```
Internet Area Working Group                                   V. Olteanu
Internet-Draft                                              D. Niculescu
Intended status: Experimental    University Politehnica of Bucharest
Expires: April 26, 2019                                 October 23, 2018
```

SOCKS Protocol Version 6
draft-olteanu-intarea-socks-6-05

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 [RFC8446] 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 potential 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.

1.1. Revision log

Typos and minor clarifications are not listed.

draft-05

- Limited the "slow" authentication negociations to one (and Authentication Replies to 2)
- Revamped the handling of the first bytes in the application data stream
    - False starts are now recommended. (Added the "False Start" section.)
    - Initial data is only available to clients willing to do "slow" authentication. Moved the "Initial data size" field from Requests to Authentication Method options.
    - Initial data size capped at 2^13. Initial data can no longer be dropped by the proxy.
    - The TFO option can hint at the desired SYN payload size.
- Request: clarified the meaning of the Address and Port fields.
- Better reverse TCP proxy support: optional listen backlog for TCP BIND
- TFO options can no longer be placed inside Operation Replies.
- IP TOS stack option
- Suggested a range for vendor-specific options.
- Revamped UDP functionality
    - Now using fixed UDP ports
    - DTLS support
- Stack options: renamed Proxy-Server leg to Proxy-Remote leg

draft-04

- Moved Token Expenditure Replies to the Authentication Reply.

- Shifted the Initial Data Size field in the Request, in order to make it easier to parse.

draft-03

- Shifted some fields in the Operation Reply to make it easier to parse.

- Added connection attempt timeout response code to Operation Replies.

- Proxies send an additional Authentication Reply after the authentication phase. (Useful for token window advertisements.)

- Renamed the section "Connection Requests" to "Requests"

- Clarified the fact that proxies don't need to support any command in particular.

- Added the section "TCP Fast Open on the Client-Proxy Leg"

- Options:

  - Added constants for option kinds

  - Salt options removed, along with the relevant section from Security Considerations. (TLS 1.3 Makes AEAD mandatory.)

  - Limited Authentication Data options to one per method.

  - Relaxed proxy requirements with regard to handling multiple Authentication Data options. (When the client violates the above bullet point.)

  - Removed interdependence between Authentication Method and Authentication Data options.

  - Clients SHOULD omit advertising the "No authentication required" option. (Was MAY.)

  - Idempotence options:

    - Token Window Advertisements are now part of successful Authentication Replies (so that the proxy-server RTT has no impact on their timeliness).

    - Proxies can't advetise token windows of size 0.

    - Tweaked token expenditure response codes.

    - Support no longer mandatory on the proxy side.

  - Revamped Socket options

    - Renamed Socket options to Stack options.

    - Banned contradictory socket options.

    - Added socket level for generic IP. Removed the "socket" socket level.

    - Stack options no longer use option codes from setsockopt().

    - Changed MPTCP Scheduler constants.

draft-02

- Made support for Idempotence options mandatory for proxies.

- Clarified what happens when proxies can not or will not issue tokens.

- Limited token windows to 2^31 - 1.

- Fixed definition of "less than" for tokens.

- NOOP commands now trigger Operation Replies.

- Renamed Authentication options to Authentication Data options.

- Authentication Data options are no longer mandatory.

- Authentication methods are now advertised via options.

- Shifted some Request fields.

- Option range for vendor-specific options.

- Socket options.

- Password authentication.

- Salt options.

draft-01

- Added this section.

- Support for idempotent commands.

- Removed version numbers from operation replies.

- Request port number for SOCKS over TLS. Deprecate encryption/ encapsulation within SOCKS.

- Added Version Mismatch Replies.

- Renamed the AUTH command to NOOP.

- Shifted some fields to make requests and operation replies easier to parse.

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           +------------------------------->
            | Address                |
            | Port                   |
            | Options                |
            +------------------------+

            +------------------------+
    --------> Initial data           +------------------------------->
            +------------------------+

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

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

                                        +------------------------+
                   Authentication Reply | Type = Success         |
     <----------------------------------+ Method                 <-----
                                        | Options                |
                                        +------------------------+

                          +------------------------+
        Operation Reply   | Reply code             |
     <--------------------+ Bind address           <------------------
                          | Bind port              |
                          | Options                |
                          +------------------------+
```

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. Requests

The client starts by sending a request to the proxy.

```
+---------------+---------+------+---------+----------+
|    Version    | Command | Port | Address | Address  |
| Major | Minor |  Code   |      |  Type   |          |
+-------+-------+---------+------+---------+----------+
|   1   |   1   |    1    |  2   |    1    | Variable |
+-------+-------+---------+------+---------+----------+
+-----------+----------+
| Number of | Options  |
|  Options  |          |
+-----------+----------+
|     1     | Variable |
+-----------+----------+
```

Figure 2: SOCKS 6 Request

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

- Command Code:

  - 0x00 NOOP: 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.

- 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 8.

The Address and Port fields have different meanings based on the Command Code: * NOOP: The fields have no meaning. The Address Type field MUST be either 0x01 (IPv4) or 0x04 (IPv6). The Address and Port fields MUST be 0. * CONNECT: The fields signify the address and port to which the client wishes to connect. * BIND, UDP ASSOCIATE: The fields indicate the desired bind address and port. If the client does not require a certain address, it can set the Address Type field to 0x01 (IPv4) or 0x04 (IPv6), and the Address field to 0. Likewise, if the client does not require a certain port, it can set the Port field to 0.

Clients can advertise their supported authentication methods by including an Authentication Method option (see Section 8.2).

5. Version Mismatch Replies

Upon receipt of a request starting with a version number other than 6.0, the proxy sends the following response:

Version	
Major	Minor
1	1

Figure 3: SOCKS 6 Version Mismatch Reply

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

A client MUST close the connection after receiving such a reply.

6. Authentication Replies

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

Version		Type	Method	Number of Options	Options
Major	Minor				
1	1	1	1	1	Variable

Figure 4: 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 8.

Multihomed clients SHOULD cache the chosen method on a per-interface basis and SHOULD NOT include Authentication Data 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 Data option. The client and proxy SHOULD NOT negotiate the encryption of the application data. Descriptions of such negotiations are beyond the scope of this memo.

When the negotiation is complete (either successfully or unsuccessfully), the proxy sends a second Authentication Reply. The second Authentication Reply MUST either signal success or that there are no more acceptable authentication methods.

7. Operation Replies

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

Reply Code	Bind Port	Address Type	Bind Address	Number of Options	Options
1	2	1	Variable	1	Variable

Figure 5: SOCKS 6 Operation Reply

- 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
  - 0x09: Connection attempt timed out

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

- 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

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

- Options: see Section 8.

Proxy implementations MAY support any subset of the client commands listed in Section 4.

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

If the client issued an NOOP command, the client MUST close the connection after receiving the Operation Reply.

7.1. Handling CONNECT

In case the client has issued a CONNECT request, data can now pass.

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

Proxies offering UDP functionality must be configured with a UDP port used for relaying UDP datagrams to and from the client, and/or a port used for relaying datagrams over DTLS.

Following a successful Operation Reply, the proxy sends a UDP Association Initialization message:

Association ID
4

Figure 6: UDP Association Initialization

- o Association ID: the identifier of the UDP association

Proxy implementations SHOULD generate Association IDs randomly or pseudo-randomly.

Clients may start sending UDP datagrams to the proxy either in plaintext, or over an established DTLS session, using the proxy's configured UDP ports. A client's datagrams are prefixed by a SOCKS Datagram Header, indicating the remote host's address and port:

Version		Association ID	Port	Address Type	Address
Major	Minor				
1	1	4	2	1	Variable

Figure 7: SOCKS 6 Datagram Header

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

- Association ID: the identifier of the UDP association
- 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.

Following the receipt of the first datagram from the client, the proxy makes a one-way mapping between the Association ID and:

- the 5-tuple of the UDP conversation, if the datagram was received over plain UDP, or
- the DTLS connection, if the datagram was received over DTLS. The DTLS connection is identified either by its 5-tuple, or some other mechanism, like [I-D.ietf-tls-dtls-connection-id].

Further datagrams carrying the same Association ID, but not matching the established mapping, are silently dropped.

The proxy then sends an UDP Association Confirmation message over the TCP connection with the client:

```
+--------+
| Status |
+--------+
|   1    |
+--------+
```

Figure 8: UDP Association Confirmation

- Status: MUST be 0x00

Following the confirmation message, UDP packets bound for the proxy's bind address and port are relayed to the client, also prefixed by a Datagram Header.

The UDP association remains active for as long as the TCP connection between the client and the proxy is kept open.

7.3.1. Proxying UDP servers

Under some circumstances (e.g. when hosting a server), the SOCKS client expects the remote host to send UDP datagrams first. As such, the SOCKS client must trigger a UDP Association Confirmation without having the proxy relay any datagrams on its behalf.

To that end, it sends an empty datagram prefixed by a Datagram Header with an IP address and port consisting of zeroes. The client SHOULD resend the empty datagram if an UDP Association Confirmation is not received after a timeout.

8. SOCKS Options

SOCKS options have the following format:

Kind	Length	Option Data
1	1	Variable

Figure 9: SOCKS 6 Option

- o Kind: MUST be allocated by IANA. (See Section 13.)
- o Length: The length of the option.
- o Option Data: The contents are specific to each option kind.

Unless otherwise noted, client and proxy implementations MAY omit supporting any of the options described in this document. Upon encountering an unsupported option, a SOCKS endpoint MUST silently ignore it.

8.1. Stack options

Stack options can be used by clients to alter the behavior of the protocols on top of which SOCKS is running, as well the protcols used by the proxy to communicate with the remote host (i.e. IP, TCP, UDP). A Stack option can affect either the proxy’s protocol on the client-proxy leg or on the proxy-remote leg. Clients can only place Stack options inside SOCKS Requests.

Proxies MAY include Stack options in their Operation Replies to signal their behavior. Said options MAY be unsolicited, i. e. the proxy MAY send them to signal behaviour that was not explicitly requested by the client.

In case of UDP ASSOCIATE, the stack options refer to the UDP traffic relayed by the proxy.

Stack options that are part of the same message MUST NOT contradict one another.

Kind	Length	Leg	Level	Code	Data
1	1	2 bits	6 bits	1	Variable

Figure 10: Stack Option

- Kind: 0x01 (Stack option)

- Length: The length of the option.

- Leg:

  - 0x1: Client-Proxy Leg

  - 0x2: Proxy-Remote Leg

  - 0x3: Both Legs

- Level:

  - 0x01: IP

  - 0x02: IPv4

  - 0x03: IPv6

  - 0x04: TCP

  - 0x05: UDP

- o Code: Option code

- o Data: Option-specific data

8.1.1. IP TOS options

Kind	Length	Leg	Level	TOS
1	1	2 bits	6 bits	1

Figure 11: IP TOS Option

- o Kind: 0x01 (Stack option)

- o Length: 4

- o Leg: Either 0x01, 0x02, or 0x03 (Client-Proxy, Proxy-Remote or Both legs)

- o Level: 0x04 (TCP).

- o Code: 0x01

The client can use IP TOS options to request that the proxy use a certain value for the IP TOS field. Likewise, the proxy can use IP TOS options to advertise the TOS values being used.

8.1.2. TFO options

Kind	Length	Leg	Level	Code	Payload Size
1	1	2 bits	6 bits	1	2

Figure 12: TFO Option

- o Kind: 0x01 (Stack option)

- o Length: 4

- o Leg: 0x2 (Proxy-Remote leg).

- o Level: 0x04 (TCP).

- Code: 0x01

- Payload Size: The desired payload size of the TFO SYN. MUST be 0 in case of a BIND command.

If a SOCKS Request contains a TFO option, the proxy SHOULD attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command. Otherwise, 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 a CONNECT command, the client can indicate the desired payload size of the SYN. The proxy MAY use a different payload size than the one indicated.

8.1.3. Multipath TCP options

In case of a CONNECT command, the proxy can inform the client that the connection to the server is an MPTCP connection.

Kind	Length	Leg	Level	Code
1	1	2 bits	6 bits	1

Figure 13: Multipath TCP Option

- Kind: 0x01 (Stack option)

- Length: 4

- Leg: 0x2 (Proxy-Remote leg)

- Level: 0x04 (TCP).

- Code: 0x02

8.1.4. MPTCP Scheduler options

In case of a CONNECT or BIND command, a client can use an MPTCP Scheduler option to indicate its preferred scheduler for the connection.

A proxy can use an MPTCP Scheduler option to inform the client about what scheduler is in use.

Kind	Length	Leg	Level	Code	Scheduler
1	1	2 bits	6 bits	1	1

Figure 14: MPTCP Scheduler Option

- Kind: 0x01 (Stack option)
- Length: 5
- Leg: Either 0x01, 0x02, or 0x03 (Client-Proxy, Proxy-Remote or Both legs).
- Level: 0x04 (TCP)
- Code: 0x03
- Scheduler:
  - 0x01: Default
  - 0x02: Round-Robin
  - 0x03: Redundant

8.1.5. Listen Backlog options

Kind	Length	Leg	Level	Backlog
1	1	2 bits	6 bits	2

Figure 15: Listen Backlog Option

- Kind: 0x01 (Stack option)
- Length: 5
- Leg: 0x02 (Proxy-Remote leg)
- Level: 0x04 (TCP)
- Code: 0x04

- Backlog: The length of the listen backlog. MUST be greater than 1.

The default behavior of the BIND does not allow a client to simultaneously handle multiple connections to the same bind address. An authenticated client can alter BIND’s behavior by adding a TCP Listen Backlog Option to a BIND Request.

In response, the proxy sends a TCP Listen Backlog Option as part of the Operation Reply, with the Backlog field signalling the actual backlog used. The proxy SHOULD NOT use a backlog longer than requested.

Following the successful negotiation of a backlog, the proxy listens for incoming connections for as long as the initial connection stays open. The initial connection is not used to relay data between the client and a remote host.

To accept connections, the client issues further BIND Requests using the bind address and port supplied by the proxy in the initial Operation Reply.

8.2. Authentication Method options

Authentication Method options are placed in SOCKS Requests to advertise supported authentication methods. In case of a CONNECT Request, they are also used to specify the amount of initial data supplied before any method-specific authentication negotiations take place.

Kind	Length	Initial Data Length	Methods
1	1	2	Variable

Figure 16: Authentication Method Option

- Kind: 0x02 (Authentication Method option)

- Length: The length of the option.

- Initial Data Size: A two-byte number in network byte order. In case of CONNECT, this is the number of bytes of initial data that are supplied by the client immediately following the Request. This number MUST NOT be larger than 2^13.

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

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

8.3. Authentication Data options

Authentication Data options carry method-specific authentication data. They can be part of SOCKS Requests and Authentication Replies.

Authentication Data options have the following format:

Kind	Length	Method	Authentication Data
1	1	1	Variable

Figure 17: Authentication Data Option

- o Kind: 0x03 (Authentication Data option)
- o Length: The length of the option.
- o Method: The number of the authentication method. These numbers are assigned by IANA.
- o Authentication Data: The contents are specific to each method.

Clients SHOULD only place one Authentication Data option per authentication method. Server implementations MAY silently ignore all Authentication Data options for the same method aside from an arbitrarily chosen one.

8.4. Idempotence options

To protect against duplicate SOCKS Requests, authenticated clients can request, and then spend, idempotence tokens. A token can only be spent on a single SOCKS request.

Tokens are 4-byte unsigned integers in a modular 4-byte space. Therefore, if x and y are tokens, x is less than y if 0 < (y - x) < 2^31 in unsigned 32-bit arithmetic.

Proxies grant contiguous ranges of tokens called token windows. Token windows are defined by their base (the first token in the range) and size. Windows can be shifted (i. e. have their base increased, while retaining their size) unilaterally by the proxy.

Requesting and spending tokens is done via Idempotence options:

Kind	Length	Type	Option Data
1	1	1	Variable

Figure 18: Idempotence Option

- o Kind: 0x04 (Idempotence option)

- o Length: The length of the option.

- o Type:

  - * 0x00: Token Request

  - * 0x01: Token Window Advertisement

  - * 0x02: Token Expenditure

  - * 0x03: Token Expenditure Reply

- o Option Data: The contents are specific to each type.

8.4.1. Requesting a fresh token window

A client can obtain a fresh window of tokens by sending a Token Request option as part of a SOCKS Request:

Kind	Length	Type	Window Size
1	1	1	4

Figure 19: Token Request

- o Kind: MUST be allocated by IANA. (See Section 13.)

- Length: 7

- Type: 0x00 (Token Request)

- Window Size: The requested window size.

If a token window is issued, the proxy then includes a Token Window Advertisement option in the corresponding successful Authentication Reply:

Kind	Length	Type	Window Base	Window Size
1	1	1	4	4

Figure 20: Token Window Advertisement

- Kind: 0x04 (Idempotence option)

- Length: 11

- Type: 0x01 (Token Grant)

- Window Base: The first token in the window.

- Window Size: The window size. This value SHOULD be lower or equal to the requested window size. Window sizes MUST be less than 2^31. Window sizes MUST NOT be 0.

If no token window is issued, the proxy MUST silently ignore the Token Request.

8.4.2. Spending a token

The client can attempt to spend a token by including a Token Expenditure option in its SOCKS request:

Kind	Length	Type	Token
1	1	1	4

Figure 21: Token Expenditure

- Kind: 0x04 (Idempotence option)

- Length: 7

- Type: 0x02 (Token Expenditure)

- Token: The token being spent.

Clients SHOULD prioritize spending the smaller tokens.

The proxy responds by sending a Token Expenditure Reply option as part of the successful Authentication Reply:

Kind	Length	Type	Response Code
1	1	1	1

Figure 22: Token Expenditure Response

- Kind: 0x04 (Idempotence option)

- Length: 4

- Type: 0x03 (Token Expenditure Response)

- Response Code:

  - 0x01: Success: The token was spent successfully.

  - 0x02: No Window: The proxy does not have a token window associated with the client.

  - 0x03: Out of Window: The token is not within the window.

  - 0x04: Duplicate: The token has already been spent.

If eligible, the token is spent as soon as the client authenticates. If the token is not eligible for spending, the proxy MUST NOT attempt to honor the client's SOCKS Request; further, it MUST indicate a General SOCKS server failure in the Operation Reply.

Proxy implementations SHOULD also send a Token Window Advertisement if:

- the token is out of window, or

- by the proxy's internal logic, successfully spending the token caused the window to shift.

Proxy implementations SHOULD NOT shift the window's base beyond the highest unspent token.

Proxy implementations MAY include a Token Window Advertisement in any Authentication Reply that indicates success.

8.4.3. Handling Token Window Advertisements

Even though the proxy increases the window's base monotonically, there is no mechanism whereby a SOCKS client can receive the Token Window Advertisements in order. As such, clients SHOULD disregard unsolicited Token Window Advertisements with a Window Base less than the previously known value.

9. Username/Password Authentication

Username/Password authentication is carried out as in [RFC1929].

Clients can also attempt to authenticate by placing the Username/Password request in an Authentication Data Option, provided that it is no longer than 252 bytes.

Kind	Length	Method	Username/Password request
1	1	1	Variable

Figure 23: Password authentication via a SOCKS Option

- Kind: MUST be allocated by IANA. (See Section 13.)

- Length: The length of the option.

- Method: 0x02 (Username/Password).

- Username/Password request: The Username/Password request, as described in [RFC1929].

10. TCP Fast Open on the Client-Proxy Leg

TFO breaks TCP semantics, causing replays of the data in the SYN's payload under certain rare circumstances [RFC7413]. A replayed SOCKS

Request could itself result in a replayed connection on behalf of the client.

As such, client implementations SHOULD NOT use TFO on the client-proxy leg unless:

- The protocol running on top of SOCKS tolerates the risks of TFO, or
- The SYN's payload does not contain any application data (so that no data is replayed to the server, even though duplicate connections are still possible), or
- The client uses Idempotence Options, making replays impossible, or
- SOCKS is running on top of TLS and Early Data is not used.

11. False Starts

In case of CONNECT Requests, the client MAY start sending application data as soon as possible, as long as doing so does not incur the risk of breaking the SOCKS protocol.

Clients must work around the authentication phase by doing any of the following:

- If the Request does not contain an Authentication Method option, the authentication phase is guaranteed not to happen. In this case, application data MAY be sent immediately after the Request.
- Application data MAY be sent immediately after receiving an Authentication Reply indicating success.
- When performing a method-specific authentication sequence, application data MAY be sent immediately after the last client message.

12. Security Considerations

12.1. Large requests

Given the format of the request message, a malicious client could craft a request that is in excess of 80 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:

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

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

- the total size of the options exceeds an imposed hard cap.

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

12.2. Replay attacks

In TLS 1.3, early data (which is likely to contain a full SOCKS request) is prone to replay attacks.

While Token Expenditure options can be used to mitigate replay attacks, the initial Token Request is still vulnerable. As such, client implementations SHOULD NOT make use of TLS early data when sending a Token Request.

13. IANA Considerations

This document requests that IANA allocate 1-byte option kinds for SOCKS 6 options. Further, this document requests the following option kinds:

- Stack options: 0x01

- Authentication Method options: 0x02

- Authentication Data options: 0x03

- Idempotence options: 0x04

- A range for vendor-specific options: 0xC0-0xFF

This document also requests that IANA allocate a TCP and UDP port for SOCKS over TLS and DTLS, respectively.

14. Acknowledgements

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

15. References

15.1. Normative References

[RFC1929] Leech, M., "Username/Password Authentication for SOCKS V5", RFC 1929, DOI 10.17487/RFC1929, March 1996, <https://www.rfc-editor.org/info/rfc1929>.

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

15.2. Informative References

[I-D.ietf-tls-dtls-connection-id] Rescorla, E., Tschofenig, H., Fossati, T., and T. Gondrom, "Connection Identifiers for DTLS 1.2", draft-ietf-tls-dtls-connection-id-02 (work in progress), October 2018.

[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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc7413>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

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