Even Censors Have a Backup: Examining China's Double HTTPS Censorship Middleboxes

Kevin Bock Gabriel Naval Kyle Reese Dave Levin
University of Maryland

ABSTRACT

The Great Firewall of China (GFW) has long censored HTTPS (via the Server Name Indication field, or SNI). Its mechanism for doing so has been studied, with various evasion strategies discovered in recent years. In this paper, we have evidence that suggests the GFW has deployed a second HTTPS censorship middlebox that runs in parallel to the first. We present a detailed analysis of this secondary censorship middlebox—how it operates, the content it blocks, and how it interacts with the primary middlebox—and present evidence that this has been in operation since at least September 2019. We also present several packet-based evasion strategies for the secondary middlebox and demonstrate that the primary censorship middlebox can be defeated independently from the secondary. Our code is publicly available.

CCS CONCEPTS

• **Social and professional topics → Censorship**.

KEYWORDS

Censorship; Geneva; Censorship-in-Depth

ACM Reference Format:
Kevin Bock, Gabriel Naval, Kyle Reese, Dave Levin. 2021. Even Censors Have a Backup: Examining China's Double HTTPS Censorship Middleboxes. In *ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet (FOCI'21), August 27, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. `https://doi.org/10.1145/3473604.3474559`

1 INTRODUCTION

As much of the web transitions to HTTPS, nation-state network censors have less information to base their decisions of whether or not to block or tear down a connection. Whereas HTTP permitted deep packet inspection (DPI) of keywords, HTTPS hides all request and response data through encryption. However, the server name indication (SNI) field in the TLS handshake reveals the website to which the client wishes to connect. Censors such as China and Iran have thus used the plaintext SNI field to guide their censorship decisions and, in some cases, outright block all traffic that seeks to hide the SNI through encryption (ESNI) [9].

As a result, significant effort has been paid to understanding and evading SNI censorship, with particular attention paid to one of the world's largest censors, the so-called Great Firewall of China (GFW). In 2019, Chai et al. [10] empirically evaluated how SNI censorship operated in China, and argued for the importance of using ESNI. Unfortunately, China began blocking all ESNI traffic the next year [9]. In 2020, Bock et al. investigated how to evade China's SNI censorship [7] and recently demonstrated how to weaponize it to launch availability attacks [5]. Through all of this work, a mental model emerged that indicated that China uses a single model of middlebox to detect and react to SNI connections.

In this paper, we reinvestigate these findings and discover that in fact China's GFW uses *two* distinct censorship mechanisms in parallel to censor HTTPS based on SNI.[1] We first discovered this second HTTPS censorship middlebox while trying to reproduce the 2019 censorship evasion results of Bock et al. [8] for HTTPS. We observed that some censorship evasion strategies could evade the GFW's known HTTPS censorship, but small modifications could cause strategies to fail unexpectedly: via a single `RST` packet deeper in the TLS handshake. Now, we understand and report on the root cause of this strange behavior: the GFW had a second censorship middlebox all along.

We present a detailed analysis of China's secondary HTTPS censorship middlebox: how it works, how it can be triggered, and how it can be defeated. We confirm this behavior is caused by a separate middlebox by identifying unique TCP-layer bugs in each middlebox, suggesting separate TCP stacks [7]. These findings are important in refining our understanding of SNI censorship in China—they resolve some of the confusing behavior previously identified and chart a clearer path forward for how to measure and evade SNI censorship more precisely. This is especially important now, as China has effectively stopped the roll-out of ESNI within its borders [9] and Russia is actively working to do the same [11].

Also, in a broader sense, our work demonstrates that censors are now deploying multiple, complementary mechanisms that operate over the same fields of the same packets. Although it is tempting to think of them as a single "black box" of censorship, this paper shows that it is both possible and important to tease them apart into their constituent components.

The rest of this paper is organized as follows. In §2, we review related work and background on nation-state censors and packet manipulation based censorship evasion. §3 discusses our methodology for our experiments. §4 shows how we can evade the newly discovered censorship middlebox and how censorship evasion is critical for our measurements of the new middlebox. §5 studies the functionality of the new middlebox. Finally, §6 discusses ethical considerations and §7 concludes.

[1]We only focus on SNI-based censorship of HTTPS, and thus use "HTTPS censorship" and "SNI censorship" interchangeably.

2 RELATED WORK

Censorship-in-Depth Our work studies two complementary censorship mechanisms that operate in tandem against the same protocol—what Bock et al. call "censorship-in-depth" [6]. One example of this is Iran's use of a protocol filter [6] to limit communication to a restricted set of protocols, in conjunction with standard DPI censorship to block certain keywords. Similarly, previous studies have shown that China uses a combination of IP blocking, DNS hijacking, and SNI filtering to block connections to HTTPS websites [3, 10].

Whereas these prior approaches investigate cooperating mechanisms that aim to censor *different* but complementary protocols, we have identified two distinct mechanisms that both aim to censor the *same exact* protocol (SNI-based HTTPS). As we will demonstrate, this makes it particularly challenging to disentangle the two, as they both operate on the same packets. Our findings demonstrate what we believe to be a novel way in which nation-states employ censorship-in-depth.

Measuring Censorship There have been many studies into how censors operate, largely broken down into two broad categories: First is the large set of papers that study and measure the *content or destinations* targeted by censorship [10, 18–22]. We do not expand on this body of knowledge, other than to test whether both of China's mechanisms of SNI-based censorship differ in the content they block (they do not appear to).

Second, and most related, is the body of work that studies *how* censors operate [2, 4, 13, 14, 17, 25, 27]. Our work refines prior understanding of China's SNI-based censorship, but in a broader sense has a much more significant takeaway message: censorship mechanisms may not be monolithic but rather may consist of multiple separate middleboxes working in tandem towards the same goal. We believe our techniques can be more broadly applied to studying other forms of such "backup" censorship.

Packet Manipulation-based Censorship Evasion In addition to measuring China's two forms of SNI-based censorship, we present ways to circumvent them both. To do so, we apply recent techniques that manipulate the packet stream from *one side* of the connection (either the client or the server) to confuse or prevent a censor from properly disrupting a connection. The canonical example of this attack is a TTL-limited `RST`: a client wishing to evade censorship, before sending a forbidden request, injects a `RST` packet with a `TTL` set high enough to reach the censor but not high enough to reach the server. When the censor processes this `RST` packet, it believes the connection has been torn down, throws away its Transmission Control Block (TCB), and stops tracking the connection. Since the `RST` packet will get dropped before it reaches the server, the server and the real TCP connection will not be affected and, for the rest of this flow, the client and server can communicate free of censorship.

Recent advances in censorship evasion have led to approaches that *automate* the discovery of packet manipulation censorship evasion strategies [7, 8, 24]. In this paper, we use one of these open-source tools (Geneva [8]) to discover four distinct evasion strategies. We show that China's primary HTTPS+SNI censorship middlebox can be defeated without defeating the secondary.

3 METHODOLOGY

Measuring two censorship mechanisms that both operate on the same packets is challenging. To understand how they both operate independently and in conjunction with one another, our methodology involves evading one of the boxes to selectively measure the other. In this section, we describe our high-level approach to evasion and measurement.

Admittedly, our approach was somewhat circular: our initial measurements provided insight that allowed us to begin evading, which let us perform more measurements, and so on. Thus, to best understand our methodology, it is useful to also understand at a high level how the two censorship mechanisms work, which we also provide here.

Vantage Points We obtained two censored vantage points inside China (Beijing) and external uncensored vantage points in Japan (Tokyo) and the United States (Iowa, Virginia). Our Chinese vantage points are located within different ISPs, but Xu et al. found that the GFW's actual deployment of certain censoring middleboxes may vary based on the type of ISP [26], so our conclusions are limited by the ISPs we can measure. We use the vantage points in China as our "client," and our vantage points outside as our "server." Throughout our experiments, we only connect to machines we control.

Detecting Evasion of One Mechanism It is straightforward to determine if we have evaded both of the censorship mechanisms—we need only see if we received the censored content. But how can we determine if we have evaded censorship of only *one* box?

The key insight is that the two mechanisms block censorship in different ways. The GFW's primary (already known) censorship middlebox operates by injecting an idiosyncratic pattern of three `RST+ACK` packets to both the client and the server once it observes a TLS Client Hello with a forbidden Server Name Indication (SNI) field [5, 7, 10]. We will refer to this primary middlebox as `MB-RA` (MiddleBox `RST+ACK`). The GFW's secondary SNI censorship middlebox, by contrast, tears down connections by injecting one single `RST` packet: we will refer to this middlebox as `MB-R` (MiddleBox `RST`).

Unless otherwise specified, we configured our vantage points to drop all outbound `RST` and `RST+ACK` packets. Thus, we expect any `RST` or `RST+ACK` packets received by our client to come from the `MB-R` or `MB-RA` middleboxes, respectively.

Triggering Censorship We trigger censorship by injecting forbidden domain names in the SNI field (though all communication is strictly between the client and server machines we control). However, we have found that it is not always sufficient to stop sending packets at that time.

Unlike `MB-RA`, `MB-R` does not tear down a connection immediately after observing a forbidden SNI. Instead, it waits to inject its `RST` packet until the client sends the next packet in the TLS handshake: the `ClientKeyExchange` or the `ClientChangeCipherSpec`. Note that the forbidden SNI field is not present in either of these messages. `MB-R` is a stateful middlebox that is *triggered* by the forbidden SNI field in the Client Hello message but does not *act* until after the client continues the handshake. We believe this is the reason researchers have not reported on this middlebox until now. Figure 1 illustrates the TCP 3-way handshake and TLS handshake and where each of the two middleboxes acts.

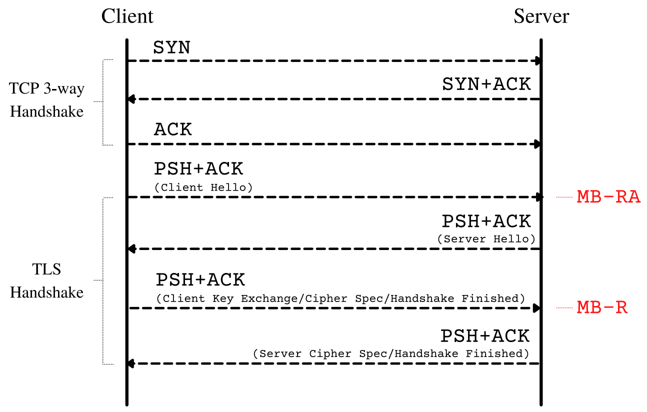

Figure 1: A waterfall diagram of the TCP 3-way handshake and the TLS handshake, denoting where the already known `MB-RA` and newly discovered `MB-R` middleboxes act during the connection. Note that `MB-R` does not act until deeper in the handshake than `MB-RA` (and *only* if `MB-RA` does not act), seemingly acting as a backup middlebox for China's HTTPS (SNI) censorship.

Isolating the Second Middlebox Studying `MB-R` is also made more difficult because `MB-RA` and `MB-R` seem to interact with one another. Specifically, when `MB-RA` takes action to tear down a connection, `MB-R` *does not act* even if `MB-RA` fails to tear down the connection or the connection continues. We performed an experiment in which we instrumented a vantage point within China and a server outside of China to drop all inbound `RST+ACK` packets and tried to complete a TLS handshake with a forbidden SNI between them. If `MB-R` and `MB-RA` operated independently, after both sides of the connection drop the `RST+ACK`s injected by `MB-RA`, we would expect `MB-R` to inject its `RST` packet once the client continues the TLS handshake.

Instead, we find that any time `MB-RA` injects packets, `MB-R` stops paying attention to the connection entirely. We believe the injected `RST+ACK` packets from `MB-RA` are causing `MB-R` to tear down its TCB (Transmission Control Block) for the connection. This experiment suggests that `MB-R` is a backup censorship middlebox for `MB-RA`: it only injects `RST` packets if `MB-RA` fails to take action.

This interaction between `MB-RA` and `MB-R` also offers `MB-R` a way to avoid state exhaustion: once a connection is torn down by `MB-RA`, `MB-R` does not need to continue tracking it. We believe this interaction also explains why other components of the GFW will stop paying attention to a connection if the client injects a `RST+ACK` packet (a TCB Teardown attack). Researchers have wondered why the GFW continues to be vulnerable to TCB Teardown attacks to this day, despite having been reported for years [7, 8, 15, 16, 23, 24]. If the GFW is architected to internally use the `RST+ACK` packets injected by one middlebox to prevent state exhaustion in other middleboxes, this would explain why TCB Teardown attacks have not been patched.

Unfortunately, this interaction between `MB-R` and `MB-RA` makes studying it in isolation difficult. The only signal we have to measure `MB-R` is its injection of `RST` packets, but it does not inject these packets until deeper in the TLS handshake than `MB-RA`. We could repeatedly make forbidden connections until `MB-RA` fails to inject packets, but to make reliable measurements, instead we leverage packet manipulation evasion strategies to *evade* `MB-RA` without affecting `MB-R`.

Evading Censorship We leveraged an open-source tool called Geneva (*Gen*etic *Eva*sion), a genetic algorithm designed to discover packet manipulation-based censorship evasion strategies. Geneva has been used successfully against the GFW in the past [7–9], as well as censorship infrastructure in other countries [6, 7].

The output of Geneva is sequences of packet manipulations that confuse or disable a censoring middlebox. Central to Geneva's ability to find evasion strategies is its *fitness function*, which evaluates how successful a strategy is against a given censor. For this work, we made a small modification to Geneva's reward function to optionally ignore inbound `RST` packets on both sides of the connection. This enables us to optionally train Geneva to find strategies that defeat only the `RST+ACK` middlebox (since `MB-RA` injects `RST+ACK` packets, not `RST` packets).

After using Geneva, we performed manual follow-up experiments to understand how each strategy works. To compute reliability for each strategy, we used each strategy 100 times while trying to complete a full TLS handshake with a censored keyword in the SNI field (`wikipedia.org`) between vantage points within China and outside of China.

Because of the interaction between the two middleboxes, we are only able to defeat either `MB-RA` alone or both of them together. Recall that the only signal we have to measure `MB-R`'s reaction is it injecting `RST` packets, but it does not do this injection until later in the TLS handshake after `MB-RA` may act. It is possible that there exist packet sequences that confuse or disable `MB-R` without disabling `MB-RA`, but we are unable to confirm this.

4 EVASION

In this section, we will report on client-side strategies we discovered with Geneva that defeat *only* `MB-RA` and both `MB-RA` and `MB-R`. Following precedent from prior work, we will report on the strategies we find both in text and include the Geneva syntax that implements the strategy.

4.1 `MB-RA` Evasion Strategies

The most reliable working client-side strategy that we found first sends two `SYN` packets, then splits the TLS Client Hello in half to make two TCP segments, and sends them out of order[2]. In our testing, this strategy worked with 99% reliability.

Strategy 1: `MB-RA`: Double-`SYN` Segmentation

```
[TCP:flags:S]-duplicate-|
[TCP:flags:PA]-fragment{tcp:-1:False}-|
```

The fact that this strategy works is strange and surprising. The GFW is known to be capable of reassembling TCP segments, even if sent out of order [7]. Indeed, if the second `SYN` packet is removed,

[2]Note that Geneva's syntax represents TCP segmentation with the `fragment` action with the `tcp` parameter.

the strategy no longer works, as MB-RA reassembles the TLS Client Hello and censors the connection. This strategy suggests that MB-RA is keeping track of both the TCP handshake and the TLS handshake, but seeing the unexpected SYN packet interferes with its ability to reassemble messages. We do not know why this is. Note that this strategy does not evade MB-R; this *only* disables MB-RA.

The second type of client-side strategy we discovered that defeats MB-RA also involves abusing MB-RA's ability to reassemble TCP segments. This strategy involves performing 6 TCP segmentations to create 7 total TCP segments out of the original TLS Client Hello, with each segmentation reversing the order of the segments. In the end, this strategy reverses the order of the segments exactly. This strategy worked with 100% reliability.

Strategy 2: MB-RA: Segmentation Overload

```
[TCP:flags:PA]-fragment{tcp:-1:False}(
    fragment{tcp:-1:False}(
        ,fragment{tcp:-1:False}),
        fragment{tcp:-1:False}(
            fragment{tcp:-1:False},
            fragment{tcp:-1:False})
)-| \/
```

This is not the only variant of this strategy that works to defeat MB-RA, but it is not sufficient to simply split the TLS Client Hello into any seven segments. Geneva found dozens of strategies with similar number and ordering of segmentation that function, and hundreds more that do not.

Without a second SYN packet, at least 7 segments are required for this strategy to work, and further segmentation does not negatively affect the reliability of the strategy. Exactly reversing the order of the segments too is not a requirement; other variants of this strategy exist that defeat MB-RA without defeating MB-R without this property. Previous researchers found that different parts of the GFW have issues reassembling segments less than 8 bytes long [7, 24], but each segment in this example is at least 24 bytes long.

We originally hypothesized that this series of segmentations must simply split up the SNI field across multiple packets, but when this strategy is used, the SNI field is intact and unchanged in a single TCP segment. Leaving the SNI field intact is also not a requirement; other versions of this strategy that split the SNI field across multiple segments and work equally well. Frankly, we do not understand why this strategy defeats MB-RA.

4.2 Evading MB-RA and MB-R

Next, we will discuss strategies that can defeat both MB-RA and MB-R. Geneva discovered variants of the aforementioned Segmentation Overload strategy that defeat both MB-RA and MB-R simultaneously, with 99% reliability. Like before, this strategy performs multiple rounds of TCP segmentations on the TLS Client Hello packet to produce 7 individual packets, most of which are out of order. Again, it is not clear why this strategy works.

The most salient difference between strategies that defeat MB-R compared to the previously discussed MB-RA-beating strategies is that these strategies contain at least one middle pair of segments that remain in-order. The location of the SNI field does not impact the reliability of this strategy; it can be included in any segment or be split across multiple segments.

Strategy 3: MB-R & MB-RA: In-Order Segmentation Overload

```
[TCP:flags:PA]-fragment{tcp:-1:False}(
    fragment{tcp:-1:False}(
        ,fragment{tcp:-1:False})
    ,fragment{tcp:-1:False}(
        fragment{tcp:-1:True},
        fragment{tcp:-1:False})
)-| \/
```

Geneva also found that it could combine pieces of the In-Order Segmentation strategy to reduce strategy complexity. This next strategy works by duplicating the SYN packet and performing three TCP segmentations of the TLS Client Hello.

Strategy 4: MB-R & MB-RA: Double SYN, Triple Segmentation

```
[TCP:flags:S]-duplicate-|
[TCP:flags:PA]-fragment{tcp:-1:False}(
    ,fragment{tcp:-1:False}(
        fragment{tcp:-1:True},)
)-| \/
```

In our follow-up experimentation, we find that in order for this strategy to defeat both MB-RA and MB-R, the segments must be sent in a specific order: the fourth segment must be sent first, then the second segment, then the third, and finally the first segment. Any deviation from this order causes MB-R to detect the sequence, though any order in which the first segment is not sent first is sufficient to evade MB-RA.

We verified that only *the order* in which the segments are sent matters, not the content or size of the segments. We manually tested different strategies that would make a single segment 188 bytes long (making each of the other segments just a single byte long); as long as the correct segment order is maintained, the strategy evades MB-RA and MB-R. We do not understand why these constraints apply.

We also rediscovered several strategies that researchers had found in the past for other components of the GFW [8, 9, 23]: TCB Teardowns (injecting a TTL-limited or checksum corrupted RST) and TCB Desynchronization (injecting a TTL-limited or corrupt checksum with data).

5 HOW DOES MB-R WORK?

Now that we have a robust way to trigger MB-R in isolation, we can explore how MB-R works. In this section, we report on MB-R's functionality.

Which packets from the client will MB-R act upon? We performed a series of experiments in which we instrumented a client to send a TLS Client Hello with a forbidden SNI field (such as wikipedia.org), followed by different client handshake messages or packet payloads, including empty packets, garbage messages, and HTTP payloads. We did not observe a response from MB-R for any non-TLS messages nor for `ClientHandshakeFinished` messages. We find that MB-R will only take action if it sees a `ClientKeyExchange` or `ClientChangeCipherSpec`.

Is MB-R bidirectional? Yes, both MB-R and MB-RA track connections that originate from both inside and outside of China. First, we confirmed that MB-RA is still bidirectional: we made requests from vantage points we controlled outside of China to our vantage points inside China, and in the opposite direction; in both cases, we can trigger MB-RA. Next, we tested if MB-R also monitors traffic inbound to China by sending multiple different packet sequences that evade MB-RA (in different ways) but trigger MB-R and confirmed that MB-R is also bidirectional.

What is the reliability of MB-R and MB-RA? Previous researchers have found that the GFW is not 100% reliable in its censorship (usually around 97%) [7, 8, 23]. To test the reliability of both the primary and secondary middleboxes, we sent 2,000 packet sequences with small sleeps in between for both MB-R and MB-RA from a vantage point outside of China to servers we controlled inside China, each from a fixed source port to a unique destination port. By observing which ports are interfered with, we can estimate the reliability of each middlebox.

We find that MB-R interfered with 87.0% of the connections, and MB-RA interfered with 88.2% of connections. Interestingly, these numbers composed together explain the approximately 97% total reliability found by previous researchers [7]: the likelihood of both middleboxes failing is approximately 1.8%, for a total reliability of 98%.

What ports does MB-R monitor? Researchers in the past have reported that the GFW's SNI censorship middlebox (MB-RA) monitors all ports 1-65,535 [7]. To test which ports MB-R monitors, we conducted an experiment in which we sent the sequences of packets that trigger MB-R from our vantage points outside of China to servers we control within China on every destination port. For this experiment, we configured the server within China to drop all outbound RST and RST+ACK packets, so we expect any RST or RST+ACK packet received by our vantage points outside of China to originate from the middlebox. We also verified the sequence numbers of inbound RST packets to prevent any spurious RST packets from interfering with the experiment. To account for MB-R not being 100% reliable, for any port that did not elicit censorship, we repeat the packet sequences to confirm whether or not the failure was a fluke. We find that MB-R, like the already known MB-RA, monitors all ports.

Does MB-R monitor ESNI or omit-SNI? In 2020, researchers discovered that China had deployed a new censorship middlebox to censor uses of HTTPS with Encrypted SNI (ESNI) [9]. They found that the new ESNI censorship middlebox does not censor *omit-SNI* (Client Hello messages with the SNI field omitted), although other censorship middleboxes have been observed censoring omit-SNI [1]. They determined that this censorship middlebox was different from the already known MB-RA HTTPS (SNI) censorship middlebox and confirmed that MB-RA does not monitor or censor uses of ESNI or omit-SNI. Does MB-R censor ESNI or omit-SNI?

To test this, we modified the sequence of packets we discovered that trigger MB-R. In the first experiment, we replaced the forbidden SNI TLS Client Hello with a TLS 1.3 Client Hello with an ESNI extension. In the second experiment, we replaced the forbidden SNI TLS Client Hello with a TLS Client Hello with no SNI extension at all. We find that MB-R does not censor ESNI or omit-SNI connections.

Does MB-R middlebox have residual censorship? *Residual censorship* is a feature of some censorship middleboxes in which after a censorship event occurs between a pair of hosts, the censor continues to interfere with benign connections between them for a short amount of time [5]. Some prior work has reported that MB-RA has residual censorship [10], but other researchers have reported that this residual censorship may be specific to certain vantage points [5]. From our vantage points in China, we do not observe residual censorship for MB-RA: after a censorship event, future benign connections between the same pair of hosts are not affected.

To test if MB-R has residual censorship, we issued packet sequences that trigger MB-R, and then sent follow-up benign connections. We find the same result as MB-RA: we do not observe residual censorship. Unfortunately, like all censorship measurement research, we are limited in what vantage points we can access, and absence of evidence for residual censorship at both of our vantage points is not evidence of its absence throughout the network. It is possible that MB-R's residual censorship varies by geographic location.

Does MB-R and MB-RA have the same blocklist? To test if MB-R and MB-RA have different blocklists, we downloaded CitizenLab's China (567 domains) and Global (1,435 domains) test lists [12] to see if there were any domains censored by one middlebox that was not censored by the other. For each domain on the test list, we sent trigger packet sequences for both MB-RA and MB-R from the vantage points we controlled in China to a vantage point outside of China with the test domain in the SNI field of the TLS Client Hello. We used a unique source port for each of these connections and our vantage points were configured to drop all outbound RST and RST+ACK packets, so we expect any RST or RST+ACK packets we receive to originate from the GFW. Note that since our vantage points do not experience residual censorship for MB-RA or MB-R, residual censorship is not a concern for this experiment. Since the reliability of MB-R and MB-RA are not 100%, we repeated this experiment 5 times. As long as a test domain triggers a middlebox at least once, we know it is censored.

We find that both middleboxes had the same response to all of the domains we tested; if MB-RA censored it, so did MB-R and vice versa. This experiment supports our theory that MB-R acts as a backup middlebox to MB-RA.

Where is MB-R deployed relative to MB-RA? To test where MB-R and MB-RA are located on the network, we performed an experiment in which we TTL limited the packet trigger sequences for both MB-R and MB-RA. By repeatedly sending a trigger sequence of packets with increasing TTL values, we can see at what hop each middlebox performs traffic injection. We repeated this experiment from both of our vantage points inside of China destined to multiple vantage points outside the country and then again in the reverse direction. We find that MB-RA and MB-R were the same number of hops away from each test vantage point; this suggests that they are collocated on the network level. This finding aligns with a previous exploration of China's censorship middlebox, which also found that China collocated the censorship infrastructure for other protocols [7].

6 ETHICAL CONSIDERATIONS

We designed our experiments to minimize impact on other hosts and to minimize risk to other users. All of our experiments with `MB-R` and training with Geneva was done strictly between hosts we controlled and hosts not located in residential networks. Geneva does not spoof IP addresses and generates a fairly small amount of traffic while training [8]. We also followed the original experiment design of Geneva and evaluated strategies serially to limit the volume of data we sent at once.

7 CONCLUSION

In this paper, we show that China's SNI-based censorship has continued to evolve. We discover and report on the existence of a secondary SNI censorship middlebox and show that the middleboxes can be studied in isolation.

It is somewhat surprising that China continues to invest in its SNI-based censorship, as TLS is evolving to incorporate encrypted versions with Encrypted SNI (ESNI) and Encrypted Client Hello (ECH). Indeed, China continues to do so, and they (as with other countries [11]) are working to block ESNI outright [9]. This indicates that there is not yet enough critical mass behind ESNI/ECH to make the collateral damage of blocking them prohibitively large for China. Until it is, SNI-based censorship will remain a threat.

Our work also uncovers a more fundamental finding: censors are employing censorship-in-depth not just by blocking multiple intersecting protocols but by deploying middleboxes that target the same protocol in slightly different ways. The techniques we presented in this paper provide a potential path forward for understanding and evading these robust forms of censorship. To assist in these efforts, we have made our code publicly available at `https://geneva.cs.umd.edu`

ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd, Rob Jansen, for their helpful feedback. This research was supported in part by the Open Technology Fund and NSF grants CNS-1816802 and CNS-1943240.

REFERENCES

[1] 2019. Russia Censoring Omitted SNI. `https://github.com/net4people/bbs/issues/10`. (2019).
[2] Anonymous. 2012. The Collateral Damage of Internet Censorship. *ACM SIGCOMM Computer Communication Review (CCR)* 42, 3 (2012), 21–27.
[3] Anonymous, Arian Akhavan Niaki, Nguyen Phong Hoang, Phillipa Gill, and Amir Houmansadr. 2020. Triplet Censors: Demystifying Great Firewall's DNS Censorship Behavior. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. USENIX Association. `https://www.usenix.org/conference/foci20/presentation/anonymous`
[4] Simurgh Aryan, Homa Aryan, and J. Alex Halderman. 2013. Internet Censorship in Iran: A First Look. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*.
[5] Kevin Bock, Pranav Bharadwaj, Jasraj Singh, and Dave Levin. 2021. Your Censor is My Censor: Weaponizing Censorship Infrastructure for Availability Attacks. In *USENIX Workshop on Offensive Technologies (WOOT)*.
[6] Kevin Bock, Yair Fax, Kyle Reese, Jasraj Singh, and Dave Levin. 2020. Detecting and Evading Censorship-in-Depth: A Case Study of Iran's Protocol Whitelister. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*.
[7] Kevin Bock, George Hughey, Louis-Henri Merino, Tania Arya, Daniel Liscinsky, Regina Pogosian, and Dave Levin. 2020. Come as You Are: Helping Unmodified Clients Bypass Censorship with Server-side Evasion. In *ACM SIGCOMM*.
[8] Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. 2019. Geneva: Evolving Censorship Evasion. In *ACM Conference on Computer and Communications Security (CCS)*.
[9] Kevin Bock, iyouport, Anonymous, Louis-Henri Merino, David Fifield, Amir Houmansadr, and Dave Levin. 2020. Exposing and Circumventing China's Censorship of ESNI.
[10] Zimo Chai, Amirhossein Ghafari, and Amir Houmansadr. 2019. On the Importance of Encrypted-SNI (ESNI) to Censorship Circumvention. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*.
[11] Catalin Cimpanu. 2020. Russia wants to ban the use of secure protocols such as TLS 1.3, DoH, DoT, ESNI. `https://www.zdnet.com/article/russia-wants-to-ban-the-use-of-secure-protocols-such-as-tls-1-3-doh-dot-esni/`. (2020).
[12] CitizenLab. [n. d.]. CitizenLab Censorship Test Lists. `https://github.com/citizenlab/test-lists`. ([n. d.]).
[13] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. 2015. Examining How the Great Firewall Discovers Hidden Circumvention Servers. In *ACM Internet Measurement Conference (IMC)*.
[14] Jill Jermyn and Nicholas Weaver. 2017. Autosonda: Discovering Rules and Triggers of Censorship Devices. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*.
[15] Sheharbano Khattak, Mobin Javed, Philip D. Anderson, and Vern Paxson. 2013. Towards Illuminating a Censorship Monitor's Model to Facilitate Evasion. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*.
[16] Fangfan Li, Abbas Razaghpanah, Arash Molavi Kakhki, Arian Akhavan Niaki, David Choffnes, Phillipa Gill, and Alan Mislove. 2017. lib.erate, (n): A library for exposing (traffic-classification) rules and avoiding them efficiently. In *ACM Internet Measurement Conference (IMC)*.
[17] Zubair Nabi. 2013. The Anatomy of Web Censorship in Pakistan. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*.
[18] Kei Yin Ng, Anna Feldman, and Chris Leberknight. 2018. Detecting Censorable Content on Sina Weibo: A Pilot Study. In *Hellenic Conference on Artificial Intelligence (SETN)*.
[19] Paul Pearce, Ben Jones, Frank Li, Roya Ensafi, Nick Feamster, Nick Weaver, and Vern Paxson. 2017. Global Measurement of DNS Manipulation. In *USENIX Security Symposium*.
[20] Ram Sundara Raman, Prerana Shenoy, Katharina Kohls, and Roya Ensafi. 2020. Censored Planet: An Internet-wide, Longitudinal Censorship Observatory. In *ACM Conference on Computer and Communications Security (CCS)*.
[21] Rachee Singh, Rishab Nithyanand, Sadia Afroz, Paul Pearce, Michael Carl Tschantz, Phillipa Gill, and Vern Paxson. 2017. Characterizing the Nature and Dynamics of Tor Exit Blocking. In *USENIX Security Symposium*.
[22] Benjamin VanderSloot, Allison McDonald, Will Scott, J. Alex Halderman, and Roya Ensafi. 2018. Quack: Scalable Remote Measurement of Application-Layer Censorship. In *USENIX Security Symposium*.
[23] Zhongjie Wang, Yue Cao, Zhiyun Qian, Chengyu Song, and Srikanth V. Krishnamurthy. 2017. Your State is Not Mine: A Closer Look at Evading Stateful Internet Censorship. In *ACM Internet Measurement Conference (IMC)*.
[24] Zhongjie Wang, Shitong Zhu, Yue Cao, Zhiyun Qian, Chengyu Song, Srikanth V. Krishnamurthy, Kevin S. Chan, and Tracy D. Braun. 2020. SymTCP: Eluding Stateful Deep Packet Inspection with Automated Discrepancy Discovery. In *Network and Distributed System Security Symposium (NDSS)*.
[25] Xueyang Xu, Morley Mao, and J. Alex Halderman. 2011. Internet Censorship in China: Where Does the Filtering Occur?. In *Passive and Active Network Measurement Workshop (PAM)*.
[26] Xueyang Xu, Z. Morley Mao, and J. Alex Halderman. 2011. Internet Censorship in China: Where Does the Filtering Occur?. In *Passive and Active Measurement*, Neil Spring and George F. Riley (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 133–142.
[27] Tarun Kumar Yadav, Akshat Sinha, Devashish Gosain, Piyush Kumar Sharma, and Sambuddho Chakravarty. 2018. Where The Light Gets In: Analyzing Web Censorship Mechanisms in India. In *ACM Internet Measurement Conference (IMC)*.

APPENDIX

In this appendix, we provide more information about Geneva itself and how Geneva works. For more information on Geneva, see Bock et al.'s original paper [8], or the documentation at `geneva.readthedocs.io`.

Geneva (*Gen*etic *Eva*sion) is a genetic algorithm we developed that evolves censorship evasion strategies against a censor. Unlike most anti-censorship systems, it does not require deployment at both ends of the connection: it runs exclusively at one side (client or server) and defeats censorship by manipulating the packet stream to confuse the censor without impacting the underlying connection. A "censorship evasion strategy" describes how that traffic should be modified. Since Geneva will be evolving these strategies, they are expressed in a domain-specific language that comprises the DNA of each strategy. (For a full rundown of the strategy DNA syntax, see the documentation).

Geneva is comprised of two main components. First, the genetic algorithm, which can evolve new ways to defeat a censorship system given an application that experiences censorship and a fitness function. Second, the strategy engine, which applies a given strategy on the fly to modify active network traffic.