USENIX Security '25 Artifact Appendix: Exposing and Circumventing SNI-based QUIC Censorship of the Great Firewall of China

Ali Zohaib*
University of Massachusetts Amherst

Qiang Zao*
GFW Report

Jackson Sippe
University of Colorado Boulder

Abdulrahman Alaraj
University of Colorado Boulder

Amir Houmansadr
University of Massachusetts Amherst

Zakir Durumeric
Stanford University

Eric Wustrow
University of Colorado Boulder

A Artifact Appendix

A.1 Abstract

Our paper presents the first study of China's new capability to censor QUIC connections based on the Server Name Indication (SNI) field. We find that China has started decrypting QUIC Initial packets at scale, employing unique filtering rules and a distinct blocklist different from its other censorship mechanisms. We measure the blocked domains, reverse-engineer the filtering rules (e.g., filtering connections where source port > destination port), and demonstrate how modest QUIC traffic surges can overwhelm the system and reduce the effectiveness of its blocking. Furthermore, we show how the GFW's QUIC blocking mechanism can be exploited to block UDP connections between arbitrary hosts.

This artifact provides the code and data from our experiments. It is structured to make all figures and results in the paper reproducible and to enable independent verification of our claims about the GFW's QUIC-SNI blocking.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The tools used in this artifact do not pose any risk to the evaluators' machines or data privacy. All measurements performed using this artifact are non-destructive and are for verification purposes only.

A.2.2 How to access

The repository containing the code and data can be cloned from GitHub: https://github.com/gfw-report/usenixsecurity25-quic-sni. Archived versions of the artifact are available under the DOI: 10.5281/zenodo.15626727.

*Ali Zohaib and Qiang Zao contributed equally to this work.

A.2.3 Hardware dependencies

For the AE reviewers, we have provisioned two Virtual Private Servers (VPSes) for experiments. Both servers are configured with a single CPU core and 2GB of RAM. The first is hosted in China at the Tencent Cloud Guangzhou Datacenter (AS45090) and uses an Intel Xeon Platinum 8255C CPU. The second is in the U.S. at AWS Oregon (AS16509) and uses an Intel Xeon E5-2686 v4 CPU. Reviewers can SSH into the servers using the credentials provided.

A.2.4 Software dependencies

Both servers provided to the reviewers run Ubuntu 20.04 LTS. The following software tools and libraries are required on the local machine to compile and run the tools and experiments in this artifact:

- GNU make utility
- Docker >= 20.10.0
- Python 3.9+

The primary evaluation method uses a Docker container to build all necessary tools. However, individual tools can be compiled and run directly on your local machine without Docker. If you choose this path, you will also need to install:

- Go 1.22+
- Rust (rustc,cargo) 1.82+

A.2.5 Benchmarks

None.

A.3 Set-up

We provide AE reviewers with two pre-configured VPSes (usenix-ae-us and usenix-ae-cn) that include all the required tools and dependencies. For those who wish to set up their own environment, we also offer a one-click script to automatically compile and transfer the tools on new VPSes.

A.3.1 Installation

- Docker: https://docs.docker.com/get-docker/
- GNU make: https://gnu.org/software/make/
- Download the repository: https://github.com/gfw-report/usenixsecurity25-quic-sni using Git clone.
- Compile and transfer tools to VPSes:

```
cd usenixsecurity25-quic-sni/utils;
make  \
  SERVER_HOST=$USENIX_AE_US \
  SERVER_USER=$USENIX_AE_US_USER \
  SERVER_SSH_KEY=$USENIX_AE_SSH_KEY \
  CLIENT_HOST=$USENIX_AE_CN \
  CLIENT_USER=$USENIX_AE_CN_USER \
  CLIENT_SSH_KEY=$USENIX_AE_SSH_KEY
```

Running the `make` command will compile the necessary tools and transfer tools to the servers. The `SERVER_HOST/USER/KEY` and `CLIENT_HOST/USER/KEY` variables should be set using the credentials provided.

A.3.2 Basic Test

Log in to the two VPSes using the provided credentials:

```
ssh -i $USENIX_AE_SSH_KEY \
  $USENIX_AE_CN_USER@$USENIX_AE_CN

# In a separate terminal window:
ssh -i $USENIX_AE_SSH_KEY \
  $USENIX_AE_US_USER@$USENIX_AE_US
```

Then run the following command on the usenix-ae-us VPS:

```
sudo tcpdump udp and host $USENIX_AE_CN \
 -Uw - | ./server-parser
```

On the usenix-ae-cn VPS:

```
echo baidu.com | ./quic-sni-sender  -p 443 \
  --sport=65000 --dip=$USENIX_AE_US \
  --socket-pool-size 1
```

After 10 seconds, check the terminal on the server (usenix-ae-us). You will see output in CSV format. The value in the last column indicates the test result. A value of `True` means the QUIC connection was blocked, while `False` means the connection was successful.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): GFW decrypts QUIC Initial packets to inspect SNI fields and block QUIC connections. This is proven by experiment (E1).

(C2): The GFW blocks QUIC connections where the client's source port is greater than the server's destination port. This is supported by experiment (E2).

(C3): The GFW exhibits varied responses to different QUIC and QUIC-like payloads. This can be confirmed by experiment (E3).

(C4): Sending a dummy UDP payload before the Client Initial packet can bypass the GFW's QUIC-SNI blocking. This is proven by experiment (E4).

A.4.2 Experiments

Before proceeding, ensure the VPSes are set up with the tools and dependencies detailed in the previous section. The experiments will be conducted between a client machine in China (usenix-ae-cn) and a server in the U.S. (usenix-ae-us). In each experiment, the client will send QUIC packets/probes to the server. To prevent residual interference, a 5-minute interval should be observed between each test. We recommend running the experiments multiple times, as the GFW's QUIC-blocking behavior can exhibit diurnal variations (see Section 3.4 of the paper).

(E1): *[Test QUIC-SNI Blocking] [5 human-minutes + 5 compute-mintues]: This experiment verifies the GFW's ability to decrypt and block QUIC connections based on SNI in the Initial Packet.*

Preparation: Log in to the two VPSes using the provided credentials.

Execution: On the usenix-ae-us VPS, run:

```
sudo tcpdump udp and host $USENIX_AE_CN \
-Uw - | ./server-parser
```

On the client (usenix-ae-cn), run the following command to send a QUIC probe. This probe consists of a Client Initial packet, followed by a 5 second delay, and then subsequent 10-byte UDP payloads.

```
echo google.com | ./quic-sni-sender -p 443 \
--socket-pool-size 1 --dip=$USENIX_AE_US \
--sport=55000
```

Results: Wait approximately 10 seconds for the server-parser script on the server to produce its output. A typical result will appear as follows:

```
# 2025/06/04 03:01:34 Started parsing: /dev/stdin
# tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
# 2025-06-04T03:02:08Z,{usenix-ae-cn_IP},{usenix-ae-us_IP},55001,4437,google.com,true
```

The parser determines that a connection is blocked if it receives the QUIC Initial Packet but none of the subsequent UDP payloads arrive. This outcome is indicated by the true value in the final column. A false value signifies that no blocking was detected. Test other SNI values like `baidu.com` (to verify behavior for exempt domains) and `cloudflare-dns.com`, `youtube.com` (to verify blocking for other domains).

(E2): *[Testing Rule: SourcePort > Destination Port] [5 human-minutes + 5 compute-minutes]: This experiment tests the GFW's filtering rule for QUIC Initial packets in which the source port must be greater than the desti-*

nation port.

Preparation: Run the parser `( sudo tcpdump udp and host $USENIX_AE_CN -Uw - | ./server-parser )` from E1 on the server (usenix-ae-us).

Execution: Run the following command on the client (usenix-ae-cn):

```
echo google.com | ./quic-sni-sender \
-p 5000 --socket-pool-size 1 \
--dip=$USENIX_AE_US \
--followup-payloads=10 \
--bind-ip=0.0.0.0 \
--sport=4000 \
--no-use-greater-srcports
# SrcPort (4000) < DestPort (5000)
-> Expected Result: NOT BLOCKED


echo google.com | ./quic-sni-sender \
-p 443 --socket-pool-size 1 \
--dip=$USENIX_AE_US \
--followup-payloads=10 \
--bind-ip=0.0.0.0 \
--sport=65000
# SrcPort (65000) > DestPort (443) \
-> Expected Result: BLOCKED
```

Results The results from the first command should show a `False` value in the last column, indicating that the connection was not blocked. The second command should show a `True` value, indicating that the connection was blocked due to the source port being greater than the destination port.

(E3): *[Test Different QUIC Payloads] [30 human-minutes + 30 compute-minutes]: This experiment evaluates the GFW's blocking behavior based on various QUIC payloads as detailed in Table 3 of the paper. The payloads can be generated using the `quic-packet-builder` utility in the `utils` directory.*

Preparation: Begin by capturing UDP traffic on the server (usenix-ae-us) using: `sudo tcpdump udp and host $USENIX_AE_CN`

Execution: On the client (usenix-ae-cn), run the following commands using the payloads files linked in the table here.

```
# Send QUIC payload using netcat
nc -u -q 0 -p 60001 $USENIX_AE_US 444 \
< ./payloads/exp1.bin
```

Then send follow-up payloads repeatedly to the same destination port using the same source port using the following command:

```
# Send arbitrary UDP payload to the server
echo "10101010101010101010" | xxd -r -p \
| nc -u -q 0 -p 60001 $USENIX_AE_US 444
```

Results: Depending on if any follow-up payload packets arrive at the server, one can determine if the connection was blocked or not. The results should match findings in Table 3 of the paper. Change the source and destination ports between each test to avoid residual interference.

(E4): *[Test Dummy Payload Bypass] [5 human-minutes + 5 compute-minutes]: This experiment shows how sending a dummy UDP payload before the Client Initial packet can bypass the GFW's blocking.*

Preparation: Capture UDP traffic on the server (usenix-ae-us) using: `sudo tcpdump udp and host $USENIX_AE_CN`

Execution: Run the following command on the client (usenix-ae-cn):

```
# Send an arbitrary UDP payload to the server
echo "00000000000000000000000000000000" \
| xxd -r -p | nc -u -q 0 \
-p 65535 $USENIX_AE_US 6126
```

Wait a few seconds and then send the QUIC packet containing a forbidden SNI (google.com in this example):

```
echo google.com | ./quic-sni-sender -p 6126 \
--socket-pool-size 1 --dip=$USENIX_AE_US \
--sport=65535
```

Results: Successful arrival of all subsequent packets on the 4-tuple `(src_ip, 65535, dst_ip, 6126)` at the server confirms the bypass.

A.5 Reproducing Paper Resources

To reproduce the figures and data presented in the paper, follow these instructions: First, set up your Python environment:

```
python3 -m venv venv
. venv/bin/activate
pip install -r requirements.txt
```

Each figure and table in the paper can be reproduced individually via the make command in its respective experiment directory. A comprehensive list of the commands required to reproduce each figure and table is provided in Tables 1 and 2. These commands should be run from the `experiments` directory.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.

Section	Figure	Command	Output file
3.2 GFW's Blocking Latency	Fig 2 & 10	cd how-fast-gfw-blocks; make clean; make	how-fast-the-gfw-blocks.pdf, how-fast-the-gfw-blocks-boxplot.pdf
3.3 Blocking Rule: Source Port >= Destination Port	Fig 3 & 11	cd rule-srcport-greater-than-dst-port; make clean; make	heatmap-ports-401-450-step-1_heatmap.pdf, heatmap-ports-1-65000-step-1000_heatmap.pdf
3.4 Diurnal Blocking Pattern	Fig 4	cd diurnal-blocking; make clean; make	diurnal-timeseries-three-sources.pdf
3.6 Parsing Idiosyncrasies	Fig 5	cd what-triggers-blocking; make clean; make	quic_parse_heatmap.pdf
4. Monitoring the Blocklist over Time	Fig 6	cd sni-blocklist; make clean; make	domains-blocked-over-quic-weekly.pdf
4.1. Comparison with Other Blocklists	Fig 7	cd overlap-between-blocklists; make clean; make	venn-intersection-between-lists.pdf
5. GFW Degradation Attack	Fig 8	cd degradation-attack; cd Figure_8_experiments /AVG_exp23-22-20_sensitive-stressing_and_exp25-26-27_random-stressing; make clean; make	stressing_rates.eps

Table 1: This table lists the commands required to reproduce each figure from the main paper using the provided artifact. For each figure, the corresponding section, command, and output file are specified.

Table	Data sources	Command
1	./network-tap/data/tuple-count-2025-01-22-16-00.statistics-quic-conn.txt, ./network-tap/data/tuple-count-2025-01-22-16-00.statistics-udp-pkt.txt	—
2	./ttl-location/data/DNS/city-dns-and-traceroute-result.txt	—
3	./what-triggers-blocking/payloads, ./what-triggers-blocking/results.txt	—
4	./sni-blocklist/	`make clean && make`
5	./overlap-between-blocklists	`make clean && make`
6	./availability-attack/data/ec2/	—

Table 2: This table lists the data sources and commands required to reproduce each table from the main paper using the artifact repository.