New in -05

- Different handling of first bytes of application data
- Reverse TCP proxy: can now handle concurrent incoming connections to the same port
- UDP behaviour revamped
False start

- Simple core state machine
False start

- Simple core state machine
- Proxy can’t complicate it unless client asks for it
False start

• Send application data ASAP
  – Just make sure not to break the state machine

• Right after Request, if unwilling to do “long” authentication

• Right after Authentication Reply, if 0-RTT authentication succeeds

• Right after last message in authentication sequence, otherwise
Initial data

- Serves no purpose unless “long” authentication is performed
- “Initial Data Length” field moved
  - Request → Authentication Method option
- Capped at 16K
- Can no longer be dropped by proxy
  - Removed “Initial Data Offset” field from Operation Reply
Handling TFO

• Added “Payload Length Field” to TFO Option
• Preserve TFO semantics
  – Data in TFO payload has weaker guarantees
• Ensure good timing in certain corner cases
  – Payload should be big enough to elicit a data response
TFO corner case: fragmented payload

Request + GET /index.h

SYN + ACK

GET /index.h

Low RTT

High RTT
TFO corner case: fragmented payload

Request: GET /index.h

SYN + ACK

Client

Proxy

Server

GET /index.h

tml HTTP/1.1

Low RTT

High RTT
TFO corner case: fragmented payload

Request + GET /index.html

SYM + ACK

tml HTTP/1.1

GET /index.html

SYM + ACK

tml HTTP/1.1

Low RTT

High RTT
TFO corner case: fragmented payload

Request + GET /index.h
SYN + ACK
tml HTTP/1.1

Client

Proxy

GET /index.h
SYN + ACK
tml HTTP/1.1

Server

HTTP 200 OK

Low RTT

High RTT
Using the correct TFO payload

Client

Request + GET /index.h

SYN + ACK

tml HTTP/1.1

HTTP 200 OK

Proxy

GET /index.html HTTP/1.1

HTTP 200 OK

Server

HTTP 200 OK

Low RTT

High RTT
TCP Reverse Proxy

- The BIND command handles one incoming connection
  - listen(), accept() once and close() listening socket
- Want to emulate typical server behavior
  - listen(), accept(), accept(), accept()…
Listen Backlog Option

• First BIND: include a Listen Backlog Option
  – Prompts proxy to listen() for as long as connection is open

• Each further BIND to same address+port
  – Has the proxy accept() an incoming connection from the same listen()ing socket

• Authenticated clients only
UDP Relay

- Revamped from v5
- DTLS support
- Firewall-friendly: same relay port for all clients (1080 by default; DTLS port TBD)
UDP Relay

- **Request**: UDP ASSOC, bind addr+port
- **Op. Reply**: bind addr, port
- **Assoc. Init.**: assoc. ID

- A UDP port is bound
- An Association ID is generated for the binding
The first datagram triggers an Association Confirmation

The assoc. ID is mapped to the UDP/DTLS conversation
UDP Relay

Client

Request: UDP ASSOC, bind addr+port

Op. Reply: bind addr, port

Asoc. Init.: assoc. ID

UDP datagram

Asoc. Confirmation

Proxy

• UDP traffic can pass in both directions now

TCP       UDP
**SOCKS Datagram Header**

<table>
<thead>
<tr>
<th>Version</th>
<th>Association</th>
<th>Port</th>
<th>Address Type</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major</td>
<td>Minor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>Variable</td>
</tr>
</tbody>
</table>

- Carried by all datagrams on client-proxy leg
- Contains address of remote host
- Association ID is used for multiplexing
Nits

- TOS Stack option (useful for UDP)
- All Idempotence options now either in Requests or Authentication Replies
- Limited authentication phases to 1 (oversight)
- Removed TFO options from Operation Replies (no use case)
Implementation

- Complies with -04

- Message library: https://github.com/45G/libsocks6msg

- Utility library: https://github.com/45G/libsocks6util

- Proxyfier + proxy: https://github.com/45G/sixtysocks
What’s next?

- SOCKS Sessions
  - Killer use case: ToR (different session = different circuit)
  - Better granularity for idempotence and “multi”-bind
    - Proxy holds state per session, rather than per user