SOCKS Protocol Version 6 (update)
draft-olteanu-intarea-socks-6-01

Vladimir Olteanu, Dragoș Niculescu
University Politehnica of Bucharest
Improvements over v5

- Shave off RTTs: Client sends as much information as possible upfront
  - Optimistic, doesn’t wait for authentication to conclude
  - Method advertisement, server address, some application data
- Client can specify if it wants TFO on the proxy-server leg
- Extensible: TCP-like options
- 0-RTT authentication support via options
SOCKSv5 vs. SOCKSv6 [1/2]
SOCKSv5 vs. SOCKSv6 [2/2]

- Can include authentication data in the request on subsequent connections
New Security Features

• Deprecate support for encryption
• Just run SOCKS over TLS
  – Request new port from IANA

• TLS 1.3 has support for early data
  – 0-RTT overhead
  – Likely to contain a full SOCKS request
  – Prone to replay attacks
• Need mechanism that makes SOCKS requests idempotent
SOCKS Request idempotence

- Leverage SOCKS options

- **Authenticated** clients can be granted single-use tokens
  - Tokens are assigned on a per-user basis

- A token can only be spent on a single operation
  - Proxies and clients keep track of spent tokens
Requesting Tokens

Diagram:
- SOCKSv6
- Proxy Client
- Proxy Server
- Request
- Auth Reply
- Operation Reply
Requesting Tokens

SOCKSv6

Proxy Client

Proxy Server

Request + Token Request (Size)

Auth Reply

Operation Reply

+ Window Advertisement (Base, Size)
# Token Request

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Type</th>
<th>Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

- Client starts by requesting a number of tokens
  - Can be done as part of a NOOP request
  - Only needs to be done once (or in corner cases)
  - Secure, as long as TLS early data is not used
# Token Window Advertisement

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Type</th>
<th>Window Base</th>
<th>Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

- Proxy offers a number of consecutive Tokens
  - Window Base: first token
  - Window Size: number of tokens

- E.g.: base=10, size=5 means that the following tokens are available: 10, 11, 12, 13, 14
Spending Tokens

Request + Token Expenditure (Token)

Auth Reply

Operation Reply

- Expenditure Reply
- (Optional) Window Advertisement (Base, Size)
Token Expenditure

+-----------------------------------------------+
| Kind | Length | Type   | Token |
|-----------------------------------------------|
| 1    | 1      | 1      | 4     |
+-----------------------------------------------+

- Client spends Tokens on Operations
  - Clients SHOULD attempt to spend tokens in order
Token Expenditure Reply

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Type</th>
<th>Response Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Server replies:
  - Duplicate or out-of-window tokens are rejected
Shifting the token window

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Type</th>
<th>Window Base</th>
<th>Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

• Proxies can **unilaterally increment** the Window Base
  - Lowest-order tokens are discarded, new high-order tokens are created
  - Send unsolicited Token Window Advertisements to let clients know
• Use cases
  - Ideal: Lowest-order Tokens are spent; shift the base past them
  - The client has begun spending higher-order tokens; shift window past low-order gaps
What’s next?

- Options for influencing the proxy’s behavior
  - MPTCP Path Manager
  - MPTCP Scheduler
- Better reverse proxy support
- Ability to listen() on a socket and have connections forwarded
Comparison to 0-RTT TCP converters

- draft-bonaventure-mptcp-converters-02

**Similarity:** No control data aside from initial exchange

**Different starting point:** purely layer 5 protocol
- Can be run over TLS
- TFO data not required, but highly beneficial
- Middlebox doesn’t kill TCP => middlebox doesn’t kill SOCKS
Extra Slides
Token Space

• Tokens are
  – 32-bit unsigned integers
  – in a 32-bit modular space

• $x < y$ if $(y-x) < 2^{31}$