KB and proxies could be prone to DDoS attacks.

To mitigate such attacks, proxy implementations SHOULD be able to incrementally parse the requests. Proxies MAY close the connection to the client if:

o the request is not fully received after a certain timeout, or

o the number of options exceeds an imposed hard cap, or

o the total size of the options exceeds an imposed hard cap, or

o the size of the initial data excedes a hard cap.

Further, the server MAY choose not to buffer any initial data beyond what would fit in a TFO SYN's payload.

9. IANA Considerations

This document requests that IANA allocate option codes for SOCKS 6 options. Further, this document requests an option code for authentication options.

10. Acknowledgements

The protocol described in this draft builds upon and is a direct continuation of SOCKS 5 [RFC1928].

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

11.2. Informative References

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-20 (work in progress), April 2017.

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI 10.17487/RFC1928, March 1996, <http://www.rfc-editor.org/info/rfc1928>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <http://www.rfc-editor.org/info/rfc7413>.

Authors' Addresses

Vladimir Olteanu
University Politehnica of Bucharest

Email: vladimir.olteanu@cs.pub.ro

Dragos Niculescu
University Politehnica of Bucharest

Email: dragos.niculescu@cs.pub.ro