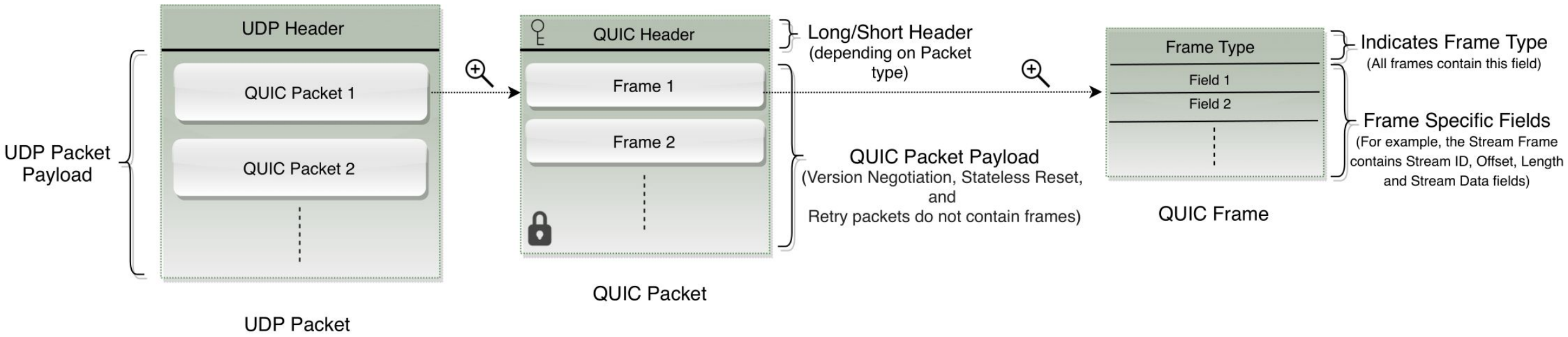
DPIFuzz: A Differential Fuzzing Framework to Detect DPI Elusion Strategies for QUIC

Gaganjeet Reen and Christian Rossow

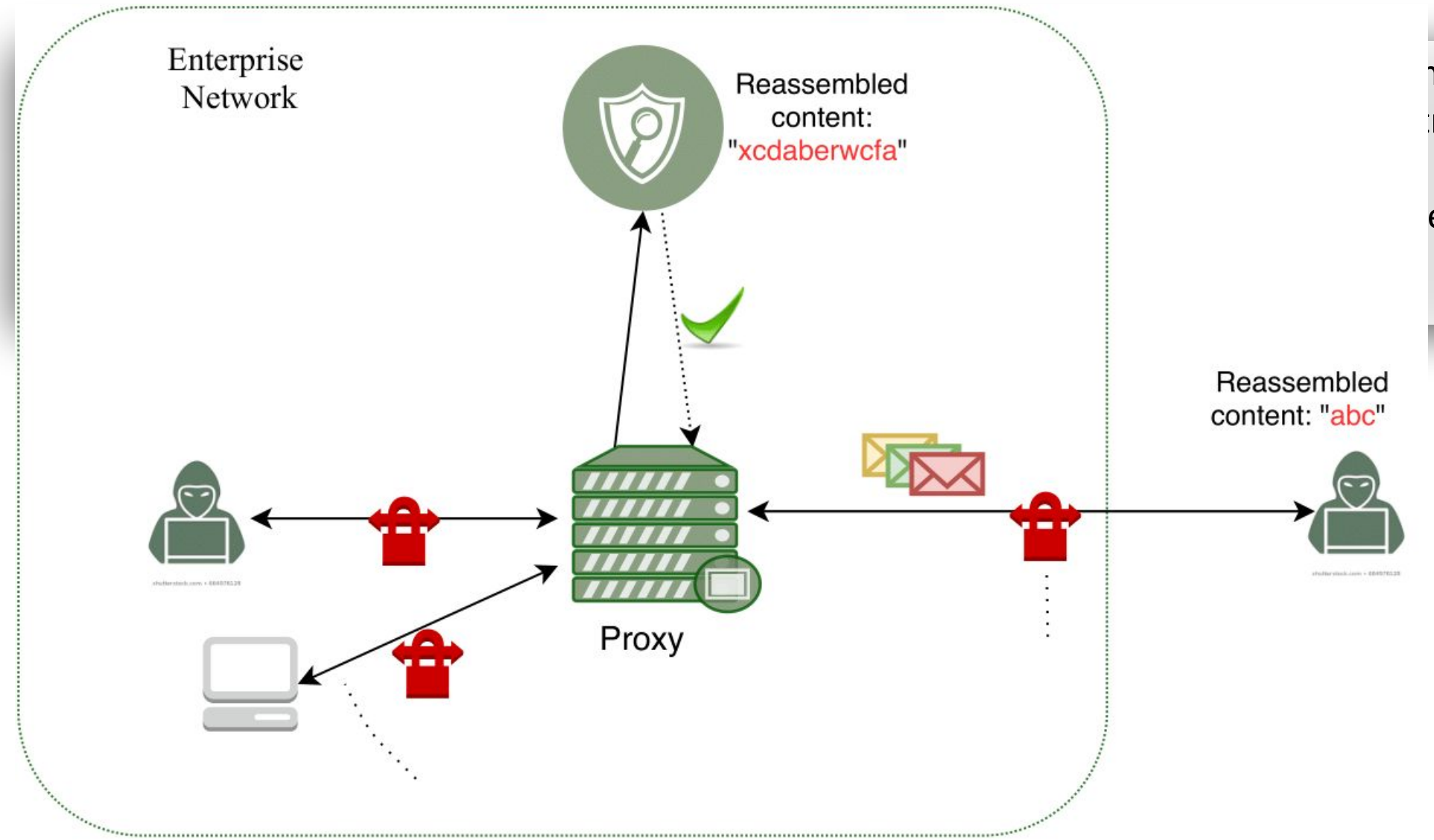
CISPA Helmholtz Center for Information Security

QUIC Overview

- Transport protocol that has the potential to replace TCP in the near future.
- Encrypted-by-default. Conceptually similar to a combination of TCP, TLS, and HTTP/2 implemented on UDP.



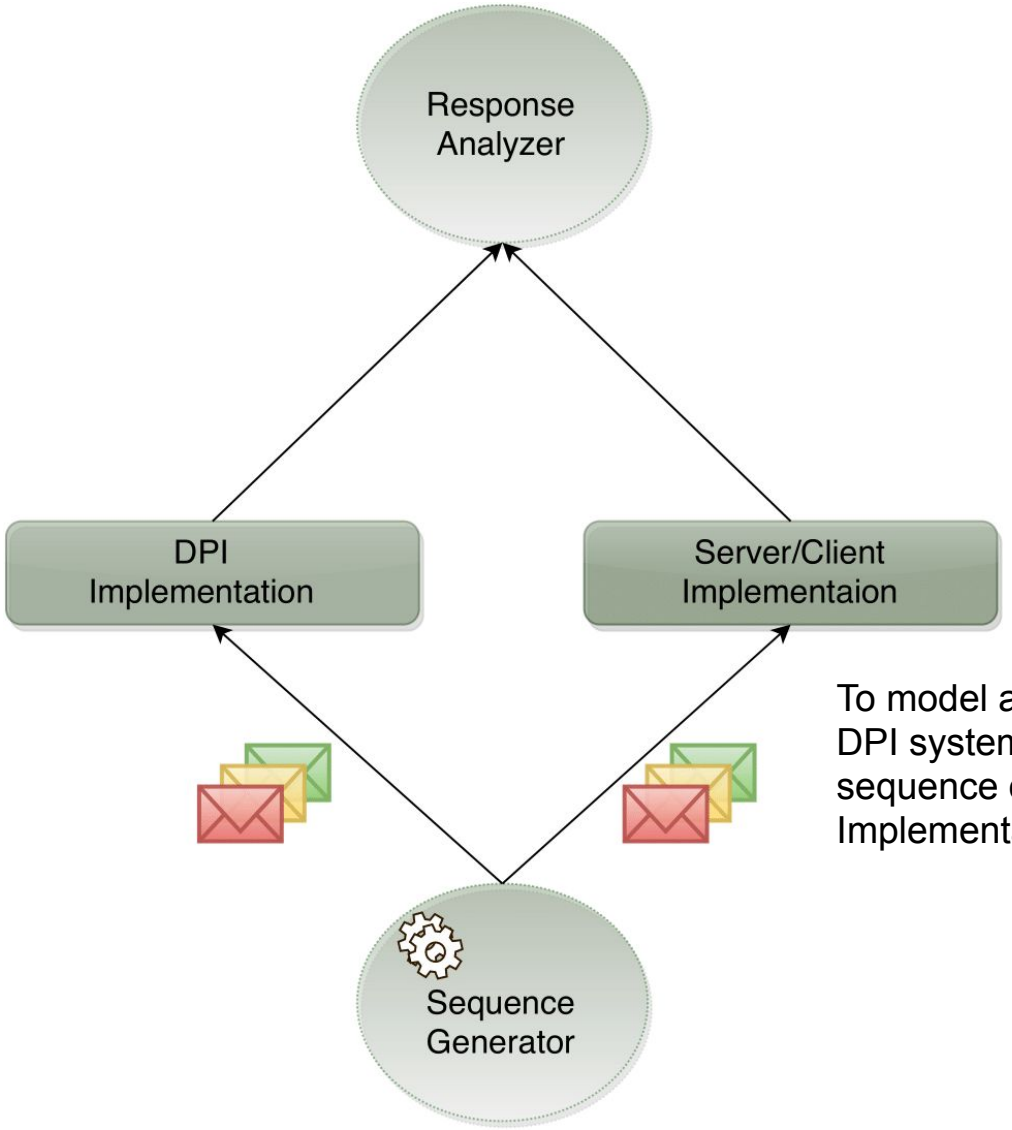
Threat Model



ntial strategies that
raffic.

ed by the DPI

Broad Overview of the design



We use 5 popular open source implementations to model a DPI system.

To model an inline proxy that forwards the packets to a DPI system for further inspection, we feed same logical sequence of packets to the DPI system and the Implementation Under Test (IUT).

Uses fuzzing to generate sequences

Challenges

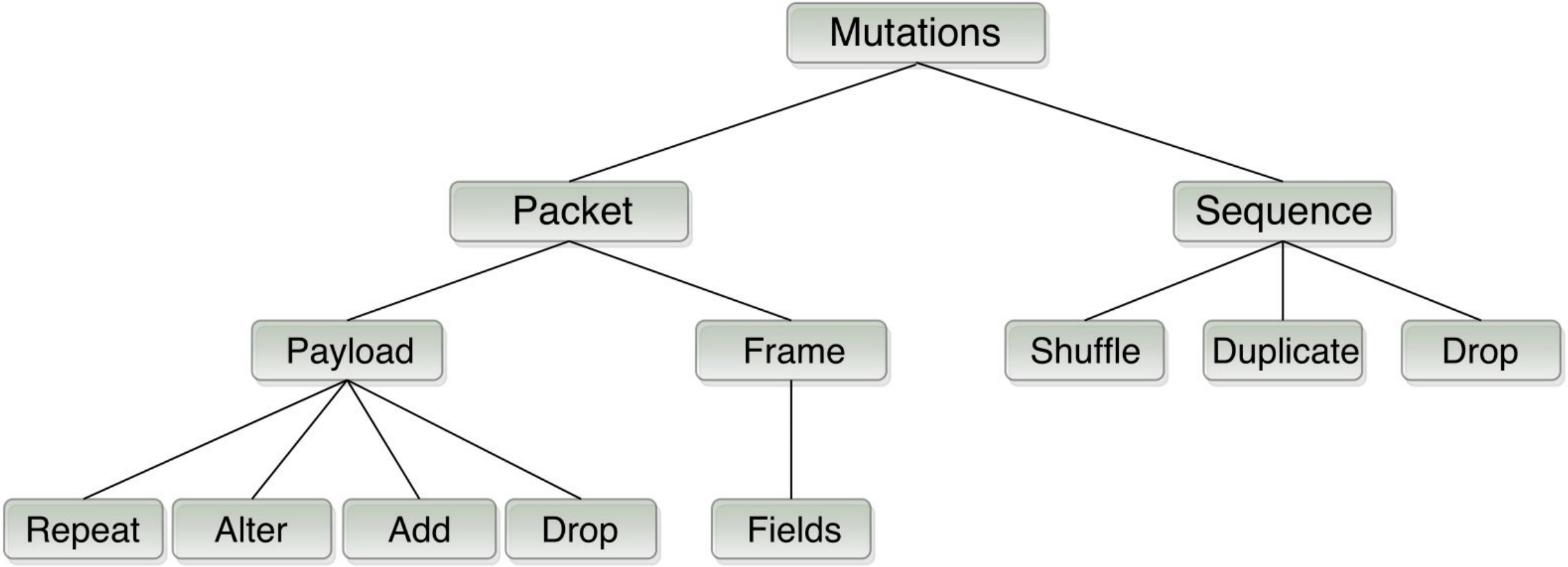
CISPA HELMHOLTZ-ZENTRUM i. G.

- **Challenge**: Implementations of protocols like QUIC expect packets with very specific structure and field values in the absence of which, the packets are discarded right away. A fuzzer will only be able to detect meaningful vulnerabilities if it can inject unexpected inputs deep into the state space of a protocol.
  Solution: We design a "smart" fuzzer which has structure-aware "generators" and "mutators".

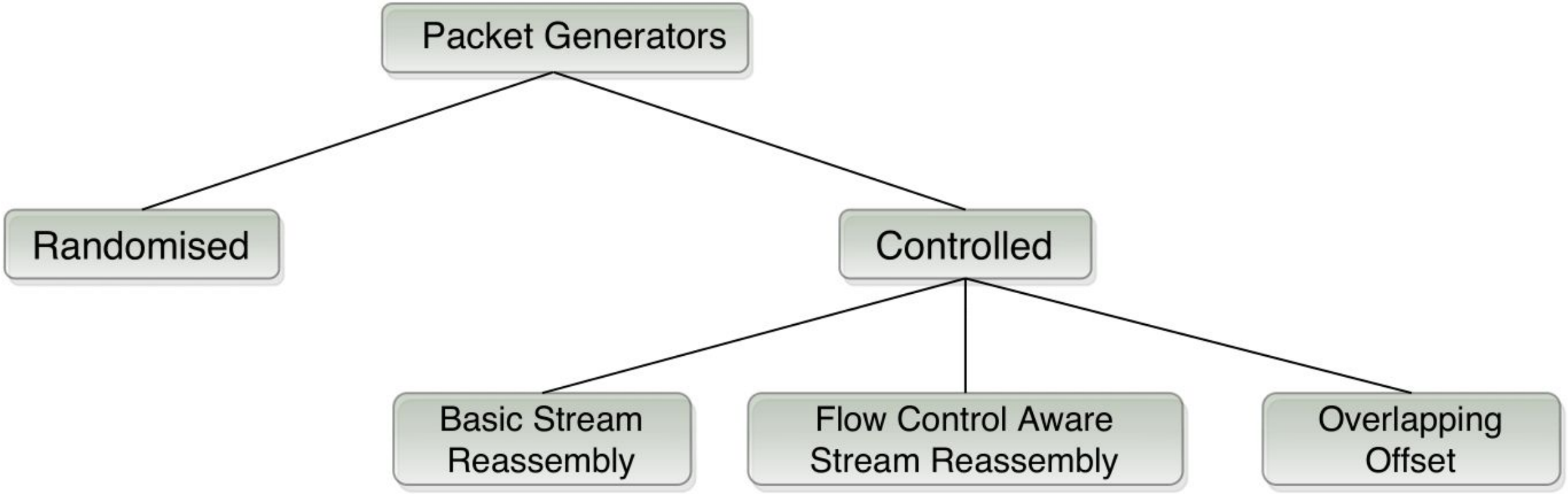
- **Challenge**: QUIC is an encrypted by default protocol. Fuzzing encrypted packets will almost certainly render a packet structurally invalid.
  Solution: We fuzz packets before they are encrypted.

- **Challenge**: Cannot replay previously captured traffic because completing a handshake is essential before any data can be exchanged
  Solution: Our fuzzer is capable of (i) completing handshakes and (ii) interacting actively with the implementations.

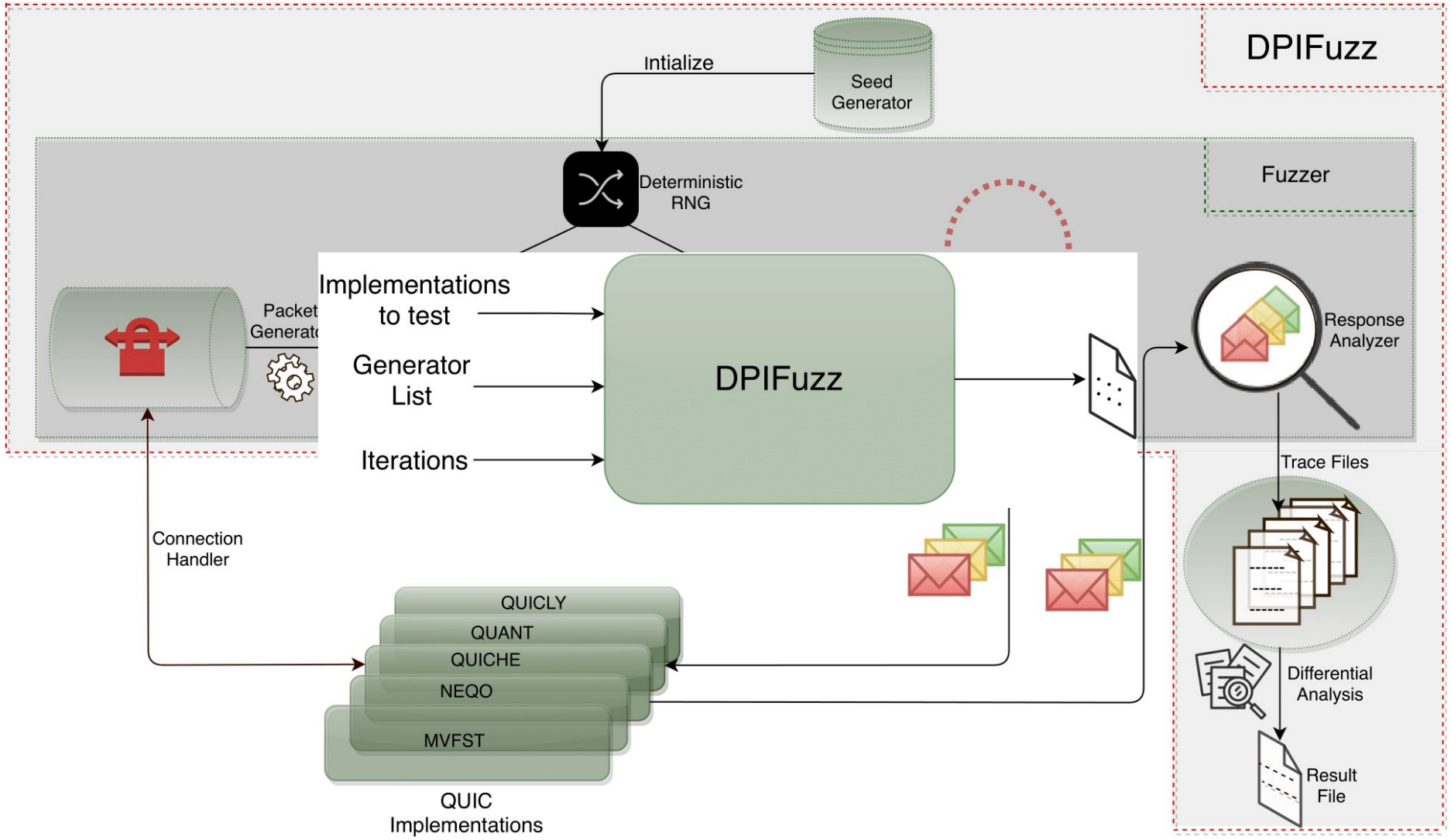
Mutations


Mutations
Packet
Sequence
Payload
Frame
Shuffle
Duplicate
Drop
Repeat
Alter
Add
Drop
Fields

Packet Generators



DPIFuzz Architecture



Experiment Setup

- The experiments were run on a machine with a Quad-Core Intel i5 processor with a 16GB RAM running Ubuntu 18.04.
- To remove the effects of network latency and unintentional packet reordering, we run our echo servers as well as DPIFuzz locally.
- In total, we use our framework to create 600 unique sequences, i.e., we run our fuzzer against each IUT, with each of the 3 specified generators, with 200 different seed values.

Transport Parameter	Value
initial_max_data	1048576
initial_max_stream_data_bidi_local	66560
initial_max_stream_data_bidi_remote	66560
initial_max_stream_data_uni	66560
initial_max_streams_bidi	2048
initial_max_streams_uni	2048
max_idle_timeout	60000
max_packet_size	1500
ack_delay_exponent	3

Reassembly Differences

- Exploiting packets with **duplicate packet numbers**

Seed Value	Generator	QUICHE Reassembled Data	MVFST Reassembled Data	QUICLY Reassembled Data
4373445819122772715	Basic Stream Reassembly	2Z?b"?@N5?#48SZyBp????;yGJ-+$0P7cdWYaBMetcem=+@Wf-Sja1xZwhYKFFA26AN&YI_	yGJ+$0P7cdWYaBMetcem=+@-WfSja1xZwhYKFFA26AN&YI_	yGJ+$0P7cdWYaBMetcem=+@-WfSja1xZwhYKFFA26AN&YI_

- The QUICHE server reassembles data from 2 streams while the QUICLY and MVFST servers only reassemble data from 1 stream. This happens because the sequence includes a packet with a duplicate packet number.

Continued…

- Exploiting **Stream Offset Overlaps**

Packet No.	Stream Frame Payload	Stream Offset	Stream Finbit	QUICHE Reassembled Data (Destination Server)	QUICLY Reassembled Data (DPI system)
1	OCKED	2	True	__OCKED	__OCKED
2	BLIN	0	False	BLOCKED	BLINKED

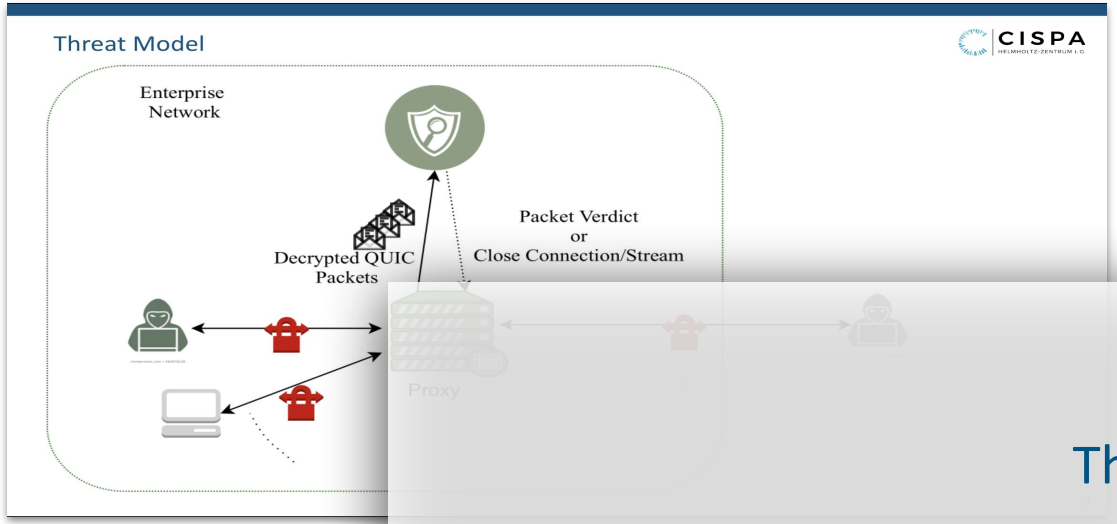
Packet No.	Stream Frame Payload	Stream Offset	Stream Finbit	MVFST Reassembled Data (Destination Server)	QUICLY Reassembled Data (DPI system)
1	OCKED	2	True	__OCKED	__OCKED
2	INKED	2	False	__OCKED	__INKED
3	BL	0	False	BLOCKED	BLINKED

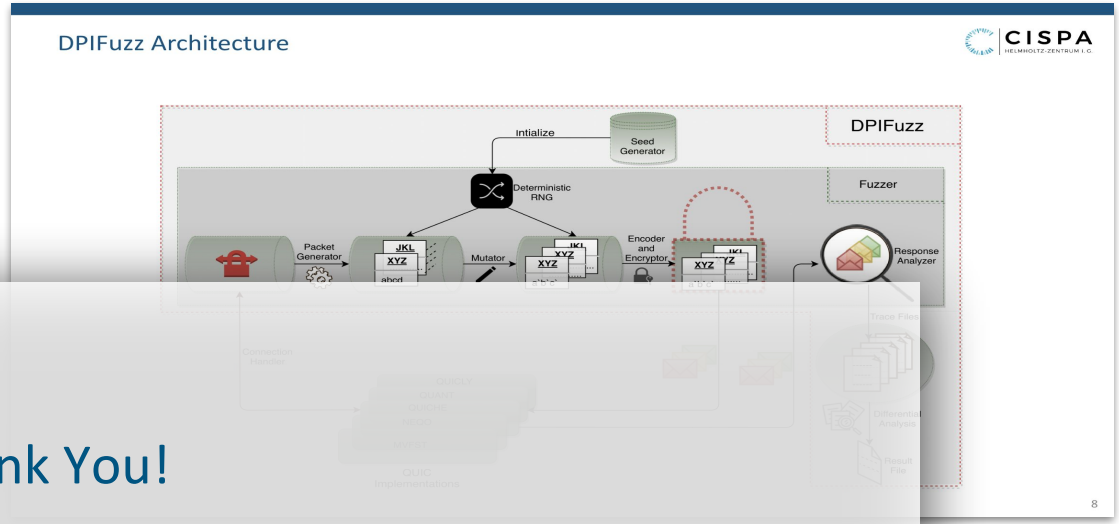
Bugs and Vulnerabilities

Seed Value	Generator	Implementation	Error Description
5224880393376231849	Basic Stream Reassembly	MVFST	Null Pointer Dereference
6867396659762739268	Flow-Control-Aware Stream Reassembly	QUANT	Heap use after free
3544824671711368728	Flow-Control-Aware Stream Reassembly	QUICLY	Null Pointer Dereference
8969571667189322506	Basic Stream Reassembly	NEQO	Assertion Failed

- MVFST: Unable to handle a stream frame in which the Offset field has a non-zero value but value of the OFF bit is set to 0 (which indicates that the Offset field is absent), the FIN bit is set to true and the Payload field empty.
- QUANT: Implementation tries to access the state of a stream, i.e., check whether a stream is closed or not after the memory allocated to the stream has already been freed.
- QUICLY: Unable to handle stream IDs not permitted by the server.
- NEQO: Attempts to close an already closed connection.

Summary

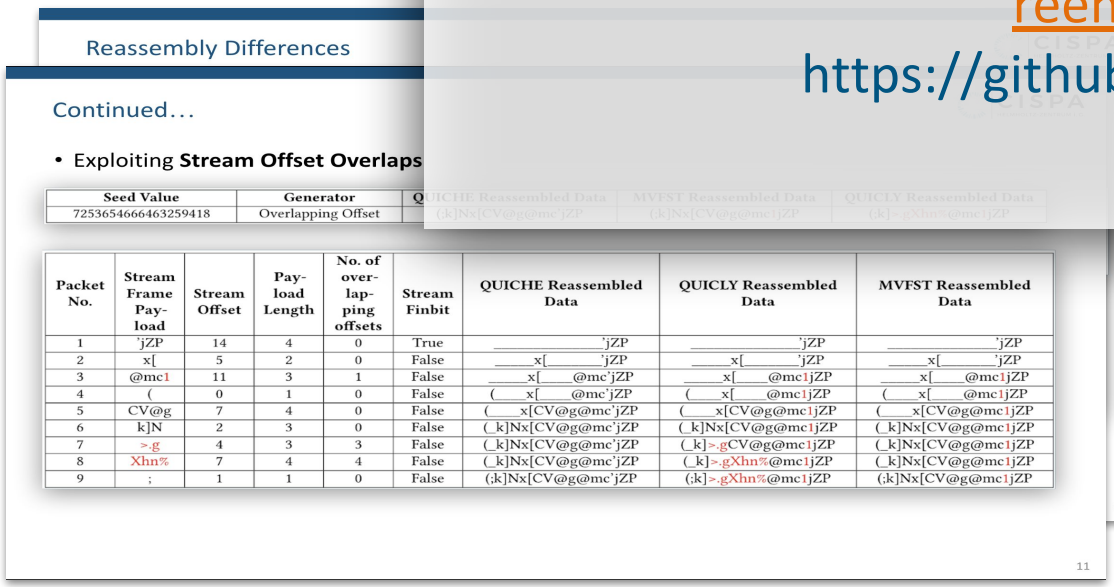




Thank You!

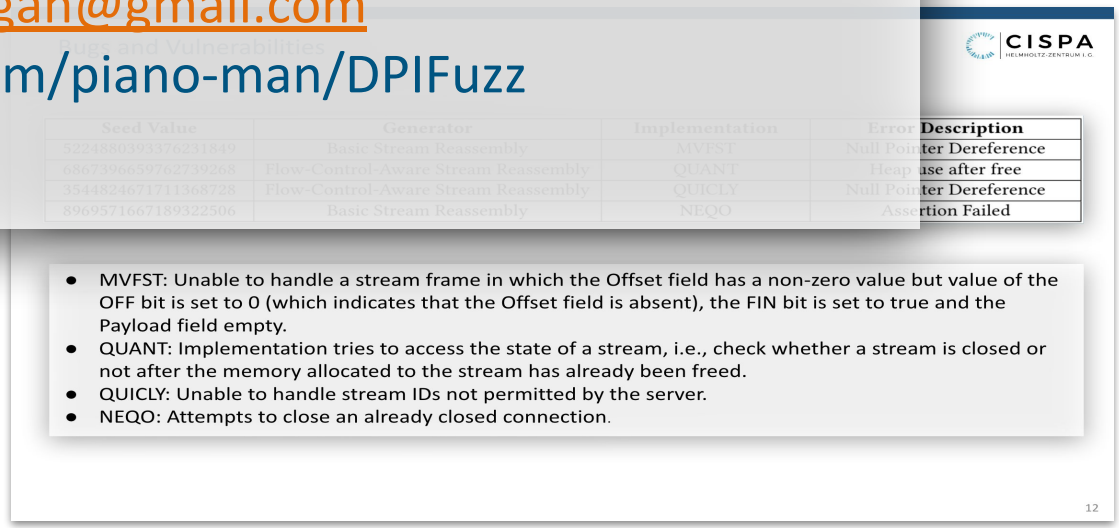
reen.gagan@gmail.com

https://github.com/piano-man/DPIFuzz



Seed Value	Generator	QUICHE Reassembled Data	MVFST Reassembled Data	QUICLY Reassembled Data
7253654666463259418	Overlapping Offset	(k]Nx[CV@g@mc'jZP	(k]Nx[CV@g@mc1jZP	(k]>.gXhn%@mc1jZP

Packet No.	Stream Frame Pay-load	Stream Offset	Pay-load Length	No. of over-lap-ping offsets	Stream Finbit	QUICHE Reassembled Data	QUICLY Reassembled Data	MVFST Reassembled Data
1	'jZP	14	4	0	True	_____'jZP	_____'jZP	_____'jZP
2	x[	5	2	0	False	_____x[_____'jZP	_____x[_____'jZP	_____x[_____'jZP
3	@mc1	11	3	1	False	_____x[____@mc'jZP	_____x[____@mc1jZP	_____x[____@mc1jZP
4	(	0	1	0	False	(____x[____@mc'jZP	(____x[____@mc1jZP	(____x[____@mc1jZP
5	CV@g	7	4	0	False	(____x[CV@g@mc'jZP	(____x[CV@g@mc1jZP	(____x[CV@g@mc1jZP
6	k]N	2	3	0	False	(_k]Nx[CV@g@mc'jZP	(_k]Nx[CV@g@mc1jZP	(_k]Nx[CV@g@mc1jZP
7	>.g	4	3	3	False	(_k]Nx[CV@g@mc'jZP	(_k]>.gCV@g@mc1jZP	(_k]Nx[CV@g@mc1jZP
8	Xhn%	7	4	4	False	(_k]Nx[CV@g@mc'jZP	(_k]>.gXhn%@mc1jZP	(_k]Nx[CV@g@mc1jZP
9	;	1	1	0	False	(;k]Nx[CV@g@mc'jZP	(;k]>.gXhn%@mc1jZP	(;k]Nx[CV@g@mc1jZP



Seed Value	Generator	Implementation	Error Description
5224880393376231849	Basic Stream Reassembly	MVFST	Null Pointer Dereference
6867396659762739268	Flow-Control-Aware Stream Reassembly	QUANT	Heap use after free
3544824671711368728	Flow-Control-Aware Stream Reassembly	QUICLY	Null Pointer Dereference
8969571667189322506	Basic Stream Reassembly	NEQO	Assertion Failed

- MVFST: Unable to handle a stream frame in which the Offset field has a non-zero value but value of the OFF bit is set to 0 (which indicates that the Offset field is absent), the FIN bit is set to true and the Payload field empty.
- QUANT: Implementation tries to access the state of a stream, i.e., check whether a stream is closed or not after the memory allocated to the stream has already been freed.
- QUICLY: Unable to handle stream IDs not permitted by the server.
- NEQO: Attempts to close an already closed connection.