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

Abstract

Despite QUIC handshake packets being encrypted, the Great Firewall of China (GFW) has begun blocking QUIC connections to specific domains since April 7, 2024. In this work, we measure and characterize the GFW's censorship of QUIC to understand *how* and *what* it blocks. Our measurements reveal that the GFW decrypts QUIC Initial packets at scale, applies heuristic filtering rules, and uses a blocklist distinct from its other censorship mechanisms. We expose a critical flaw in this new system: the computational overhead of decryption reduces its effectiveness under moderate traffic loads. We also demonstrate that this censorship mechanism can be weaponized to block UDP traffic between arbitrary hosts in China and the rest of the world. We collaborate with various open-source communities to integrate circumvention strategies into Mozilla Firefox, the quic-go library, and all major QUIC-based circumvention tools.

1 Introduction

Since its standardization in 2021, QUIC has rapidly become a major Internet protocol. It now serves as the cryptographic basis of HTTP/3 [7] and in 2024, Cloudflare estimated that over 30% of web requests use QUIC [14]. QUIC's popularity also poses a problem for censors, who must adapt their previous techniques to the new protocol. While censors have previously altogether blocked the protocol [25] [70 §5.2], for the first time, user reports began to suggest that the Great Firewall of China (GFW) had started blocking QUIC connections for specific domains [33] in April 2024, similar to their SNI-based censorship of TLS traffic [11, 37].

Censoring QUIC connections to specific websites is challenging at the state-level because QUIC encrypts all packets, unlike TLS where the destination server name is sent in plaintext. In QUIC, even the first handshake message, the QUIC client Initial, is encrypted, albeit under a key that is derivable by a passive network observer. This means that a censor that wants to block QUIC connections based on the Server Name Indication (SNI) field needs to decrypt the first packet of every QUIC connection to reveal the destination site. It is thus important for the anti-censorship community to understand the GFW's new censorship design and implementation details to update circumvention strategies.

In this work,[1] we measure China's new capability to inspect and block QUIC connections—the first nation-wide inspection and targeted censorship of QUIC. We confirm that China is decrypting and inspecting the first packet in QUIC connections at scale. Through several experiments, we infer the rules and high-level parsing logic of how the GFW processes QUIC connections. For instance, we discover that the GFW ignores QUIC packets with a source port lower than the destination port, likely as an optimization to inspect client-only traffic.

We use traceroute-like measurements to show that the devices responsible for QUIC censorship are co-located at the same hop as existing GFW devices, indicating that they may use shared infrastructure or have similar management. However, despite this proximity, we measure the set of domains that trigger QUIC censorship, and find that the GFW's QUIC blocklist substantially differs from blocklists used for TLS, HTTP, or DNS censorship in China. In particular, the QUIC blocklist is roughly 60% of the size of the DNS blocklist in terms of number of domains. Surprisingly, a large number of these domains do not even support QUIC, making it unclear why they ended up on a QUIC-specific censorship list.

We further demonstrate that China's targeted QUIC censorship can be overwhelmed such that the GFW is not fully able to censor QUIC connections. This reveals an exploitable flaw in the GFW's QUIC censorship where an attacker can send a moderate number of QUIC Initial packets—even to uncensored domains—and overwhelm the GFW such that other QUIC connections to censored domains are blocked at dramatically lower rates.

Finally, we show that the GFW's QUIC censorship system makes the whole country vulnerable to attack. We present

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

[1] Project homepage: https://gfw.report/publications/usenixsecurity25/en/

an availability attack that weaponizes the QUIC censorship mechanism to block any host in China from communicating over UDP with any foreign host. For example, this attack could be used to block access to all DNS servers outside of the country causing widespread Internet outages. We demonstrate this attack against ourselves using our own servers around the world, and show that a single spoofing machine can prevent the majority of these hosts from communicating with our vantage point in China. Because of the potential severity, we disclose this vulnerability to China's CERT. We conclude with a discussion of implications for the censorship circumvention community and of the complex ethical considerations of exploiting vulnerabilities against a harmful actor, the GFW.

2 Background and Related Work

QUIC Protocol. QUIC is a UDP-based network protocol that was initially developed by Google [27] and later standardized by the IETF as RFC 9000 [43] in 2021. QUIC is akin to TLS but operates over UDP, reducing latency and enabling browser-controlled congestion control. QUIC was adopted to serve as the cryptographic basis of HTTP/3 [7] and in 2024, Cloudflare estimated that over 30% of web requests use QUIC [14]. QUIC also poses a shift for the anti-censorship community as it encrypts all packets to prevent tracking and tampering by middleboxes [27 §3].

QUIC Client Initial. The first packet in a QUIC handshake is the *Client Initial* packet. Since QUIC packets are encrypted from the outset but a key exchange has not occurred, initial packets are encrypted with a key derived from the Destination Connection ID (DCID) and a version-specific salt [58]. Both of these fields are sent in plaintext in the QUIC client initial packet, allowing the server (and a passive network observer) to decrypt the payload. As such, this protection does not provide confidentiality or integrity against observing parties, but protects against off-path spoofing attacks.

Once the payload of the initial packet is decrypted, it reveals a set of one or more `CRYPTO` frames containing a TLS 1.3 Client Hello message that lists the cipher suites and TLS extensions supported by the client. Typically, one of these TLS extensions will be the Server Name Indication (SNI), which specifies the hostname the client is attempting to connect to. Because initial keys can be computed by any network observer, the TLS Client Hello and its plaintext contents, including the SNI, can be decrypted.

QUIC Blocking. In 2021, Elmenhorst et al. found that while many QUIC websites were not accessible in Iran and China, it was not because of any SNI-based censorship. Instead, it's because Iran blocked UDP traffic to those QUIC endhosts [25 §5.2], and China blocked both TCP and UDP traffic to those specific QUIC endhosts [25 §5.1]. Later, in March 2022, ValdikSS found that the Russian TSPU blocked all QUIC connections that used QUIC version 1 (0x00, 0x00, 0x00, 0x01), were destined to port 443, and had a payload size of at least 1001 bytes [70 §5.2] [63]. In December 2024, Uzbekistan blocked QUIC connections with Encrypted Client Hello (ECH) extensions [16]. To our best knowledge, China initiated the blocking of QUIC connections based on the SNI field in April 2024, *making it the first and only country being able to do so as of June 2025*.

Other Censorship Mechanisms. The GFW employs multiple methods to implement its blocking policies, including DNS poisoning [4, 6, 20, 29, 38], IP blocking [25, 66], keyword-based filtering [11, 13, 37], and active probing [2, 26] [66 §5] [65 §4.5] [22 §4.3]. For UDP-based DNS requests, the GFW injects fake responses to queries for forbidden domains. For HTTP(S) traffic, it performs stateful inspection of TCP connections and injects forged RST packets upon detecting censored domains in HTTP Host headers or TLS SNI extension fields [67]. This is followed by a brief period of "residual blocking," primarily enforced via additional forged SYN+ACK and RST injections, though recent work has shown that packet dropping is also used [37 §5.4] [10].

3 QUIC Censorship Mechanism

In this section, we investigate how the GFW detects and blocks QUIC connections to forbidden domains. We show that the GFW blocks QUIC connections based on client Initial SNI, regardless of the server IP address. The GFW inspects the first packet in a UDP flow, and if it is a QUIC client Initial containing a domain name on China's QUIC-specific blocklist, the GFW drops subsequent packets from the client to server for 3 minutes (Figure 1).

Experiment Setup and Vantage Points. We used a set of vantage points inside and outside China for our experiments, which allowed us to test connections bidirectionally over the GFW. In total, we used seven vantage points in China: four in Beijing, two in Guangzhou, and one in Shanghai. We chose these regions in China since they are home to major Internet Exchange Points (IXPs) where the GFW is known to be deployed [52 §4.5] [36] [28 VI.C]. These vantage points were provisioned through Tencent Cloud (AS45090) and Alibaba Cloud (AS37963). Outside China, we utilized six vantage points located in Singapore (AS16509), San Jose (AS14618), San Francisco (AS14061), N. Virginia (AS14618), Cape Town (AS16509), and a U.S. university (AS32).

We developed a custom QUIC client using Quiche [46] that allows us to craft specific client Initial packets. As we observed blocking to be triggered by clients' packets regardless of server response, the servers in our experiments ran `tcpdump` rather than a QUIC server. To ensure accurate measurements and avoid interference, we configured `iptables` rules on the servers to drop any outgoing ICMP packets directed to the clients.

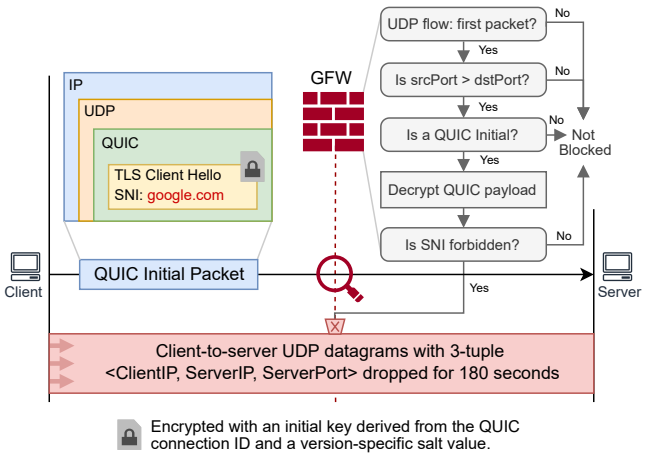

Figure 1: Overview of QUIC SNI Censorship, including the decision flow, initial packet decryption, SNI-based filtering, and residual blocking rules triggered for flagged connections.

3.1 QUIC Connection Blocking

Upon observing a QUIC client Initial message with a forbidden SNI, the GFW drops all subsequent UDP packets sharing the same source IP, destination IP, and destination port. We discovered this behavior by sending QUIC client Initial messages from the three locations of our vantage points in China to a server in the US. These messages used QUIC version 1, and `google.com` in the SNI. We found that while the QUIC client Initial messages reached our server, any subsequent UDP packets in the connection from client to server were dropped by the censor for 180 seconds.

During this time, if the client sent 10 byte random data packets from even a different source port to the same server endpoint (destination IP and port), these were dropped by the GFW as well. However, random data packets sent to a new destination port on the server were not blocked, indicating the GFW blocks based on the 3-tuple (source IP, destination IP, destination port) to prevent trivial circumvention attempts via source port changes. As confirmed by having the server send a Server Initial message along with additional UDP packets carrying random 10-byte payloads after receiving the QUIC client Initial message, we also observed that this blocking was only client-to-server; server-sent packets were not dropped.

A Single Packet Triggers Residual Blocking. Unlike previously documented TLS-SNI censorship by the GFW [38 §3.1], which requires the detection of at least two packets (SYN followed by a PSH/ACK), the QUIC censorship mechanism can be activated by a single QUIC client Initial packet containing a forbidden SNI. This is the first known instance of the GFW implementing *residual blocking via packet dropping for a UDP-based protocol* . While the GFW has historically censored DNS traffic over UDP through spoofed packet injections [4, 6, 38], it has not employed packet dropping as a method of blocking for UDP-based protocols. This may be because this type of censorship requires an *in-path* capability to drop packets, compared to prior injection-only censorship which can be accomplished with *on-path* techniques where the censoring device only sees a copy of packets. The residual packet-dropping behavior of the GFW also introduces a new vector for availability attacks, where an attacker can use the GFW to block communication between arbitrary hosts. We explore this attack in Section 6.

Inconsistent Bidirectional Blocking. While our preliminary experiments showed that traffic entering or exiting China could trigger blocking, this behavior changed on September 30, 2024. Since then, inbound traffic to most of our vantage points has no longer triggered blocking, with the exception of traffic to Beijing and Guangzhou.

Blocking Latency. Our experiments show a brief delay between detection of a QUIC client Initial packet containing a forbidden SNI and when the GFW begins to drop packets, which allows several packets to reach the server. The fact that GFW was not able to drop the QUIC client Initial with forbidden SNI shows a level of *on-path* deployment of the GFW [64 §2.1]. In combination with the *in-path* packet dropping capability, we consider its deployment architecture to be a *hybrid* of both on-path and in-path, which is similar to the GFW's blocking of TLS ESNI traffic [32].

To precisely measure this blocking delay, we adopted a methodology similar to what used in a 2020 study that measured the delay in blocking of TLS traffic containing an Encrypted Server Name Indication (ESNI) extension [32]. Specifically, we measured how long we continued to receive subsequent packets after a triggering QUIC Initial packet.

We conducted a day-long experiment to determine the GFW's blocking latency for QUIC. From our vantage point in Beijing, for the first five minutes of each hour, we initiated 25 UDP flows (unique source and destination ports) to a server we controlled in Singapore. In each connection, we continuously sent unique 10-byte payloads at a rate of 100 packets per second. Five seconds into the experiment, we sent a QUIC Initial packet with a forbidden SNI `google.com` to each of the 25 destination ports corresponding to the ongoing connections, but from different source ports. These QUIC Initials trigger the GFW to block each of the 25 destination ports (for our client/server pair for any source port) such that the server would stop receiving the 10-byte payloads.

On the server side, we captured packets and looked for connections in which there was at least a 120-second gap between UDP packets, indicating that the QUIC client Initial successfully triggered censorship. For these connections, we looked at the sending time on the client of the blocking QUIC Initial and the last UDP payload the server received before the censorship-induced gap. This represents the blocking latency when packets were still allowed to pass the GFW immediately following a censorship trigger to a 10 ms granularity.

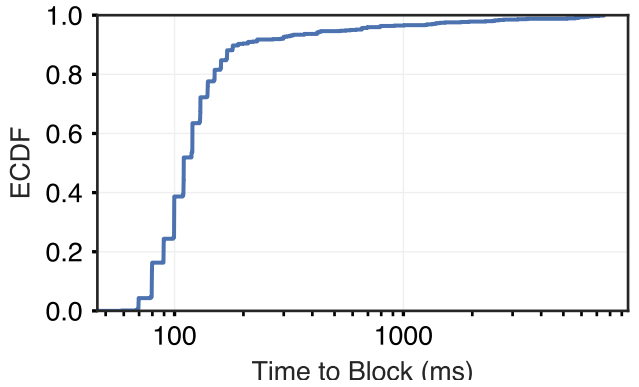

Figure 2: The CDF shows the distribution of the time taken by the GFW to enact a blocking rule causing subsequent packet drops. In over 90% of cases, the GFW blocks the connection within one second. The observed blocking times range from a minimum of 0.06 seconds to a maximum of 7.5 seconds.

Blocking latencies ranged 60 ms to 7.5 seconds (Figure 2). Over 90% of connections were blocked within one second, but there is a long tail that takes longer. We hypothesize that the variable delay in blocking corresponds to the variable volume of QUIC traffic the GFW must process (See Appendix A). We explore exploiting this property in Section 5 to degrade the performance of the GFW's QUIC censorship.

3.2 Flow Tracking Logic

Unlike TCP, which has a clear three-way handshake that marks the start of a new connection, UDP is connectionless without any explicit transport layer handshake. This makes it challenging for middleboxes to identify the beginning of a new UDP flow. Although, QUIC connections can be traced using Connection IDs (CIDs), we found that the GFW does not use CIDs to track QUIC flows. Instead, it uses the UDP 4-tuple (source IP, destination IP, source port, destination port) and employs a *60-second timeout for keeping state* in its flow tracking system. To learn this, we relied on the fact that the GFW will only block a connection if the first packet in a UDP flow is a QUIC client Initial message with a forbidden SNI. If any other UDP packet precedes the Initial packet, the connection will not be blocked.

From one of our Beijing vantage points, we sent a UDP packet with a random 10-byte payload to a server in the U.S. We then waited a variable delay before sending three QUIC client Initial messages in the same connection as the 10-byte random, each spaced one second apart. We repeated this experiment, increasing the delay between the random payload packet and client Initial packets by one second each iteration, until we observed blocking (i.e. no packets received for 180 seconds).

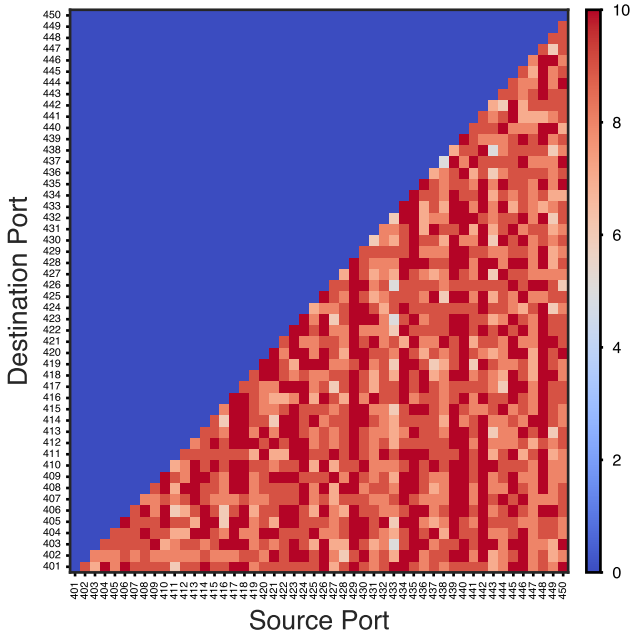

Figure 3: GFW exempts connections where the source port of the QUIC Initial packet is equal to or lower than the destination port. The experiment was conducted on December 2, 2024, from a vantage point in Beijing, China.

We found that blocking occurred when the delay between the first random payload and the client Initial packet reached 60 seconds implying that the flow initiated by the random UDP payload was tracked for 60 seconds. The QUIC Initial packets sent after this 60 second window triggered blocking, indicating that the GFW had reset the state for the flow and was treating the QUIC Initial packets as a new flow.

No UDP Reassembly. We found that the GFW does not reassemble QUIC Initial packets that are split across more than one UDP datagram. This design choice may be reasonable at the time of its deployment on April 7, 2024, considering that there had been few QUIC clients sending large QUIC Initial packets that do not fit in a typical UDP datagram. However, as detailed in Section 7, since September 13, 2024 [1], Chrome introduced a series of changes to its QUIC Initial packets, making them too large to fit into a single UDP datagram. These changes to the widely used browser render the GFW less effective, as it can only block if the SNI extension appears in the first UDP datagram.

3.3 Source Port Must Exceed Destination Port

We found that sending a QUIC client Initial packet with a forbidden SNI to the server in the U.S. did not always trigger blocking. To further investigate this behavior, we conducted multiple experiments aimed at determining the specific rules the censor uses to filter QUIC connections.

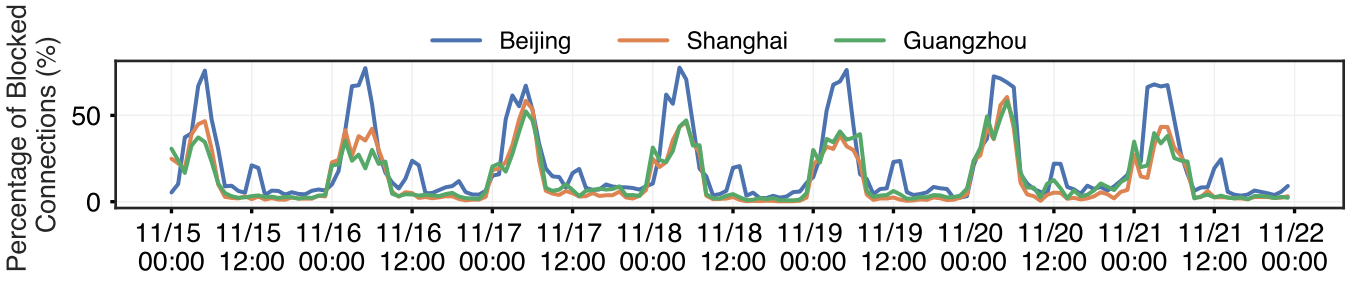

Figure 4: Percentage of blocked QUIC connections over time for clients sending connections from three major cities in China to a server in the US. The timestamps shown are in China Standard Time (CST, UTC+8) and span from November 15 to 22, 2024.

We selected a range of ports from 401 to 450 with a step size of 1. We sent QUIC client Initial messages with SNI `google.com` from our vantage point in Beijing to our U.S. server, enumerating all possible source and destination port pairs in the range. After sending each QUIC client Initial message, we waited one second, then sent five additional UDP packets–each with a unique 10-byte payload–spaced one second apart. This process was repeated ten times, using a different destination IP address (from the /28 subnet assigned to our server) in each iteration. We waited for five minutes between iterations to avoid any residual blocking from previous connections. For each port pairing, we then recorded the number of connections that were successfully received and the number that were blocked (i.e. no follow-up UDP packets received).

As shown in Figure 3, the GFW does not block connections where the source port number is less than or equal to the destination port number. However, blocking is not uniform, which suggests variability in how consistently connections are blocked. We also conducted the same experiment for the full range of ports (1 to 65535) with a step size of 1,000 and found this filtering rule to be consistent across all ports (Appendix B).

The GFW Limits the Number of Connections to Inspect. The censor applies this heuristic port checking rule to limit the number of connections it needs to inspect. Since most clients will choose a (high) ephemeral port[2] and connect to lower well-known port numbers (e.g., 443), the GFW can discard likely server-to-client traffic by ignoring packets that have a source port lower than destination port.

Two questions arise when the censor employs this rule: 1) how much traffic does the censor quickly rule out? 2) how many QUIC client Initials does the censor miss? To evaluate the efficiency and false negative rate of this rule, we collected UDP flows from a tap in a US university and analyzed the distribution of source and destination port numbers.

Table 1 shows the distribution of QUIC client Initial packets (Inits) and UDP datagrams based on source and destination ports, observed on a tap in a US university between 8:00 and 9:00 local time (Pacific Standard Time, UTC-8), on January 22, 2025. The censor only processes a packet where UDP $sport > dport$, meaning they capture more than 90% of all QUIC client Initial packets, while looking up flow table for only 30% of all UDP packets. The actual percentage of UDP payloads attempted for decryption is even lower: as detailed in Section 3.2, the GFW only parses the payload of the first UDP datagram in a flow, defined as a five-tuple (source IP, destination IP, source port, destination port, UDP protocol), that has not been seen in the last 60 seconds.

	QUIC Client Inits		UDP datagrams	
sport > dport	6.7 M	(92.3%)	3.7 B	(29.8%)
sport < dport	0.6 M	(7.6%)	8.4 B	(68.0%)
sport = dport	4.6 K	(0.06%)	27.7 M	(2.2%)

Table 1: Distribution of packet counts based on source and destination ports, observed on a tap in a US university between 8:00 and 9:00 local time (Pacific Standard Time, UTC-8), on January 22, 2025. The censor only further considers flow tracking if a UDP header has $sport > dport$, making it possible to capture more than 90% of all QUIC client Initials while looking up flow table for only 30% of all UDP packets.

3.4 Diurnal Blocking Pattern

The variability in Figure 3 indicates that connections are not consistently blocked and that blocking is non-deterministic. To explore this, we ran a week-long experiment from different vantage points to observe the frequency of QUIC connection blocking throughout the day and across all destination ports. We used our three locations in China to establish a connection to a U.S.-based server. We sent a 1,000 concurrent probes (i.e. a QUIC client Initial packet containing SNI `google.com`, followed by 1 second delay and 5 subsequent UDP packets containing unique 10-byte payloads, every 5 seconds from our three China vantage points to 10 IPv4 addresses and all ports of our server in the U.S.) In all cases, we ensured that the source port was greater than the destination port

[2]Linux hosts typically use an ephemeral port range of 32768 to 60999, while macOS and Windows Vista or later use the range 49152 to 65535. [56]

Table 2: Traceroute results identifying GFW's UDP censorship points: path to devices performing QUIC and DNS blocking, including the final uncensored hop from three different client locations.

City	Hops Away (QUIC/DNS)	Blocking Hop - 1 (ISP/AS)	Blocking Hop (ISP/AS)
Shanghai	9/9	ChinaNet Shanghai Province Network (AS4812)	ChinaNet Backbone (AS4134)
Beijing	12/12	China Unicom Backbone (AS4837)	China Unicom Backbone (AS4837)
Guangzhou	11/11	ChinaNet Guangdong Province Network (AS4134)	ChinaNet Backbone (AS4134)

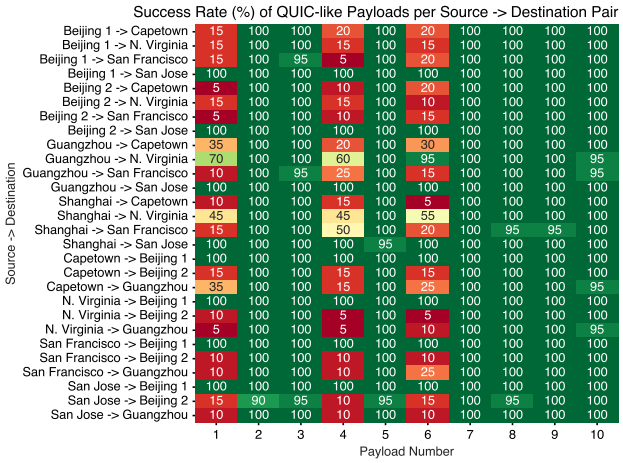

Figure 5: Percentage of blocked QUIC-like packets for each experiment run. For each payload, we created 20 connections and measured how many were received by the destination host. Each payload, described in Table 3, tests a modification to the standard QUIC client Initial and provides insight into the parsing logic of the GFW QUIC censor.

per Section 3.3. We mark a connection as censored if none of the 10-byte follow-up UDP payload packets are received by the server, after the QUIC client Initial. We then calculate the percentage of blocked connections by aggregating the data for each hour for each client location.

Blocking Rate Is Influenced by the Time of the Day. As can be seen in Figure 4, there is a clear diurnal pattern across all three cities, with blocking percentages peaking during early morning hours and dropping to the lowest levels during the day. Beijing consistently shows the highest levels of blocking, followed by Shanghai and Guangzhou. This pattern suggests that the blocking rate is influenced by the Internet usage patterns in China, with the highest blocking rates observed during periods of low network traffic.

We hypothesize that this behavior occurs because the GFW can only handle a limited volume of traffic at any given time. The operational cost of decrypting QUIC Initial packets is substantial at scale, making the blocking rate sensitive to network load, which varies during the day. Diurnal patterns of blocking have also been observed in prior studies on GFW's keyword filtering and DNS injection mechanisms [15 §3.2] [4 §7], suggesting computational limitations that render the GFW less effective during peak hours. We note that this becomes increasingly relevant in the context of QUIC connections, as parsing QUIC traffic is computationally expensive compared to other plaintext protocols like HTTP and DNS. In Section 5, we present further evidence that increasing the number of QUIC Initial packets past the GFW can overwhelm it, leading to a degradation in its censorship effectiveness.

3.5 Locating the Censorship Devices

We performed an incremental IP TTL measurement to locate the censorship devices. We set a fixed IP TTL value in the QUIC client Initial messages, starting from 1 and incrementing by 1 in each experiment. In the first second of each experiment, we sent 10 QUIC client Initial messages with the SNI of `google.com` to port 53 of our server in the US, ensuring that the blocking would be triggered as long as it reaches the censor. After 5 seconds, we then sent 100 UDP datagrams with the same 4-tuple as the QUIC client Initial message. The payload of these UDP packets consisted of 10-bytes that included the encoded TTL value used in the QUIC client Initial packets. We inferred if a QUIC Initial message reached the censor by observing if the 100 UDP datagrams were dropped. This measurement was performed from three vantage points in China: Beijing, Shanghai, and Guangzhou, with each experiment repeated 10 times.

As shown in Table 2, we found the QUIC blocking is not triggered until the IP TTL value is 9, 11, and 12 for our clients Shanghai, Beijing and Guangzhou, respectively. The hop triggering the blocking is located in the backbone network of ChinaNet for Shanghai and Guangzhou, and in the backbone network of China Unicom for Beijing.

Similarly, we sent DNS queries for `google.com` using the same 4-tuple with incrementing IP TTLs to port 53 of the server, We observed that DNS injection was not triggered until the IP TTL value matched those observed for QUIC blocking, suggesting that the new devices are co-located at the same hop as the existing GFW devices.

3.6 QUIC Parsing Idiosyncrasies

The GFW's QUIC censorship does not strictly follow the QUIC specifications [43, 58] in several ways. We crafted and sent several modified QUIC payloads that should be rejected by RFC-compliant implementations, to see if they would still

Exp. No.	Descriptions of the Tested QUIC Initial Packets	Blocked?	Degrades?
1	Packet number is one-byte.	✓	✓
2	Remove last byte from QUIC packet.	×	✓
3	Bad version number with incorrect auth tag. Version Number: `0x00000002`.	×	×
4	Both connection IDs have a length of `0x00`.	✓	✓
5	Source connection ID has a length of `0x255`.	×	✓
6	`CRYPTO` frame has a length of `0x00` but still contains a payload.	✓	✓
7	Sensitive domain in an extension other than the SNI extension (e.g. ALPN contains `google.com`).	×	✓
8	QUIC payload contains a single `CRYPTO` frame along with multiple `PING` and `PADDING` frames.	×	✓
9	A QUIC Initial packet whose TLS Client Hello contained an Encrypted Client Hello extension with an outer SNI of `cloudflare-ech.com`.	×	✓
10	A QUIC Version 2 packet.	×	×

Table 3: Description of each experiment we run to characterize the GFW's QUIC parsing mechanism. For each, we mark if the payload is ever observed to be blocked (Section 3.6), and if it can be used to degrade the GFW (Section 5).

trigger the GFW's censorship. If they do, it indicates that the GFW does not properly ignore non-compliant QUIC payloads, potentially presenting an opportunity for circumvention methods or other vulnerabilities. Our modified QUIC payloads are described in Table 3, and Figure 5 shows the results of sending these over the GFW. For each payload, we sent 20 connections in both directions—from vantage points in China to servers outside the country, and vice versa—to determine whether they would trigger censorship.

No Need for Padding. While the QUIC specification requires that Initial packets must be padded to a minimum of 1200 bytes, we found that the GFW does not enforce this requirement. We were able to trigger censorship with payloads as small as 137 bytes. However, since the GFW does not inject responses, there is not a risk of amplification attacks.

Length Field Ambiguity. Connection ID lengths are defined in the specification to be between 8 and 20 bytes; however, the field supports lengths up to 255 bytes. We find that setting both source and destination connection IDs to a length of `0x00` (too short) is blocked, though this should be ignored as per the specification. On the other hand, a length of `0xff` is not blocked, indicating that the GFW correctly checks the upper limit. Curiously, we find that the GFW will block a payload even if the `CRYPTO` frame has a specified length of `0x00`, as long as the actual payload contains a forbidden SNI. The GFW appears to assume the `CRYPTO` frame length from the rest of the payload, meaning it cannot correctly handle split `CRYPTO` frames (such as used by Google Chrome browsers).

Version Specific Blocking. Only QUIC version 1 packets containing the plaintext byte pattern `0x00000001` in the version field are subject to blocking. The recently standardized QUIC version 2 [21], which uses a different salt value for initial encryption keys, remains unblocked. This suggests the GFW either has not updated its filtering mechanisms for new version salt values or relies on version 1-specific plaintext byte pattern matching for packet inspection.

4 Monitoring the Blocklist over Time

In this section, we investigate the websites that are blocked by the GFW's QUIC-SNI censorship mechanism. We consider currently blocked sites, how the blocklist has changed over time, and how the QUIC blocklist compares to blocklists used by other censorship methods like TLS-SNI, HTTP, and DNS. As noted in Section 3.4, the GFW's QUIC censorship mechanism is non-deterministic, which requires an experimental methodology that minimizes false negatives. For each name that we test, we send QUIC client Initial messages carrying the SNI from several vantage points and repeat the process over multiple trials. Additionally, to avoid inaccuracies from residual censorship on a specific destination port, we do not send connections to the same 3-tuple (source IP, destination IP, destination port) within any 180 second window.

We monitor the GFW's QUIC blocklist over a period of more than three months. Because of the inconsistency in bidirectional blocking—specifically, that most of our vantage points stopped experiencing bidirectional blocking after September 30, 2024—we adopted an inside-out measurement approach. We deployed ten vantage points in Beijing (AS45090) to run the client-side script and a vantage point in a US university (AS32) for the server. The server was assigned a /28 IPv4 subnet. For each name to test, the client sends a QUIC client Initial message to the server, waits for one second, and then sends 5 unique 10-byte UDP payload packets spaced 1 second apart. We mark an SNI as blocked if *none* of the follow-up UDP payload packets are received.

We use the *full* Tranco list (ID: 664NX) [3] obtained on October 2, 2024, which consists of approximately seven million fully qualified domain names (FQDNs) for testing. We acknowledge that this list may not exhaustively capture all censored names. However, we argue that a reasonably large list of popular names provides a representative sample of the GFW's QUIC blocklist.

[3]The list available at: https://tranco-list.eu/list/664NX/full

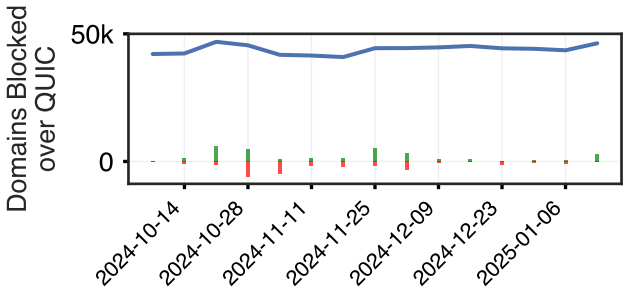

Figure 6: Number of FQDNs blocked by the GFW's QUIC SNI censorship for the full Tranco list (ID: 664NX), between October 8, 2024 and January 15, 2025. The number of blocked domains are aggregated weekly. The bar chart shows weekly churn in the blocklist over time.

In each test, for each name, each of the ten client vantage points in Beijing sends a QUIC client Initial message to a distinct IP address of our US server. Based on our finding in Section 3.3, we always use source ports greater than destination port to trigger blocking. We run these experiments as cronjobs between 3 AM and 6 AM CST. This is because we observed the highest rate of blocking during these hours (Section 3.4).

Since the blocking rate during this time is observed to be at least 50% for our connections, we repeat each QUIC client Initial test ten times to ensure that the accuracy of our blocklist extraction is above $1-(1-50\%)^{10} = 99.9\%$. On the server side, we aggregate the data for each SNI that is blocked for each day. These experiments have been running for over three months, starting from October 8, 2024, to January 15, 2025.

QUIC Blocklist. On a weekly basis, we found that the GFW blocked an average of 43.8K FQDNs from the Tranco list (Figure 6). Over the full duration of our experiments, we observed that the GFW blocked 58,207 unique FQDNs from the Tranco list (Table 4).

Fully Qualified Domain Name	Count
Total Tested (Tranco List)	6,955,968
Supporting QUIC	1,489,967
Ever Blocked over QUIC	58,207
Blocked & Supporting QUIC	38,451

Table 4: Number of Fully Qualified Domain Name (FQDNs) supporting QUIC, blocked by QUIC-SNI censorship, and their intersection. QUIC censorship test was conducted between October 8, 2024 and January 15, 2025.

Domains Blocked Over QUIC May Not Support QUIC. We tested domains for QUIC support by making direct HTTP/3 requests rather than relying on `Alt-Svc` headers, because some servers support HTTP/3-over-QUIC without advertising it. From our measurements, we identified 58,207 FQDNs that were blocked over QUIC, of which 38,451 actually support HTTP/3-over-QUIC (see Table 4). Within this larger set of blocked names, 9,345 popular second-level domains (e.g., `google.com`, `hrw.org`, `youtube.com`, `tiktok.com`) were found blocked, although only 3,233 of them actually support QUIC. Notably, a substantial number of `googlevideo.com` subdomains (35,443) appeared on the blocklist, suggesting a broader blocking rule targeting `*.googlevideo.com` and resulting in an increase the number of QUIC-supporting domains. Since not all QUIC-blocked domains actually support QUIC, it is difficult to determine the exact logic behind the GFW's blocklist. The GFW may be blocking these domains preemptively, anticipating potential future QUIC support or it may be using other criteria unrelated to QUIC for its blocking decisions.

4.1 Comparison with Other Blocklists

We conducted a comparative analysis of the GFW's QUIC-SNI blocklist against other established GFW censorship mechanisms, including TLS-SNI, HTTP Host, and DNS-based blocking. To evaluate TLS-SNI blocking, we employed a methodology based on prior work [11, 37]. We established a client in Beijing and a sink server in the U.S. to perform inside-out measurements, maintaining consistency with our QUIC-SNI blocking analysis. Our sink server was configured to accept TCP connections but not respond with any data. In each test, after completing the TCP handshake, the client transmitted TLS Client Hello messages containing test domain SNI values. We monitored the connection for TCP `RST`s packets—a characteristic signature of SNI blocking. For HTTP Host testing, we applied a similar approach but replaced the TLS Client Hello with HTTP GET requests containing the test domain in the Host header field.

For DNS censorship testing, we followed established methodologies from previous research [6, 38]. We configured our Beijing-based client to send DNS queries to a controlled US-based IP address where no DNS server was running. This configuration allowed us to definitively attribute any received DNS responses to GFW injection, as our server was configured to ignore all queries. To ensure consistent comparison across all three testing methods (TLS-SNI, HTTP Host, and DNS), we utilized domains from the same Tranco list. We performed these measurements and collected blocklisted domains over a one-week period from January 9, 2025, to January 15, 2025.

Figure 7 illustrates the overlap between the blocklists for TLS-SNI, HTTP Host, DNS, and QUIC protocols. For our tested Tranco list domains, DNS blocking affected the largest number of domains (106,973), followed by HTTP (105,488) and HTTPS (102,216). The QUIC blocklist was notably smaller, containing approximately 55 percent the number of domains compared to the other three blocklists. Among the 58,207 domains that were ever blocked over QUIC, 11,854

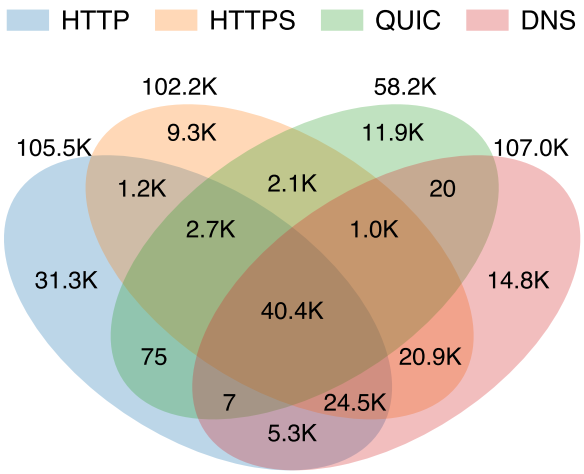

Figure 7: Venn diagram showing the overlap between the blocklists for HTTPS, HTTP, DNS, and QUIC. The blocklists for each protocol are aggregated over a period of 1 week from January 9, 2025 to January 15, 2025.

were exclusively blocked through this protocol. Notably, of these QUIC-exclusive blocked domains, we found only 2,329 domains that actually supported QUIC.

We found 40,447 domains common to all four blocklists, representing a 24.4% overlap (measured as intersection divided by union). When comparing QUIC blocking against the other three protocols individually, we found the highest overlap (intersection over union) with HTTPS at 46,251 domains (40.51%), followed by HTTP with 43,191 domains (35.84%), and DNS with 41,484 domains (33.54%). These findings indicate that each censorship mechanism operates with distinct but overlapping blocklists, creating a complementary system that maximizes the GFW's censorship coverage. For instance, an HTTP/3-over-QUIC browsing session in a modern browser typically starts with a DNS query, followed by request over HTTP/2 (or earlier) and then upgrades to HTTP/3 over QUIC. The GFW's censorship strategy is designed to affect each stage either exclusively or in combination, ensuring that the user is unable to access the forbidden content on the web.

Table 5 shows the Jaccard Index (Intersection over Union) of the GFW's blocklists for DNS-, HTTP-, TLS-, and QUIC-based censorship of the Tranco Top 10k, alongside websites supporting QUIC, and a randomly selected sample of 500 FQDNs.

5 GFW Degradation Attack

In Figure 4, we observed that the GFW's QUIC censorship was less effective during times corresponding to high traffic volume in China. This led us to hypothesize that the GFW's effectiveness could be purposefully degraded by sending QUIC packets that the GFW would need to process. While our experiments provide valuable insights into the design of China's censorship system, they also raise several ethical concerns which we carefully considered and discuss in Section 9. We designed our experiments to ensure that users and other Internet devices were not impacted, and specifically ensured our tests only degraded the GFW.

	DNS	HTTP	TLS	QUIC	Support QUIC	Sample 500
DNS	-	-	-	-	-	-
HTTP	0.57	-	-	-	-	-
TLS	0.67	0.43	-	-	-	-
QUIC	0.19	0.20	0.26	-	-	-
Support QUIC	0.19	0.20	0.13	0.05	-	-
Sample 500	0.03	0.03	0.03	0.01	0.05	-

Table 5: The Jaccard Index (Intersection over Union) of the GFW's blocklists for DNS-, HTTP-, TLS-, and QUIC-based censorship of the Tranco top 10k, alongside websites supporting QUIC, and a randomly selected sample of 500 fully qualified domain names (FQDNs).

In this experiment, we use three vantage points. The first is in China (Beijing, Alibaba, AS37963) which we refer to as ChinaVP, the second is in the US (East, Digital Ocean, AS14061) which we refer to as USVP, and the third is in a research institution in the US (University of Michigan, AS36375) which we refer to as StressVP. Our goal is to measure the effectiveness of the GFW's QUIC censorship in the presence of moderate volumes of QUIC traffic. This experiment consists of two parts that are run simultaneously: a measurement part and a stressing part.

In the measurement part, we configure our three vantage points to do the following: the ChinaVP sends a QUIC Initial packet (267 bytes of payload) containing a forbidden domain name, namely `google.com` in the SNI field, to USVP where the destination port is less than the source port (to trigger censorship as shown in Section 3). After a 1-second pause, we send 100 UDP packets in the same flow containing a fixed innocuous payload of 1,111 bytes. This process is repeated for 1000 different source-destination port pairs. We mark a connection as permitted (evaded censorship) if the server (USVP) receives the QUIC Initial packet and 95% or more of the packets that follow.

In the stressing part, we use the StressVP to send two types of traffic in varying sending rates (from 100kpps to 1500kpps, in 100kpps increments for seven minutes each and spaced by a three-minute pause) towards the IPv4 addresses in the /14 network prefix in which the ChinaVP is hosted. Our goal is to send enough traffic to stress the GFW without impacting the network link or routers. By choosing a large network to send over, the impact of our traffic is diluted for individual hosts.

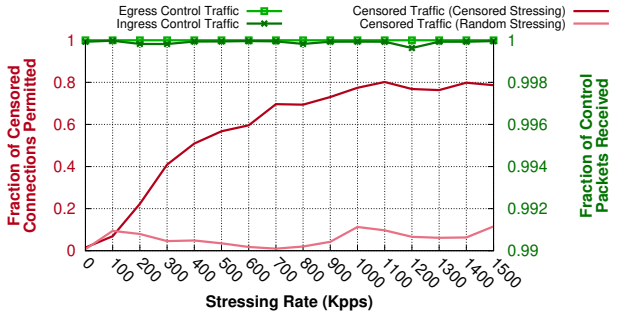

Figure 8: We stress-test the GFW by sending two types of equal-length packets at 0–1500 Kpps: QUIC Initial packets containing a forbidden SNI (Censored Stressing) and UDP packets containing a random payload (Random Stressing). We measure the effectiveness of QUIC censorship during this test by sending (at a fixed rate) QUIC Initial packets followed by 100 data packets, mimicking 1000 QUIC connections from a vantage point in China to a vantage point in the US and calculating the fraction of connections the GFW fails to censor (Censored Traffic). The GFW is less effective at censoring our measurement QUIC connections as we increase the number of QUIC Initial packets we stress-test with, increasing the difficulty on the GFW to process QUIC packets. We ensure our test impacts only the GFW and not the network by measuring the rate of uncensored QUIC traffic (Egress/Ingress Control Traffic) to and from both vantage points during our test.

Furthermore, to avoid having our stressing packets reach end-hosts, we estimate the hop-distance between the IPs in the /14 and StressVP. We run TTL-limited DNS scans using ZMap to resolve `example.com` on the entire /14. We approximate the hop-distance of each IP from StressVP by the cessation of DNS resolutions we receive from the 164 DNS servers in the /14. We then set our TTL value for our stressing packets we send from the StressVP to one less than the smallest (closest) hop-distance DNS server in the /14.

During the stressing part, we send two types of (TTL-limited) traffic past the GFW: QUIC Initial packets, and UDP packets with a fixed payload generated at random. For QUIC packets, we use the same QUIC Initial payload sent between ChinaVP and USVP containing a forbidden SNI to trigger the GFW's censorship. For the random payload we use innocuous bytes with the same length as the QUIC payload. We use ZMap [23] to send each of these payloads and configure the sending rate such that we send on average no more than 6 *pps* to each IP in the /14.

Our experiment consists of sending (TTL-limited) QUIC Initial packets from our StressVP to the /14 of ChinaVP. This is repeated three times on different days, once in an ascending order and twice in a shuffled order of sending rates. Simultaneously, we measure from ChinaVP to USVP the fraction of permitted connections. We then repeat the experiment three times (not coinciding with QUIC stressing), but send a random payload instead of QUIC Initial packets and measure the fraction of permitted connections. Figure 8 shows the impact of our experiment (averaged) on the GFW. As we increase the rate of QUIC Initial packets, the GFW is less able to censor connections between ChinaVP and USVP. We also do not see this pattern when sending random payloads, meaning that the degradation is only due to processing QUIC payloads, and not due to network volume degrading the network, as further supported by our network monitoring. Note that all the experiments were conducted in the early morning hours in China, during which the GFW is more effective at censoring QUIC traffic (see Figure 4).

Network Monitoring. While we run our experiment, we monitor the network between ChinaVP and USVP in two ways. First, we send uncensored QUIC connections in both directions, and monitor the fraction of packets received by both ends. Second, we use ZMap to scan the /14 network on `tcp/443` at a slow rate (650 *pps*), and measure the response rate in the network. If either of these metrics decreases significantly during our experiments, it may be due to a saturated network link, indicating we must halt the experiment. Since we never observed a decrease in either metric during our experiments, we believe our experiments had negligible impact on networks and devices beyond the GFW.

Reverse Engineering. In addition to helping users evade censorship, the degradation attack is also helpful in understanding the GFW's processing. For example, if a particular QUIC payload can be used to degrade the GFW's censorship effectiveness, we know it has been processed at some level by the GFW, even if that same payload is not blocked. On the other hand, if a payload has no influence on the GFW, then it is likely discarded prior to a computationally expensive step.

Table 3 shows if each payload was successful in degrading the GFW's effectiveness to censor. We tested this by sending each payload at 1200 Kpps, and observing if the fraction of permitted censored connections exceeded 60%, indicating that the payload had an impact on the GFW. These results suggest the GFW processes all payloads with the default QUIC version, and that even payloads that do not decrypt or have invalid authentication tags can degrade the GFW. However, a valid tag is necessary to trigger censorship, implying that the "slow" part of the GFW is likely the cryptographic operations in decrypting the payload.

6 Availability Attack

Prior work has shown that residual censorship can sometimes be "weaponized" by attackers to conduct availability attacks [8, 9]. In this type of attack, attacker sends a censorship-triggering request to Destination B, spoofing the source IP address to be from Victim A. If the request triggers residual blocking in a firewall between A and B, then the two hosts will be unable to communicate, as the firewall believes that

Victim A sent a forbidden request. Residual censorship is commonly on the order of 1–3 minutes [9–11, 37, 64, 66], and an attacker can simply spoof additional triggering packets during or after the residual censorship expires to keep the victim and destination blocked.

Our study represents the first known instance of the GFW implementing residual blocking for a UDP-based protocol. While the GFW has historically censored DNS traffic over UDP through spoofed packet injections, it has not employed packet dropping as a method of blocking for UDP-based protocols. However, the GFW's new QUIC blocking mechanism employs packet-dropping in a way that introduces a new vector for availability attacks, impacting all of China. In particular, an attacker could use this availability attack to block UDP connections from hosts inside China from communicating with servers outside. For example, this attack could block all open or root DNS resolvers outside of China from being accessed from within China, leading to widespread DNS failures in the country.

In this section, we investigate the practicality of this attack by performing it against our own hosts and servers.

Attack Setup. This attack requires the ability to spoof IP packets, which requires a server that is not limited by egress filtering. We obtained such a host from a public VPS provider, and verified that we can spoof IP packets and have them received within China. Inside China, we used a VPS under our control in Guangzhou, as this host experienced QUIC censorship both incoming and outgoing, meaning clients from outside China connecting to this server also experienced QUIC censorship. To simulate "victim" hosts, we acquired an AWS EC2 instance in each of the 32 regions that AWS operates in outside of China.

For each EC2 instance, we sent a DNS query to our VPS in Guangzhou. We then measured if this request was received by our VPS, indicating that the connection was initially available.

Next, from our attack machine capable of spoofing packets, we spoofed ten censored QUIC client Initial packets for each EC2 instance to our VPS in China. These packets are designed to trigger the GFW's residual censorship, between the EC2 instances's IP address and the IP:Port of the VPS in China. The path that these packets take to the VPS in China may differ significantly from the path that packets from the EC2 machine would take, meaning they may pass different GFW nodes, rendering the attack ineffective. We sent these spoofed packets every second.

Meanwhile, we measured the attack effectiveness from each EC2 instance, sending a DNS request to the VPS in China every five seconds. If the residual censorship was active on the path between instance and VPS, the request would be blocked, indicating a successful availability attack.

Table 6 and Figure 9 show the locations of the EC2 instances and the effects of the attack. Over half (17) of the 32 EC2 instances were heavily impacted by our attack. While some packets still get through for heavily impacted hosts, we find this is largely due to the timing of when the 3-minute residual censorship expires. When it does, there is up to one second before the next spoofing packet arrives to reblock the instance. Sending faster or timing with the expiration improves the blocking rate. We also observed that seven hosts were affected approximately half the time, indicating that there may be multiple network paths between a given instance and VPS, and only some of those paths experienced the residual censorship.

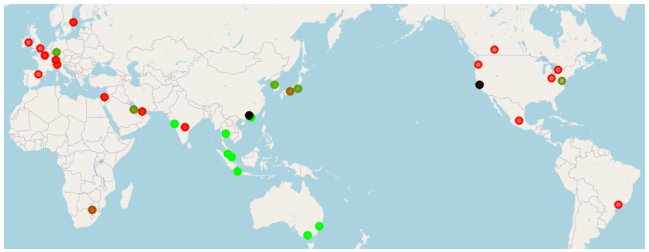

Figure 9: The map shows the locations of the EC2 instances that were affected by the availability attack. Hosts that are most affected are shaded in red, while hosts that are less affected are shaded in green. The black point in China is the location of the victim server and the black point in the US is the location of the spoofing attack server.

The remaining eight instances were not affected at all, largely in the Pacific region, suggesting that the spoofing location did not share a network path with these hosts. We confirm that all 32 hosts were capable of triggering censorship when the censored QUIC client Initial packet was sent directly from the real client.

Defense. Defending against this attack while still censoring is difficult due to the stateless nature and ease of spoofing UDP packets. One potential mitigation approach involves only triggering censorship after detecting a corresponding QUIC Server Hello and later client packets, ensuring that the connection is live and not being spoofed from one end. However, this approach has significant limitations. First, it necessitates stateful tracking of connections, which imposes substantial overhead on middleboxes. Furthermore, the inherent challenges of asymmetric routing—where the client-to-server (C2S) and server-to-client (S2C) paths differ—complicate the feasibility of accurately tracking connections. If the paths are asymmetric, a middlebox might fail to observe the Server Hello entirely, potentially leaving the connection uncensored and vulnerable to exploitation. Finally, an attacker could still spoof both sides of the connection to trigger the blocking, making it an ineffective defense.

Alternatively, the censor may employ an injection-based blocking mechanism, avoiding packet-dropping based residual censorship. However, this approach also has risks and constraints. For instance, with the current latency associated with decryption (shown in Figure 2), a QUIC Server Initial

Continent/Region	City/Area	# Packets Received	% of 360
Africa	Cape Town	110	30.56%
Asia Pacific	Hong Kong	360	100.00%
Asia Pacific	Hyderabad	13	3.61%
Asia Pacific	Jakarta	360	100.00%
Asia Pacific	Malaysia	360	100.00%
Asia Pacific	Melbourne	360	100.00%
Asia Pacific	Mumbai	360	100.00%
Asia Pacific	Osaka	145	40.28%
Asia Pacific	Seoul	246	68.33%
Asia Pacific	Singapore	360	100.00%
Asia Pacific	Sydney	360	100.00%
Asia Pacific	Thailand	360	100.00%
Asia Pacific	Tokyo	229	63.61%
Canada	Calgary	26	7.22%
Canada	Central	13	3.61%
Europe	Frankfurt	244	67.78%
Europe	Ireland	16	4.44%
Europe	London	12	3.33%
Europe	Milan	17	4.72%
Europe	Paris	10	2.78%
Europe	Spain	15	4.17%
Europe	Stockholm	14	3.89%
Europe	Zurich	17	4.72%
Israel	Tel Aviv	18	5.00%
Mexico	Central	13	3.61%
Middle East	Bahrain	201	55.83%
Middle East	UAE	22	6.11%
South America	Sao Paulo	12	3.33%
US East	N. Virginia	195	54.17%
US East	Ohio	21	5.83%
US West	N. California	21	5.83%
US West	Oregon	19	5.28%

Table 6: Shows how many packets were received by the server from each AWS region. The availability attack ran for 30 minutes and a packet was sent by the real client every five seconds. The spoofing server was in the U.S. and the victim server was in Guangzhou, China. For each AWS host, the attack server sent ten spoofed QUIC client Initial packets, each in a new connection, every second.

could reach the client and establish a shared secret, before the injected packet arrives, rendering the injection ineffective.

Defending against this attack is uniquely challenging in QUIC, because the protocol is designed to resist injection-based teardown attacks, motivating the need for residual censorship. At the same time, the connectionless nature of UDP makes spoofing the client Initial trivial, opening the door for availability attacks when residual censorship is employed. Careful engineering will be needed to allow censors to apply targeted blocks in QUIC, while simultaneously preventing availability attacks.

7 Circumvention

As described in Section 3.2, the GFW makes several simplifying assumptions to efficiently parse and block QUIC traffic at line speed. These design choices again demonstrate that the designers and developers of the GFW follow the "worse-is-better" philosophy [31]. Such assumptions come at the cost of reducing the censorship system's accuracy and robustness, which opens up opportunities for circumvention. We responsibly disclosed circumvention strategies we identified to the anti-censorship and open-source communities.

Use Source Port <= Destination Port. As detailed in Section 3.4, the GFW focuses on client-to-server traffic by ignoring UDP datagrams whose $srcport <= dstport$. A stopgap solution to bypass this blocking is to use destination ports that are higher than or equal to the source port. For the case of circumvention proxies, one may run the server on a port higher than or equal to the client's ephemeral port range. Web services could also be run on non-standard higher ports, and provide these to web clients via `Alt-Svc` fields in HTTP headers or via DNS HTTPS records. An easy and application-independent way to listen on a higher port is to use iptables rules to redirect all traffic sent to a higher port (e.g. 65535) to the current listening port (e.g. 443) using `iptables -t nat -A PREROUTING -p udp --dport 65535 -j REDIRECT --to-port 443`. This is especially useful for software that cannot change its listening port or cannot listen on multiple ports.

Precede QUIC Client Initial With Any UDP Datagram. The GFW's QUIC censorship mechanism relies on the assumption that the QUIC client Initial is the first packet in a new flow. A simple way to bypass this is to precede the client Initial with a UDP datagram with a random payload. For a real QUIC server, the first UDP datagram will be ignored, but the GFW will not be able to parse the SNI value from the first packet and exempt the flow. The subsequent client Initial packets will not be inspected and the connection will be established. We confirmed this defense works against the GFW to exempt connections from blocking by sending a UDP datagram with random payload before the QUIC client Initial. We also tested against the Chromium Quiche [35] QUIC server implementation to verify it ignores random UDP payloads.

Connection Migration. QUIC's connection migration capability leverages connection IDs to maintain sessions across network changes. The GFW employs a selective filtering strategy: it permits the initial QUIC packet but blocks subsequent packets from client to server, while not monitoring connection IDs. Since server-to-client packets remain unblocked, clients that complete 1-RTT handshakes before the blocking is activated and then migrate to a different network 4-tuple (source IP, source port, destination IP, destination port) can bypass the GFW.

A related approach is presented in QUICstep [44], which introduces a connection migration technique designed to circumvent QUIC censorship. This method exploits QUIC's connection migration capability by performing the QUIC handshake over a secure channel, which may have low bandwidth and high latency. After successfully completing the handshake, the connection migrates to a regular communication channel so that all data is fully encrypted.

QUIC Client Initial Fragmentation. A QUIC client Initial message can be sent either as multiple UDP datagrams or as a single UDP datagram containing multiple QUIC frames. As of January 2025, the GFW does not reassemble a TLS Client Hello when it is split across multiple UDP datagrams or fragmented into multiple QUIC frames within a single UDP datagram. This behavior can be leveraged to circumvent the GFW's QUIC censorship by splitting the SNI across multiple QUIC `CRYPTO` frames in the client Initial message.

Notably, Chrome's Chaos Protection mechanism [19], introduced in 2021, disperses the QUIC client Initial message into multiple QUIC frames that are shuffled across the UDP datagrams. Additionally, Chrome (since version 124 [12]) supports post-quantum key agreement in TLS 1.3, that enables the use of ML-KEM and Kyber keys. Enabling this feature fragments the QUIC client Initial into multiple UDP datagrams due to the larger key size exceeding the maximum QUIC packet size. These features happen to exploit the GFW's inability to reassemble fragmented QUIC client Initial packets, allowing Chrome packets to bypass the GFW's QUIC censorship.

Encrypted Client Hello (ECH). ECH [51] allows a client to encrypt part of their TLS Client Hello message to a server with a key obtained via DNS HTTPS record. The SNI extension is thus encrypted, allowing a client to hide it from a censor. Unlike QUIC's client Initial encryption, ECH encryption is asymmetric and cannot be decrypted by a network observer.

A censor could choose to block all ECH-containing payloads. However, modern browsers have started to send "dummy" ECH payloads in TLS, even when a server does not support it. As of January 2025, the GFW does not block QUIC payloads that contain ECH, unless the outer (decryptable) SNI is to a blocked domain.

Version Negotiation. QUIC's version negotiation [43 §6] mechanism presents an interesting circumvention opportunity. This process typically begins when a server receives an Initial packet with an unsupported version number. In response, the server sends a Version Negotiation packet and waits for the client to submit a new Initial packet using a supported version. A client can strategically exploit this mechanism by deliberately sending an Initial packet with an unknown version, making the payload of the first packet undecryptable. As a result, subsequent packets in the connection flow are able to bypass the GFW's filtering mechanisms. The client can then proceed with the handshake using a supported version, effectively circumventing the censorship measures.

Are "Stopgap" Solutions Worth Deploying? While many of these solutions opportunistically exploit implementation details in the GFW, it may not be trivial for China to patch all of these, due to resource constraints and other priorities [5]. In past work, we have seen similar stopgap solutions work for multiple years against censors [66 §8.3]. On the other hand, many of these circumvention strategies can be easily deployed by QUIC-using proxies and circumvention tools, who do not face the same kinds of bureaucratic constraints [5].

Responsible Disclosure. We shared our findings on China's QUIC censorship and the circumvention strategies with the anti-censorship and open-source communities. In specific, we contacted the developers of Mozilla Firefox [48], Mozilla Neqo library [49], quic-go library [53], Lantern [18], Hysteria [39], TUIC [59], sing-box [54], V2Ray [61], and Xray [68].

The SNI-slicing feature (implemented through client Initial Fragmentation) was included in the Neqo v0.12.0 release on January 27, 2025 [24, 50]. Mozilla Firefox has since integrated this feature and shipped it as a default feature in version 137 on April 30, 2025 [24, 47] (configurable via the `network.http.http3.sni-slicing` parameter in the `about:config` page).

Table 7: Integration timeline for quic-go v0.52.0 [53]. Following its release on May 23, 2025, popular circumvention tools updated their dependency, which enables SNI slicing by default to bypass GFW's QUIC SNI-based censorship.

Project	Version	Release Date
sing-box	1.12.0-beta.17	May 22, 2025 [55]
V2Ray	5.33.0	May 26, 2025 [62]
Xray	25.6.8	June 6, 2025 [69]
Hysteria	2.6.2	June 7, 2025 [40]

The quic-go library introduced SNI-slicing in its v0.52.0 release [53] on May 23, 2025 [53]. As summarized in Table 7, this update allows circumvention tools that depend on quic-go to bypass the GFW's QUIC SNI-based censorship.

As of June 2025, we are working with the Mozilla Neqo and Firefox team to integrate a complementary circumvention technique (prepending dummy payload before the handshake) in Mozilla Firefox for more resilience against the GFW [72].

8 Discussion

Our findings raise two crucial questions about the GFW's blocking of QUIC connections: (1) its impact on regular web traffic, and (2) its implications for QUIC-based proxying. When accessing websites, browsers will first connect to servers using HTTP(S)-over-TCP, and only attempt to use

QUIC if the server announces to support it (via the Alternate Service header). Consequently, the HTTP Host-based and TLS SNI-based blocking are still the primary mechanisms for blocking web traffic, and only when a website is not censored by these two mechanisms will the GFW's QUIC blocking come into play. The GFW's QUIC blocking essentially acts as a secondary censorship mechanism for web traffic.

Focusing on QUIC-based proxies, the growing popularity of tools like Hysteria [39] and ongoing standardization efforts—particularly by the IETF's MASQUE [41] working group—shows the protocol's future potential for VPNs and proxies. QUIC's flow-controlled and multiplexed streams, rapid connection establishment, and support for connection migration offer significant performance gains. By using unprompted authentication in HTTP/3 servers, QUIC tunnels can reside in mainstream HTTP/3 traffic and potentially elude detection even through active probing. However, our results show that the GFW's SNI-based filtering undermines these advantages early in the handshake process, effectively blocking many QUIC proxies at the outset.

A clear example is Cloudflare's WARP VPN, which recently started using MASQUE [17] (HTTP/3-over-QUIC proxying) to tunnel traffic. We discovered that the subdomain used for MASQUE was blocked by the GFW, disrupting the VPN client's startup handshake. This pattern signals an explicit targeting of MASQUE proxies by the GFW. Similarly, Hysteria faces a situation where not only its main project domain `v2.hysteria.network` is blocked, but users' custom domains used for Hysteria proxies are also blocked.

9 Conclusion

In response to the GFW's QUIC SNI-based censorship from April 7, 2024, we conducted measurement experiments to characterize, monitor, expose, and bypass it. We show this new blocking mechanism can be exploited to block arbitrary UDP traffic between hosts inside and outside China. We also propose an off-path circumvention strategy which reduces the GFW's effectiveness with moderate traffic loads. We collaborate with various open-source communities to integrate circumvention strategies into Mozilla Firefox, the quic-go library, and all major QUIC-based circumvention tools.

Acknowledgments

We thank the anonymous shepherd and reviewers for their valuable feedback. Our research was initiated by brave Chinese users who first reported the QUIC censorship, and we are deeply grateful for their courage. We extend our special thanks to the anonymous RQWDKM, who conducted initial measurements and investigation with us, participated in discussions, and provided detailed feedback on this paper.

We are indebted to the many people at Mozilla Neqo and Firefox teams for their invaluable discussions, support, and implementation of circumvention solutions. Our thanks include, but are not limited to, Christoph Kerschbaumer, Kershaw Jang, Lars Eggert, Martin Thomson, Max Inden, and Valentin Gosu.

We particularly thank Marten Seemann for integrating circumvention solutions into quic-go. We are also deeply grateful to Adam Fisk from Lantern for his support and coordination in this effort. We also thank the developers and contributors of Hysteria, Lantern, sing-box, TUIC, V2Ray, and Xray, for providing online discussion spaces and for their rapid adoption of circumvention strategies.

Finally, we are grateful to the following individuals, along with many others who prefer to remain anonymous, for their support, feedback, and insightful discussions: Bill Marczak, Dan Boneh, David Fifield, David Mazieres, Jeffrey Knockel, Juraj Somorovsky, klzgrad, nekohasekai, Nick Sullivan, Niklas Niere, and Prateek Mittal.

This work was partially supported by the U.S. National Science Foundation (NSF) under grant number CNS-2145783, until the award was prematurely terminated as part of the agency's shift in priorities [30], significantly impacting ongoing research efforts.

The work was also supported in part by the NSF under grant numbers CNS-2319080 and CNS-2333965, by a Sloan Research Fellowship, and by the Young Faculty Award program of the Defense Advanced Research Projects Agency (DARPA) under the grant DARPA-RA-21-03-09-YFA9-FP-003. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Ethics Considerations

There are two major ethical considerations in our work: the potential harm of our experiments on network infrastructure and the disclosure of uncovered weaknesses.

Availability Attack. In Section 6, we demonstrated that the GFW could be co-opted to conduct an availability attack against arbitrary Internet hosts. We demonstrated this attack against our own servers to limit the risk of unforeseen collateral harm during our experiments. While the attack does involve spoofing IP packets, the only IP addresses we pretended to be were under our control already. The result of this attack is that for a brief duration, our own EC2 instances were unable to communicate with our server in China.

We also analyze our work under the lens of two ethical frameworks suggested by prior work on computer security and ethics [45]. From a consequentialist ethical perspective, there is negligible risk of harm from this attack. From a deontological view, the choice of attacking only our own hosts limits the involvement of others, and thus our duty to others.

GFW Degradation Attack. In Section 5, we introduced a method to degrade the GFW's ability to block QUIC connections by sending a large number of QUIC Initial packets. There are several risks associated with this experiment that influenced our experiment design. First, we considered whether it is morally justified to disrupt the GFW, a system that is itself a source of harm [28 §9.c] [28 §9.B] [3,42], as acknowledged even by its creators [71 §1]. On the one hand, the GFW is not a system we control and disrupting it could have negative or unseen consequences. On the other hand, causing the GFW to fail to censor provides a benefit to Chinese users, as their Internet is otherwise restricted in opposition to their human rights [60]. These considerations led us to conclude that as long as the risk to systems beyond the GFW was minimal, disrupting the GFW itself was morally justified.

However, it is vitally important to consider the risk to other systems. For example, if disrupting the GFW caused *all* traffic to be dropped, our experiment would risk interfering with normal Internet communications between China and the rest of the world. Indeed, while our analysis in Section 3.1 suggests that the GFW's QUIC censorship is not purely in-path, we still worried that its in-path element might impact all traffic. However, our observation of the diurnal pattern of censorship effectiveness gave us strong evidence that this was not the case. During the day when QUIC connection volumes are high, the GFW is able to block only a small fraction of connections (Figure 4), but uncensored QUIC and other types of traffic remain unaffected.

Finally, we considered the risk that our experiments could disrupt the network itself. Since we send a large number of QUIC packets, there was the possibility of overwhelming a network link or destination. We took several steps to mitigate this risk. First, we limited our sending rate to 1.5 million packets/second, which consumes under 4 Gbps of bandwidth. We confirmed that our connection to upstream Internet providers was at least 40 Gbps, and transnational links are typically 100 Gbps or multi-Tbps, meaning our traffic would be only a small fraction of their capacity. Second, we limited the TTL of packets to ensure that they would pass the GFW but not reach the destination network. This approach limits the impact to only large core Internet links which can easily handle this relatively minor traffic volume. Third, we continuously monitored several health metrics across the networks we tested, including ZMap scans and bidirectional connectivity tests. We observed no network degradation during our experiments, indicating that we did not overwhelm the network.

From a deontological perspective [60 §4.1], we must consider the rights of others (e.g. Internet users in China), as well as our intentions during the study. From this view, our research methodology confronts a direct conflict between two moral duties. On the one hand, we are obligated to avoid (potentially) interfering with the network resources of others. On the other hand, our experiments also fulfill a duty to prevent an ongoing harm: namely, the censorship of the GFW. We argue that the latter constitutes a higher moral imperative and thus decide to proceed with our experiments. From a consequential perspective [60 §4.1], we must weigh the benefits against the harms. The benefits are that our attacks reveal a way to restore users' access to information, while minimizing the risk of harm to other networks and hosts.

Disclosure. Vulnerability disclosure is a standard practice for ethical security research, as it helps improve the system under study and to protect individuals impacted by the vulnerable system from attacks. In our case, disclosure is ethically complex because we are studying a system that would be harmful to improve (the GFW). On the other hand, it is important to protect Internet users that may be subject to attacks through vulnerabilities in the GFW. We carefully considered what—if any—vulnerabilities to disclose to protect users, but not improve the GFW's ability to censor. Our goal is to maximize benefits by protecting users, while minimizing the risk of harm in "helping" China strengthen their censorship.

Given these considerations, we decided to disclose the availability attack (Section 6) to the censor, as it can be used to harm users. On Jan. 22, 2025, we disclosed this vulnerability to CNCERT and Fang Binxing—widely recognized as "the father of the GFW" [34]—and recommended that the vulnerable QUIC censorship device be removed. A copy of the email is included in Appendix C. To ensure clarity, we contacted the censors via an email in both English and Chinese, and provided links to two private webpages (one in each language) that detailed the attack. Although we did not receive any response or formal acknowledgment, we observed a total of 37 visits to the private English webpage (and none to the Chinese version) between Jan. 24 and Feb. 24, 2025, suggesting that our message was received. This lack of direct engagement from CNCERT shows the challenge of vulnerability disclosure with Internet censors [9 §VIII]. Chinese authorities rarely admit the existence of censorship [57], let alone acknowledge its risks or consider dismantling their censorship systems.

However, starting Mar. 13, 2025, we observed a change in the GFW's behavior: QUIC traffic originating from outside China could no longer trigger the blocking. This change partially mitigates the vulnerability, as the availability attack can no longer be launched from outside China. It's unclear whether this change was due to our disclosure, though a similar change has been observed in the past following a public disclosure of the GFW's ESNI censorship [10].

Despite the mitigation, the availability attack remains viable if launched within China. An attacker operating a machine in China (without egress filtering) can still block arbitrary UDP flows between a host in China and any destination outside China, if the attacker's network path traverses the same GFW node as the victim's. Since the Chinese censor is unlikely to remove the QUIC censorship devices, which is the only way to fully mitigate this vulnerability, our risk mitigation strategy centers on public transparency. By publishing this paper, we hope to disclose and publicize the vulnerability

to raise broader awareness about the security implications and potential harms of large-scale censorship systems [28].

We chose not to inform the censor directly about the degradation attack (Section 5). Instead, we first privately disclosed the vulnerability to anti-censorship communities, followed by a public disclosure with this paper's publication. We chose this disclosure strategy because the degradation attack affects only the GFW's infrastructure, not users. A private disclosure to the censor would have afforded them an opportunity to strengthen their censorship mechanisms before the broader anti-censorship community could become aware of and learn from this vulnerability.

While publicizing this vulnerability might motivate the censor to fix a weakness they likely already knew existed (now knowing others are also aware), we believe the value of this public disclosure outweighs such risks. By sharing these insights on the censor's weaknesses with a broader audience, we can better inform future protocol designs and anti-censorship strategies. For example, the QUIC Initial packet is designed to be encrypted, despite being decryptable by middleboxes, partly to complicate their ability to process it. The GFW's QUIC censorship system struggled to keep pace with the decryption, demonstrating that even design choices that slightly raise the processing cost can still reduce a censor's overall effectiveness [5].

No Collection of PII. Our work does not involve human subjects, and we did not collect any personally identifiable information (PII) in any of our data.

Open science

To encourage future work and maintain reproducibility, we have publicly released the code and data from our study. For broader accessibility, this paper is also available in HTML format in both English and Chinese. The project homepage is at: https://gfw.report/publications/usenixsecurity25/en/.

References

[1] D. Adrian. A new path for kyber on the web. URL: https://security.googleblog.com/2024/09/a-new-path-for-kyber-on-web.html.

[2] Alice, Bob, Carol, J. Beznazwy, and A. Houmansadr. How China detects and blocks Shadowsocks. In *Internet Measurement Conference*. ACM, 2020. URL: https://censorbib.nymity.ch/pdf/Alice2020a.pdf.

[3] D. Anderson. Splinternet behind the Great Firewall of China: Once China opened its door to the world, it could not close it again. *Queue*, 10(11):40–49, November 2012. URL: https://queue.acm.org/detail.cfm?id=2405036, doi:10.1145/2390756.2405036.

[4] Anonymous. Towards a comprehensive picture of the Great Firewall's DNS censorship. In *Free and Open Communications on the Internet*. USENIX, 2014. URL: https://www.usenix.org/system/files/conference/foci14/foci14-anonymous.pdf.

[5] Anonymous and Anonymous. Sharing a modified Shadowsocks as well as our thoughts on the cat-and-mouse game, October 2022. URL: https://github.com/net4people/bbs/issues/136.

[6] Anonymous, A. A. Niaki, N. P. Hoang, P. Gill, and A. Houmansadr. Triplet censors: Demystifying Great Firewall's DNS censorship behavior. In *Free and Open Communications on the Internet*. USENIX, 2020. URL: https://www.usenix.org/system/files/foci20-paper-anonymous_0.pdf.

[7] M. Bishop. HTTP/3. RFC 9114, June 2022. URL: https://www.rfc-editor.org/info/rfc9114, doi:10.17487/RFC9114.

[8] K. Bock, A. Alaraj, Y. Fax, K. Hurley, E. Wustrow, and D. Levin. Weaponizing middleboxes for TCP reflected amplification. In *USENIX Security Symposium*. USENIX, 2021. URL: https://www.usenix.org/system/files/sec21-bock.pdf.

[9] K. Bock, P. Bharadwaj, J. Singh, and D. Levin. Your censor is my censor: Weaponizing censorship infrastructure for availability attacks. In *Workshop on Offensive Technologies*. IEEE, 2021. URL: https://www.cs.umd.edu/~dml/papers/weaponizing_woot21.pdf.

[10] K. Bock, iyouport, Anonymous, L.-H. Merino, D. Fifield, A. Houmansadr, and D. Levin. Exposing and circumventing China's censorship of ESNI, August 2020. URL: https://github.com/net4people/bbs/issues/43#issuecomment-673322409.

[11] Z. Chai, A. Ghafari, and A. Houmansadr. On the importance of encrypted-SNI (ESNI) to censorship circumvention. In *Free and Open Communications on the Internet*. USENIX, 2019. URL: https://www.usenix.org/system/files/foci19-paper_chai_update.pdf.

[12] Chrome Developers. Chrome 124 — release notes, April 2024. URL: https://developer.chrome.com/release-notes/124.

[13] R. Clayton, S. J. Murdoch, and R. N. M. Watson. Ignoring the Great Firewall of China. In *Privacy Enhancing Technologies*, pages 20–35. Springer, 2006. URL: https://www.cl.cam.ac.uk/~rnc1/ignoring.pdf.

[14] Cloudflare. Cloudflare Radar - Adoption and Usage Worldwide, 2025. URL: https://radar.cloudflare.com/adoption-and-usage?dateStart=2024-01-01&dateEnd=2024-12-31.

[15] J. R. Crandall, D. Zinn, M. Byrd, E. Barr, and R. East. ConceptDoppler: A weather tracker for Internet censorship. In *Computer and Communications Security*, pages 352–365. ACM, 2007. URL: http://www.csd.uoc.gr/~hy558/papers/conceptdoppler.pdf.

[16] critical_error. QUIC streams with encrypted_client_hello extensions in QUIC initials are being blocked in Uzbekistan. NTC Party Forum, 12 2024. URL: https://ntc.party/t/13953.

[17] Dan Hall. Zero Trust WARP: tunneling with a MASQUE. URL: https://blog.cloudflare.com/zero-trust-warp-with-a-masque/.

[18] L. developers. Lantern. URL: https://github.com/getlantern.

[19] dschinazi. Chaos Protection in QUIC, 2025. URL: https://quiche.googlesource.com/quiche/+/cb6b51054274cb2c939264faf34a1776e0a5bab7.

[20] H. Duan, N. Weaver, Z. Zhao, M. Hu, J. Liang, J. Jiang, K. Li, and V. Paxson. Hold-On: Protecting against on-path DNS poisoning. In *Securing and Trusting Internet Names*. National Physical Laboratory, 2012. URL: https://www.icir.org/vern/papers/hold-on.satin12.pdf.

[21] M. Duke. QUIC Version 2. RFC 9369, May 2023. URL: https://www.rfc-editor.org/info/rfc9369.

[22] A. Dunna, C. O'Brien, and P. Gill. Analyzing China's blocking of unpublished Tor bridges. In *Free and Open Communications on the Internet*. USENIX, 2018. URL: https://www.usenix.org/system/files/conference/foci18/foci18-paper-dunna.pdf.

[23] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast internet-wide scanning and its security applications. In *USENIX Security Symposium*. USENIX, August 2013. URL: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric.

[24] L. Eggert. Pull request #2228: *feat: Shuffle the client Initial crypto data*. https://github.com/mozilla/neqo/pull/2228.

[25] K. Elmenhorst, B. Schütz, N. Aschenbruck, and S. Basso. Web censorship measurements of HTTP/3 over QUIC. In *Internet Measurement Conference*. ACM, 2021. URL: https://dl.acm.org/doi/pdf/10.1145/3487552.3487836.

[26] R. Ensafi, D. Fifield, P. Winter, N. Feamster, N. Weaver, and V. Paxson. Examining how the Great Firewall discovers hidden circumvention servers. In *Internet Measurement Conference*. ACM, 2015. URL: https://conferences2.sigcomm.org/imc/2015/papers/p445.pdf.

[27] A. L. et al. The quic transport protocol: Design and internet-scale deployment. SIGCOMM '17. ACM, 2017. doi:10.1145/3098822.3098842.

[28] S. Fan, J. Sippe, S. San, J. Sheffey, D. Fifield, A. Houmansadr, E. Wedwards, and E. Wustrow. Wallbleed: A memory disclosure vulnerability in the Great Firewall of China. In *Network and Distributed System Security*. The Internet Society, 2025. URL: https://gfw.report/publications/ndss25/data/paper/wallbleed.pdf.

[29] O. Farnan, A. Darer, and J. Wright. Poisoning the well – exploring the Great Firewall's poisoned DNS responses. In *Workshop on Privacy in the Electronic Society*. ACM, 2016. URL: https://dl.acm.org/authorize?N25517.

[30] N. S. Foundation. Updates on nsf priorities. https://www.nsf.gov/updates-on-priorities, 2025.

[31] R. P. Gabriel. Worse is better. URL: https://dreamsongs.com/WorseIsBetter.html.

[32] gfw-report. Rapid blocking of connections following ESNI triggers, August 2020. URL: https://github.com/net4people/bbs/issues/43#issuecomment-673490763.

[33] P. God. QUIC connection with SNI of *.eu.org has been blocked. Telegram post, 2024. URL: https://t.me/c/1166154022/909198.

[34] J. Goldkorn. Fang Binxing and the Great Firewall. In G. R. Barmé and J. Goldkorn, editors, *China Story Yearbook 2013: Civilising China*. Australian Centre on China in the World. URL: https://www.thechinastory.org/yearbooks/yearbook-2013/chapter-6-chinas-internet-a-civilising-process/fang-binxing-and-the-great-firewall/.

[35] Google. QUICHE: QUIC, HTTP/2, HTTP/3 and related protocol toolkit. URL: https://github.com/google/quiche.

[36] L. Hanson. The chinese internet gets a stronger backbone. https://www.forbes.com/sites/lisachanson/2015/02/24/the-chinese-internet-gets-a-stronger-backbone.

[37] N. P. Hoang, J. Dalek, M. Crete-Nishihata, N. Christin, V. Yegneswaran, M. Polychronakis, and N. Feamster. GFWeb: Measuring the Great Firewall's Web censorship at scale. In *USENIX Security Symposium*. USENIX, 2024. URL: https://www.usenix.org/system/files/sec24fall-prepub-310-hoang.pdf.

[38] N. P. Hoang, A. A. Niaki, J. Dalek, J. Knockel, P. Lin, B. Marczak, M. Crete-Nishihata, P. Gill, and M. Polychronakis. How great is the Great

Firewall? Measuring China's DNS censorship. In *USENIX Security Symposium*. USENIX, 2021. URL: https://www.usenix.org/system/files/sec21-hoang.pdf.

[39] Hysteria developers. Hysteria. URL: https://github.com/apernet/hysteria.

[40] Hysteria Developers. Hysteria software release. URL: https://github.com/apernet/hysteria/releases/tag/app%2Fv2.6.2.

[41] IETF. Multiplexed Application Substrate over QUIC Encryption (masque), 2025. URL: https://datatracker.ietf.org/wg/masque/about/.

[42] Internet Society. When is the Internet not the Internet?, December 2023. URL: https://www.internetsociety.org/resources/internet-fragmentation/the-chinese-firewall/.

[43] J. Iyengar and M. Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021. URL: https://www.rfc-editor.org/info/rfc9000, doi:10.17487/RFC9000.

[44] W. Jia, M. Wang, L. Wang, and P. Mittal. QUICstep: Circumventing QUIC-based censorship, 2023. URL: https://arxiv.org/abs/2304.01073.

[45] T. Kohno, Y. Acar, and W. Loh. Ethical frameworks and computer security trolley problems: Foundations for conversations. In *32nd USENIX Security Symposium (USENIX Security 23)*, 2023. URL: https://securityethics.cs.washington.edu/.

[46] madeye. Savoury implementation of the QUIC transport protocol and HTTP/3, 2024. URL: https://github.com/cloudflare/quiche.

[47] Mozilla Developers. Bug 1942325 - update Neqo to v0.12.2 in mozilla-central. https://bugzilla.mozilla.org/show_bug.cgi?id=1942325.

[48] Mozilla Foundation. Firefox Web Browser Source Code. https://github.com/mozilla-firefox/firefox.

[49] Mozilla Foundation. Neqo: Next Generation QUIC Client and Server Library. https://github.com/mozilla/neqo.

[50] Mozilla Neqo Team. Neqo version 0.12.0 release. https://github.com/mozilla/neqo/releases/tag/v0.12.0.

[51] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood. TLS Encrypted Client Hello. Internet-draft, March 2025. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-ietf-tls-esni/24/.

[52] Sakamoto and E. Wedwards. Bleeding wall: A hematologic examination on the Great Firewall. In *Free and Open Communications on the Internet*, 2024. URL: https://www.petsymposium.org/foci/2024/foci-2024-0002.pdf.

[53] M. Seemann. quic-go: A QUIC implementation in pure Go (version 0.52.0). https://github.com/quic-go/quic-go/releases/tag/v0.52.0.

[54] Sing-box developers. Sing-box. URL: https://github.com/SagerNet/sing-boxh.

[55] sing-box Developers. sing-box software release. URL: https://github.com/SagerNet/sing-box/releases/tag/v1.12.0-beta.17.

[56] N. Software. The ephemeral port range. URL: https://www.ncftp.com/ncftpd/doc/misc/ephemeral_ports.html.

[57] M. Streisand, E. Wustrow, and A. Houmansadr. Where have all the paragraphs gone? detecting and exposing censorship in Chinese translation. In *Free and Open Communications on the Internet*, 2023. URL: https://www.petsymposium.org/foci/2023/foci-2023-0001.pdf.

[58] M. Thomson and S. Turner. Using TLS to Secure QUIC. RFC 9001. URL: https://www.rfc-editor.org/info/rfc9001.

[59] TUIC Protocol. tuic: Delicately-tuiced 0-rtt proxy protocol. https://github.com/tuic-protocol/tuic.

[60] United Nations Human Rights Council. The promotion, protection and enjoyment of human rights on the Internet. https://www.article19.org/data/files/Internet_Statement_Adopted.pdf, June 2016. Resolution A/HRC/32/L.20.

[61] V2Ray developers. V2Ray. URL: https://github.com/v2fly/v2ray-core.

[62] V2Ray Developers. V2Ray Core software release. URL: https://github.com/v2fly/v2ray-core/releases/tag/v5.33.0.

[63] ValdikSS. Restriction HTTP/3 (QUIC) - post 10. ntc.party, Mar 2022. Accessed: 2024-05-27. URL: https://ntc.party/t/1823/10.

[64] Z. Wang, Y. Cao, Z. Qian, C. Song, and S. V. Krishnamurthy. Your state is not mine: A closer look at evading stateful Internet censorship. In *Internet Measurement Conference*. ACM, 2017. URL: https://www.cs.ucr.edu/~krish/imc17.pdf.

[65] P. Winter and S. Lindskog. How the Great Firewall of China is blocking Tor. In *Free and Open Communications on the Internet*. USENIX, 2012. URL: https://www.usenix.org/system/files/conference/foci12/foci12-final2.pdf.

[66] M. Wu, J. Sippe, D. Sivakumar, J. Burg, P. Anderson, X. Wang, K. Bock, A. Houmansadr, D. Levin, and E. Wustrow. How the Great Firewall of China detects and blocks fully encrypted traffic. In *USENIX Security Symposium*. USENIX, 2023. URL: https://www.usenix.org/system/files/sec23fall-prepub-234-wu-mingshi.pdf.

[67] M. Wu, A. Zohaib, Z. Durumeric, A. Houmansadr, and E. Wustrow. A wall behind a wall: Emerging regional censorship in China. In *Symposium on Security & Privacy*. IEEE, 2025. URL: https://gfw.report/publications/sp25/data/paper/paper.pdf.

[68] XRay developers. XRay. URL: https://github.com/XTLS/Xray-core.

[69] Xray Developers. Xray software release. URL: https://github.com/XTLS/Xray-core/releases/tag/v25.6.8.

[70] D. Xue, B. Mixon-Baca, ValdikSS, A. Ablove, B. Kujath, J. R. Crandall, and R. Ensafi. TSPU: Russia's decentralized censorship system. In *Internet Measurement Conference*. ACM, 2022. URL: https://dl.acm.org/doi/pdf/10.1145/3517745.3561461.

[71] B. Yan, B. Fang, B. Li, and Y. Wang. Detection and defence of DNS spoofing attack, November 2006. URL: https://github.com/user-attachments/files/18172972/Yan2006a.pdf.

[72] A. Zohaib. Adding robustness checks to the QUIC handshake. https://github.com/mozilla/neqo/pull/2613.

A Blocking Latency Across the Day

Figure 10 shows how the GFW's blocking latency varies across the day. Blocking latencies are the time taken for the GFW to block a connection after observing a QUIC Initial packet with a blocked SNI. It is measured as the time difference between the time the client sends the QUIC Initial packet and the time the client sends the first UDP datagram that gets dropped by the GFW.

The minimum blocking latencies are consistently below 100 ms during the day, likely bounded by the GFW's internal processing and reaction speed.

The maximum blocking latencies vary throughout the day, potentially influenced by the number of QUIC connections being processed by the GFW (as also hinted in Section 5). During periods of generally lower human activity, typically in the early morning hours (12 AM to 6 AM), it takes relatively less time for the GFW to block connections, with a mean blocking latency of approximately 150 ms. In contrast, during peak human activity hours (7 AM to 11 PM), the mean blocking latency can go up to 800 ms, with a max blocking latency of 7,000 ms observed at around 3 PM.

B Port-based Traffic Filtering

To further confirm our findings from Section 3.3 regarding the GFW's filtering heuristic based on the source and destination ports, we extended our analysis to a wider range of ports. Using the same methodology, we examined a wider range of ports, from 1 to 65535, with a step size of 1,000. We also included the port 65535 in our test and analysis.

Figure 11 illustrates the GFW's blocking behavior across this expanded port range. These results corroborate our initial findings: the GFW does not track or block UDP flows if the source port of the QUIC Initial packet is less than or equal to its destination port.

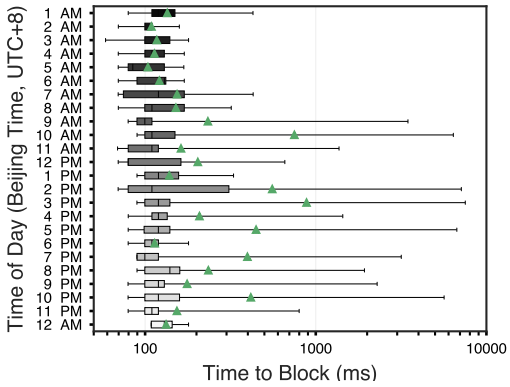

Figure 10: The box plot shows the distribution of the time taken for the GFW to block a connection. The x-axis is in log scale. The green triangle marks the mean value; and the whiskers shows the minimum and maximum values.

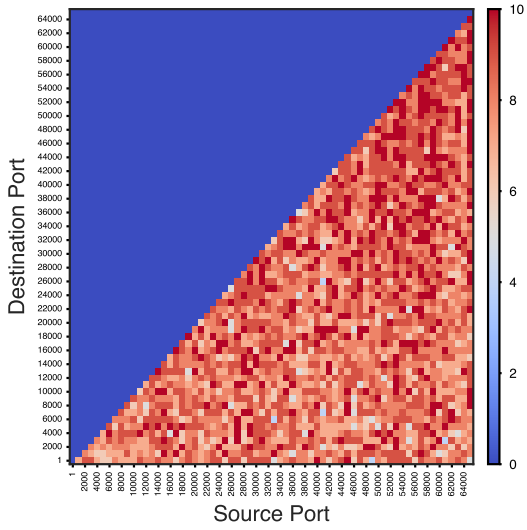

Figure 11: The censor does not track or block UDP flows if the source port of the QUIC Initial packet is less than or equal to its destination port. This rule applies to all port numbers, ranging from 1 to 65535.

C Vulnerability Disclosure Email to the Censors

As introduced in Section 9, we decided to disclose the availability attack (Section 6) to the censor, as this attack may exploit the GFW to cause additional harm to users. On January 22, 2025, we sent out the following email to CNCERT/CC and Fang Binxing who has been widely known as "the father of the GFW" [34]. We recommended removing the vulnerable QUIC censorship device and deploying egress filtering to prevent IP spoofing attacks. We wrote this email in both English and Chinese, and provided links to two private webpages, one in English and one in Chinese, with details of the attack. Although we did not receive any response or formal acknowledgment, we observed a total of 37 visits to the private English webpage (and zero visits to the Chinese one) between 2:04 PM (UTC+8) on Friday, January 24 and 9:35 AM (UTC+8) on Monday, February 24, 2025.

SUBJECT: Disclose a Vulnerability in the GFW's QUIC Filtering Mechanism
FROM: gfw.report <gfw.report@protonmail.com>
TO: CNCERT/CC <cncert@cert.org.cn>
CC: Fang Binxing <fangbx@iie.ac.cn>
DATE: Thu, 23 Jan 2025 12:01:46 +0000

Dear CNCERT Team,

We are writing to disclose a vulnerability introduced by the QUIC filtering mechanism deployed on the backbone network in China, active since at least April 7, 2024. This vulnerability allows a network attacker capable of spoofing IP packets, to use the Great Firewall of China (GFW) to disrupt or block communication between hosts inside and outside of China for an extended period.

Below we introduce the details, impact, and mitigation of this vulnerability. We also maintained an up-to-date version at:
[The URL to the English responsible disclosure page redacted.]

Vulnerability Details

An attacker can send a QUIC Initial packet (see below example) with an SNI on the firewall's blocklist (e.g., google.com) to a specific IP:port pair, triggering the GFW's residual censorship for approximately 180 seconds. If the attacker spoofs the source IP address to that of a victim inside China, this mechanism can be exploited to block the victim's IP address from connecting to the specified server IP:port for three minutes. Similarly, one can spoof the source IP address to be a victim server outside China, and send to a range of ports of a victim IP address in China. By repeatedly sending spoofed QUIC Initial packets, the attacker can sustain the block indefinitely.

When the firewall's censorship is triggered, it blocks based on three-tuple (source IP, destination IP, destination UDP port) for 3 minutes (180 seconds). Censorship can be triggered with a single UDP packet (see example below) containing a QUIC Initial packet with an SNI on the firewall's blocklist (e.g., google.com). Normally, this will only block between a client attempting to connect and the server. However, because the blocking can be triggered from a single UDP packet, a network attacker that can spoof IP packets can easily trigger the firewall into blocking other hosts.

For example, suppose there is a host in China at 19.89.5.35, and a DNS server outside China at 4.2.2.1 on UDP port 53. If an attacker sends a UDP packet (such as provided below) from 19.89.5.35:x (for any source port x) to 4.2.2.1:53, this will trigger the firewall to block 19.89.5.35 from sending any packets to 4.2.2.1:53 for 3 minutes. The attacker can continue to spoof packets from different source ports to extend the block indefinitely.

Impact

The development and deployment of the GFW, along with this identified issue, poses a severe risk to users in China and has the potential to disrupt communications on a large scale. For instance, it could be exploited to block significant portions of UDP-based DNS traffic between DNS resolvers in China and external networks, causing widespread connectivity issues.

To demonstrate the impact this attack could have, we conducted an experiment using 32 Amazon EC2 instances globally distributed. We ran the attack for 30 minutes, sending a DNS request from each EC2 instance to a VPS we control in Guangzhou. At the same time, a non-egress filtering box in the US spoofed packets from each EC2 IP to the Guangzhou box with a QUIC Initial packet containing an SNI from the firewall's blocklist. The map below shows which boxes were affected with just a single spoofing vantage point. Points in green experienced no connectivity issues, while points in red struggled to successfully send requests to the Guangzhou host. The black point in Guangzhou shows the location of our testing victim server and the black point in the US shows the location of our spoofing server.

Mitigation

Due to the potential harm from this attack, we urge taking immediate action to address this issue. As UDP is a connectionless protocol, it is difficult to prevent spoofing attacks. Therefore, the most complete mitigation against this attack is to disable the censorship middlebox responsible for blocking UDP connections. In addition to enabling these harmful attacks, the GFW also violates human rights by preventing access to information.

A less complete mitigation is to deploy egress filtering to prevent IP packet spoofing, but as long as an attacker can find one location where they can spoof packets, even outside of China, this attack will still be feasible. Given this, we recommend 1) immediately and permanently disabling the QUIC censorship national firewall and 2) deploy protections such as egress filtering to edge networks to limit IP spoofing.

Thank you for your attention to this critical matter. We remain available to provide additional technical details or answer follow up questions to ensure this issue is addressed promptly.

Sincerely,
Team

[The Chinese translation of the email above redacted.]
[The URL to the Chinese responsible disclosure page redacted.]

Example command:

```
nc -s $SRC_IP -p $SRC_PORT -vnu $DST_IP $DST_PORT <<<$(xxd -r -p \
<<< "c600000001104ebdf7c473c1c15db3ffa4534f5b3158102154b19e765d7a3caa33a20b92c56da30040e182
dcfd47c61c7fff552b8c61053c0c91ab148d199277a3b459519768aa6c79533eecd2d2e678dbac45dadef121d1d
3f5f56454c6b9305c45d919053fea8c1c1bd950d1fd14ee770d8312d10c03a18aea463538d721af70b4e732037e
ac620f361d0435114eea55204caa685dd33f8b2cb1dac6568b320e2d348f77e72a4c150ed5ac27a9ce9edf696ea
929baf34f28598320b0baa993fbdeddf7c45b724eee8f6fa9c7860a973f0138777422347161743bc6d36e519951
47d7f6d2cf4a398b7ea1066f77bcdee89e760d2568bc3c9bb8f7d5c43482a11a7d696c7dc62fe6ecade80000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000")
```