Bleeding Wall: A Hematologic Examination on the Great Firewall

Sakamoto
Shinonome Lab
54k4m070@proton.me

Elson Wedwards
ElsonWedwards@proton.me

ABSTRACT

In-depth observations of the Great Firewall of China (GFW) are challenging because it is an on-path black box, especially with limited outbound packets that seldom reveal its internals. In this paper, we had a rare opportunity to exfiltrate parts of the GFW's memory from its packet injectors with malformed DNS requests by reviving a vintage vulnerability. Through analysis, we found it contained Internet traffic going across China's borders and stack frames of the packet-handling processes of the GFW. With this insight, we evaluated the encapsulated sensitive information and inferred characteristics of the GFW's processes. Moreover, we studied the feasibility of several attacks resulting from this vulnerability, including off-path attacks and reflective amplification attacks. We further discuss this novel attack surface and potential threats caused by such defective censors.

1 INTRODUCTION

China's Great Firewall is the largest censor on the Internet, inspecting and intercepting the network traffic of over a billion netizens. The GFW employs various techniques to impose its censorship policies, including IP blocking [12], DNS poisoning [3, 24], active probing [11], entropy-based traffic identification [32], etc. As a bidirectional censorship middlebox between China and the rest of the world, the GFW had several incidents impacting the global Internet in its history, such as poisoning the DNS servers outside China [4, 28] and forging SSL certificates for MITM attacks [23].

Despite its profound influence on the Internet, it remains difficult to directly observe the GFW and its inner workings. The GFW works in an on-path manner as a black box, meaning that it does not have public IP addresses and is thus not directly accessible; it only emits limited types of packets, predominantly simple TCP RST and forged DNS responses. Prior measurement works on the GFW mainly observed its expected behaviors or the side-channel information, e.g., the IP ID of the packets from the GFW [3].

However, we find that it is surprisingly feasible to perform more proactive measurements and even launch attacks leveraging GFW's certain implementation flaws. Specifically, we identified an out-of-bounds read vulnerability in the DNS packet injector of the GFW, which is a variant of a patched vulnerability revealed in 2010 [7]. Due to the lack of proper domain name validation, specially crafted DNS requests could cause the GFW to include the data beyond the network packet in its buffer in the forged responses. Such data usually contained the remains of the last handled packet. On rare occasions, it contained the stack frames of other functions.

Free and Open Communications on the Internet (1), 13–21

This vulnerability essentially enabled one to sample the international traffic flowing through China's backbone networks, which posed a great threat to the users' data security. We evaluated the sensitive information included in the leaked data. The leaked stack frames contained memory addresses, including the return addresses and saved frame pointers, which provided a peek into the GFW's processes. We inferred some characteristics of the GFW's programs.

This vulnerability could also enable off-path attacks and reflective amplification attacks. Malicious adversaries might induce desired traffic (e.g., DNS requests) and try to "read" it to perform sophisticated attacks (e.g., DNS cache poisoning). Furthermore, the leakage itself could make the GFW a distributed amplifier with an amplification factor of 4.04×, and when combined with routing loops, the factor could be over 400×.

We found that the GFW became aware of this vulnerability and started to patch it during our measurements. We recorded a part of this process, which implied the GFW was maintained and updated city by city, except that the GFW in Shanghai was updated in two steps. As of now, the GFW has fixed this vulnerability completely.

The existence of this vulnerability implies the possibility of exploiting the censors' defective implementations and suggests a new attack surface. As we present in this work, it not only provides the researchers with chances of further in-depth studies but also shows how flawed censors may threaten the Internet.

2 BACKGROUND

Architecture of the GFW. The GFW is an on-path system located near the international Internet exchange points (IXPs) in three cities of China: Beijing, Shanghai, and Guangzhou [20, 21]. Traffic is copied with optical splitters and sent to the censorship devices in order to prevent the GFW from becoming a bottleneck. When the traffic matches specific censorship policies, the GFW intervenes in the communication by injecting packets like forged TCP RST and DNS responses, blocking ports, and/or disseminating false routing information to block the IP addresses at the BGP level [3, 12].

DNS poisoning of the GFW. When the GFW detects a DNS request on port 53 [13], it extracts the queried domain name and matches it against the blocklist. If there is a match, the GFW injects a forged response pointing to a wrong IP address to prevent the user from accessing the target [3]. There are three packet injectors responsible for DNS poisoning in the GFW, each with different characteristics and blocklists. They also use different sets of IP addresses in the forged responses. It is worth noting that this blocking method only applies to DNS over UDP. The GFW injects RST packets for TCP-based queries.

Previous works on the GFW. As the GFW evolves, many relevant studies have appeared in recent years. An anonymous paper [3] in 2014 analyzed the GFW's DNS poisoning rules and studied the topology. It further inferred the size and traffic volume of one injector through the patterns of the IP ID field of its outbound packets. In 2017, Wang *et al.* [31] analyzed the GFW's state machine. In 2020, Alice *et al.* [11] researched the GFW's active probing techniques; Anonymous *et al.* [13] fingerprinted three different DNS packet injectors of the GFW. In 2021, Hoang *et al.* [24] established a long-term monitoring platform for the GFW's domain filtering policy. In 2023, Wu *et al.* [32] characterized the GFW censoring fully encrypted traffic with entropy tests.

gfw-looking-glass.sh. In 2010, an organization named *gfwrev* published a one-line script to dump parts of the memory of the GFW, exploiting the GFW's flaw in parsing DNS compression pointers [7]. One could specify the offset in the compression pointer beyond the packet, and the GFW would copy the memory therein without validation. This vulnerability was fixed long ago, and the corresponding packet injector stopped handling compression pointers in 2014. We see no detailed measurements exploiting the old vulnerability on the Internet. However, this vulnerability inspired us to find the one we used in this work.

3 VULNERABILITY

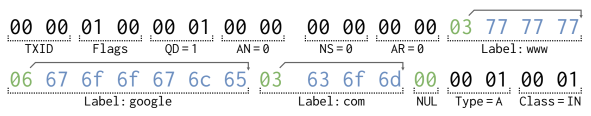


Figure 1: A Legitimate DNS Request for "www.google.com"

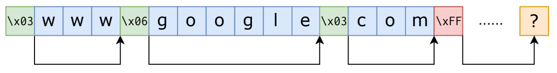
Figure 1 shows a legitimate DNS request, where the domain name is represented as a series of labels. Each label is prepended with a byte (length field) denoting its length in bytes, and a single `\x00` indicates the end of the domain name. Since a server copies the query part to the response, if it fails to validate the lengths of labels, it may be vulnerable to out-of-bounds reads. This has historically happened in closed-source implementations of DNS parsing logic, e.g., in CVE-2020-27737 [6].

We found one of the GFW's packet injectors[1] susceptible to a similar vulnerability due to a lack of proper checks on the DNS requests. In fact, this injector violated multiple principles stated in RFC 1035 [1] and RFC 9267 [18]:

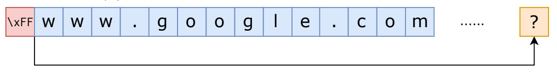
- No strict format validation: The injector did not strictly check the format of DNS requests, possibly for faster processing. It responded to any query containing a blocked domain regardless of the values of its type and class fields or the absence of these fields. In other words, even apparently malformed DNS packets would still trigger this censor. The injector copied the two fields to the response as is, and when they did not exist in the request, it copied 4 bytes beyond the packet.
- No length or null-terminator validation: The injector did not check whether the length of the domain and its labels conformed to the specification (i.e., no labels longer than 63 bytes and no domain names longer than 255 bytes) or whether the domain ended with a `\x00`. It did not check whether the label length exceeded the request packet, either. In this case, the response would contain the memory content beyond the packet.
- No character validation: DNS only allows letters, digits, and hyphens in domain labels, but the injector did not have any restriction on the characters in a label. As a result, any data could be included in the question section of the response.

[1]Injector 3 in [13] with TTL mirroring and zero IP ID.

3.1 Payloads and Delivery



(a) Format 1: Overflow the Last Label



(b) Format 2: Overflow the First Label

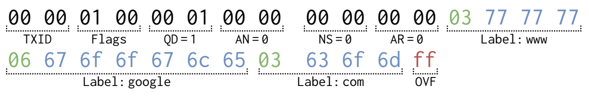
Figure 2: Two Formats of Payloads

We identified two formats of payloads that could trigger the vulnerability and read out-of-bounds data. Furthermore, we devised two methods to deliver the payloads, which had implications for the responses in terms of the leak size and location.

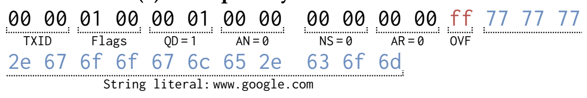
The payload is the same as a legitimate DNS request before the domain name, where we use malformed labels without the type or class field. In this section, we use the domain `www.google.com` as an example.

Format 1: Overflow the last label. As shown in Figure 2a, we insert the byte `\xFF` after the regular labels, indicating a following label of 255 bytes. Since this is the end of the packet, the injector would continue copying out-of-bounds data to the response. In this format, the payload looks similar to that of the old vulnerability, though out of different mechanisms.

Format 2: Overflow the first label. As shown in Figure 2b, we change the first length field to `\xFF` and replace the other length fields with periods (`\x2E`). Since the GFW converted the domains to strings and matched them literally [19], it could also trigger the censor and exploit the vulnerability. This payload differs from the previous work, and we could leak one more byte with the same domain. Besides, this format had a higher success rate and more triggering domains (details in §3.2).



(a) Example Payload in Format 1


00 00 01 00 00 01 00 00 00 00 00 00 ff 77 77 77
TXID
Flags
QD = 1
AN = 0
NS = 0
AR = 0
OVF
2e 67 6f 6f 67 6c 65 2e 63 6f 6d
String literal: www.google.com

(b) Example Payload in Format 2

Figure 3: Two Formats of the Payload

The two formats of payloads are illustrated in Figure 3a and Figure 3b. The TXID could be any value and would be mirrored in the response.

When the vulnerability was triggered, the packet injector sent forged responses, which included leaked memory after the domain labels we used in the payload. The flavors of the leaked data varied depending on the delivery method, implying different memory locations.

Method 1: Regular UDP datagrams. The vulnerable packet injector enforced bidirectional filtering so the payload could be sent across the GFW in regular UDP datagrams from either side. In this case, the leaked data contained mostly network traffic flowing through the GFW, as shown in Figure 4a. We redacted any potentially sensitive information in the figure.

Method 2: Fragmented UDP datagrams. The GFW is capable of assembling IP fragments [31], so we can split the payload packet in the middle into two fragments before sending them across the GFW. In this case, the leaked data usually included a domain (as a literal string, not DNS labels) amid zeros and unknown data, as shown in Figure 4b. We redacted part of the domain name in the figure.

```
      TXID  Flags  QD=1  AN=1  NS=0  AR=0
0000  00 00 81 80 00 01 00 01 00 00 00 00 03 77 77 77   ·············www
0010  06 67 6f 6f 67 6c 65 03 63 6f 6d ff 2f 66 61 76   ·google·com·/fav
0020  69 63 6f 6e 2e 69 63 6f 20 48 54 54 50 2f 31 2e   icon.ico HTTP/1.
0030  31 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 [redacted]   1··User-Agent:[redacted]
0040  [redacted] 0d 0a   [redacted]··
0050  48 6f 73 74 3a 20 [redacted]   Host:[redacted]
0060  [redacted]
0070  [redacted]
0080  [redacted] c0 0c   [redacted]··
                                                    PTR
0090  00 01 00 01 00 00 00 4d 00 04 9d f0 14 08         ·······M······
      Type=A CLS=IN TTL  Length  Address
```

(a) Regular UDP Datagrams with Payload in Format 1

```
      TXID  Flags  QD=1  AN=1  NS=0  AR=0
0000  00 00 81 80 00 01 00 01 00 00 00 00 03 77 77 77   ·············www
0010  06 67 6f 6f 67 6c 65 03 63 6f 6d ff 00 00 00 00   .google.com·····
0020  00 00 00 00 00 00 00 00 00 00 00 00 35 00 00 00   ·············5···
0030  00 00 00 00 00 00 00 00 00 00 00 00 20 15 ee 39   ················9
0040  fd 7f 00 00 25 00 00 00 fd 7f 00 00 b0 5c bf ec   ····%·······\··
0050  fd 7f 00 00 33 31 0d 0a cd cd cd cd 15 01 00 00   ····31··········
0060  00 00 00 00 2e [redacted]   ····.[redacted]
0070  [redacted] 2e 63 6e 00 00 00 00 00 00 00 00   [redacted].cn········
0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 c0 0c   ················
                                                    PTR
0090  00 01 00 01 00 00 00 c2 00 04 a8 8f ab bd         ·······M······
      Type=A CLS=IN TTL  Length  Address
```

(b) Fragmented UDP Datagrams with Payload in Format 1

Figure 4: Typical Responses for the Two Delivery Methods

The vulnerable packet injector limited the size of responses to a maximum of 158 bytes regardless of the requests. It seems like an independent limit on the outgoing packets of this injector, as RFC 1035 restricts the DNS packets over UDP to 512 bytes.

This vulnerability could not be triggered with TCP-based queries as the GFW injects simple RST packets for such queries instead of forging responses.

3.2 Trigger Domains

Not all blocked domains could trigger this vulnerability. When the exploit failed, we received no forged response. To determine the effective domains and guide our following measurements, we experimented with over 390K blocked base domains from *GFWatch* [24] in October 2023. We encapsulated each domain in both payload formats, sent the payloads to 16 different IP addresses across the GFW, and recorded the number of successful exploits.

Interestingly, we found that 3,976 domains in format 1 triggered the vulnerability at least once, while the number for format 2 was 137,722. However, only 43,545 domains in format 2 triggered it twice, 10,955 made it three times, and 4,422 made it four times. In contrast, the figures for format 1 were relatively stable.

Considering this trait, we deemed domains with at least nine successes effective, and the results are illustrated in Figure 5.

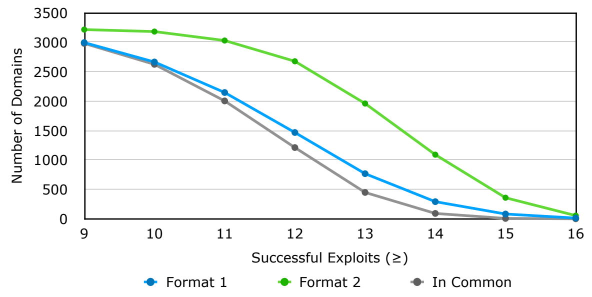


Figure 5: Number of Domains and Successful Exploits

We got several facts from the results:

- A total of 3,233 domains could effectively trigger this vulnerability in at least one format.
- The effective domains were not strongly related to how they were blocked. Only 489 of the 3,233 domains were blocked with optional trailing characters[2].
- The payload had a higher overall success rate at 86.38% in format 2 than in format 1 at 76.98%.
- The shortest effective domains were `{4,5,6,7,8,9}.tt` and `x.co`, and they only worked in format 2. On top of that, the domains in the responses were always appended with a byte `\x00` for unknown reasons.

We were unable to determine what caused these behaviors without access to the binary or source code of the GFW. However, based on the results, we adopted the 4-letter domains and the second format in later measurements as they could help us dump more bytes with fewer exploits.

3.3 Limitations

There are several limitations in exploiting this vulnerability:

- Small leak size: The vulnerable packet injector limited the responses to 158 bytes, where the query had at most 130 bytes. This restricted the leak size to a maximum of 124 bytes each time when using the second payload format and the shortest known trigger domain.
- Fixed memory location: We can only extract memory adjacent to the domain labels at two fixed locations. This prevented us from obtaining more interesting data at other memory locations.
- Random GFW instance: The GFW adopts load balancing based on the source and destination IP addresses and dispatches packets to different groups of servers [3]. As a result, we could not continuously dump the memory from the same instance and perform the long-term observation.

[2]Rule 1, 2, 5, 6, 7 and 8 in [24]

Table 1: Observation Sites

Site	Location	Landing ISP	Landing City
1	Singapore	China Unicom	Shanghai
2	South Korea	China Telecom	Shanghai
		China Mobile	Shanghai
3	Japan	China Unicom	Guangzhou
4	Japan	China Telecom	Shanghai
		China Unicom	Shanghai
5	USA	China Telecom	Guangzhou
6	USA	China Mobile	Guangzhou
7	USA	China Telecom	Shanghai
		China Unicom	Shanghai
8	Germany	China Mobile	Shanghai

4 MEASUREMENTS AND INFERENCES

We conducted measurements by collecting the leaked data to understand the vulnerability better and perform analysis. In this section, we present our methodology, results, and inferences.

We did our initial measurements in September 2023. However, when we started the formal measurements in October 2023, the GFW had patched its devices in Beijing. Therefore, the majority of the data came from the GFW nodes in Shanghai and Guangzhou. We collected over 1 TB of data over a period of three days, containing over 13B leaks. Approximately 87.43% of them have non-duplicate data per a sampling inspection of 52M leaks. The methodology is introduced in §4.1.

In §4.2, we verified the traffic leak, localized the memory location, and identified the stack frames in the leaks. In §4.3, we classified and evaluated the sensitive information included in the leaks. In §4.4, we analyzed the stack frames further and inferred some characteristics of the GFW's processes. In §4.5, we recorded the process of the GFW fixing this vulnerability.

4.1 Methodology

We set up eight observation sites in five countries to cover different ISPs in different international IXPs where the payloads entered China. The specific list is shown in Table 1, where the landing ISP and city denote through which route a packet entered China. We placed all of our observation sites outside China in order to avoid unnecessary risks.

During the measurements, each site continuously sent payloads to assigned subnets, extracted the leaked data from the responses forged by the GFW, and uploaded it to a central server. We tested the subnets on every site with `traceroute` so that the traffic flowed through the desired landing ISPs and cities.

Besides, we set two restrictions based on what we found in the initial measurements:

- The vulnerability existed in both IPv4 and IPv6 and the leaks seemed similar. However, it was hard to determine the geographic location of IPv6 addresses at the city level. For ease of analysis, we chose to measure using IPv4 only.
- We did not find much data of interest in the responses when delivering the payload in IP fragments (the second delivery method in §3.1), and the success rate was lower. To improve efficiency, we only used regular UDP datagrams to deliver the payload.

We conducted the measurements responsibly. We collected, stored and processed the data securely, and limited our rates and TTL to avoid overwhelming the networks or hosts. The detailed ethical considerations are presented in §7.

4.2 Characterizing the Leaks

In this section, we confirmed the data indeed contained the Internet traffic handled by the GFW, inferred the leaks came from the stack, and identified the stack frames included in the leaks.

4.2.1 Verifying the Traffic Leak. Most data leaked by the GFW looked like Internet traffic, but we still needed to prove it.

We set up a tool on one of the observation sites to continuously send probing packets of 256 bytes to a random port of a blackhole IP address at 1,000 PPS. This address is the first one in a prefix announced by an autonomous system, and we limited the TTL to a value that just allowed the forged responses to reach our observation site. We filled the probing packets with magic numbers and timestamps so that we could distinguish them from other network traffic and retrieve the departure time. We started another collection process to leak data from the same landing ISP and landing city at 5,000 PPS.

Over a period of 24 hours, we collected 86,647 leaks with our magic number. We examined the matched parts and confirmed that the GFW was leaking our probing packets across China's borders. Moreover, the leak times were very close to the departure times, within a round-trip time. This indicated the leaks were from a buffer holding all network packets passing through the GFW.

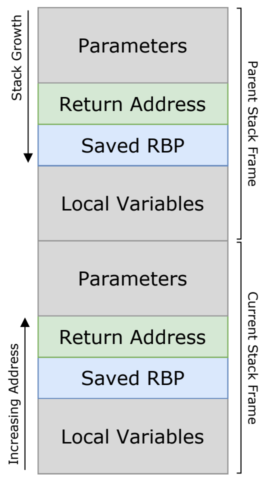
4.2.2 Localizing the Memory Location. We observed that all the matched leaks had the middle 124 bytes of our probing packets at the same position. Based on this observation, we presumed that the packets were held at a fixed memory location, likely an array buffer on the stack of the handler thread. The leading bytes were overwritten by our own payload, and the trailing bytes exceeded the maximum size of the DNS response, so we could not leak them.

To validate this hypothesis, we assumed the GFW was running Linux on the x86-64 platform. In this case, the threads would have stack frames of the layout in Figure 6a, where the return address starts with `0x00005` for functions in the executable, and the saved RBP starts with `0x00007f` for stack addresses. Although in our model, as shown in Figure 6b, we could not leak the saved RBP or return address of the current function, as the packet buffer was much larger[3] than what we could leak, there was a possibility that the leaked data contained a part of the stack frame of a function previously called by the parent.

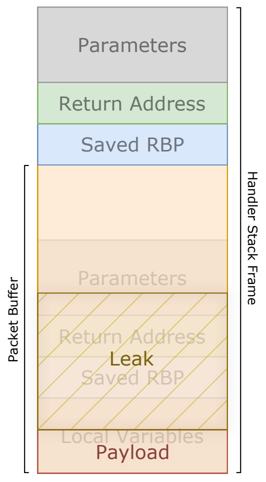
Therefore, we searched the leaks for this pattern; specifically the adjacent saved RBP and return address. We got nine matched leaks after examining a small sample of 1.5M leaks, and an example is illustrated in Figure 7.

In this leak, a saved stack address `0x00007f159609fe82` is followed by the return address `0x000055d9e8e82408`, which is consistent with the stack layout in x86-64 Linux. We can also see other

[3]Possibly 1,000 bytes, the maximum length of DNS requests that this injector would respond to.



(a) Linux Stack Frame



(b) Leak Model

Figure 6: Stack Frame and Leak Model

```
0000 [redacted] ca 23 ca 23 15 7f 00 00 00 00  [redacted]·#·#······
0010 00 00 00 00 00 00 02 00 00 00 00 00 00 00 00 17  ················
0020 8f be 15 7f 00 00 5d ce c8 be 15 7f 00 00 a8 61  ······]········a
0030 [redacted] 14 73 09 96 15 7f 00 00 06 00  [redacted]·s·······
0040 00 00 00 00 00 00 [redacted] 04 00  ······[redacted]··
0050 00 00 00 00 00 00 05 00 00 00 00 00 00 00 30 76  ··············0v
0060 09 96 15 7f 00 00 82 fe 09 96 15 7f 00 00 08 24  ···············$
0070 e8 e8 d9 55 00 00 27 00 00 00 00 00              ···U··'·····
```

Figure 7: Example Leak with Stack Frame Patterns

data that looks like pointers, all 8-byte aligned with the start of our payload. All indications highly suggest that the leaks came from a packet buffer on the stack.

4.2.3 Validating the Stack Frames. We found evidence supporting our model, but we needed to validate the stack frames as well.

First, we analyzed the other leaks with automatic tools to reduce our direct access to raw data. We found about 38K leaks with adjacent saved RBPs and return addresses of the same format from different observation sites throughout our measurements, 16K of which have exactly the same layout as in Figure 7 with slightly different addresses (due to address space layout randomization; details in §4.4). This excluded the possibility that the stack frames were random network traffic.

Second, we checked the stack signatures of Linux on AArch64, whose address prefixes differ from Linux on x86-64 and got no matches. This excluded the possibility that the GFW was also using AArch64 processors.

At last, we verified the time distribution over which we collected these stack frames. The stack frames were very rare because any network packet longer than 142 bytes would completely wipe them from our leaks, so we should have collected more stack frames when there was less traffic. To expand the sample size, we included the leaks with at least two consecutive addresses (64-bit integers) beginning with `0x00007f` (the prefix of addresses allocated by `mmap()`, including the stacks) and got 9M results. The distribution on 10/30 is illustrated in Figure 8 and conforms to the characteristics of China's international traffic in [34] substantially: when the traffic was heavy between 12:00 p.m. and 5:00 p.m. and between 8:00 p.m. and 2:00 a.m., we got fewer leaks; when the traffic was light during the rest of the time, we got more. The number dropped around 11:00 a.m. because the nodes in Guangzhou were patched at the time (details in §4.5).

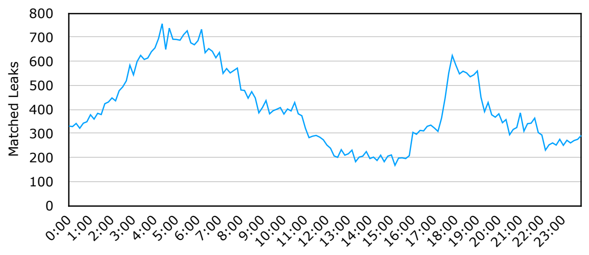


Figure 8: Time Distribution of Matched Leaks on 10/30 (CST)

4.3 Traffic Patterns

The Leak of sensitive information is the first concern associated with such a vulnerability. Despite the small leak size and the popularity of encryption nowadays, sensitive information might still be transmitted as plaintext and included in the exact part, such as passwords and tokens in various protocols.

We made tools to analyze the leaked data automatically by looking for certain signatures and patterns with regular expressions. We examined all the 13B leaks and looked for several patterns in common protocols. The results are shown in Table 2.

From the results, we collected more than 3M pieces of potentially sensitive data over a short period at a controlled low rate, which was a huge volume. The credential signatures ranged from HTTP to SMTP, and there could be many other susceptible unencrypted protocols. Since one could harvest the data much faster and it involved a whole country's transnational traffic, the risk was of great significance.

Although through our manual inspection of a small number of cases, most of the data did not contain the full credentials and/or enough information to infer the service locations (as the leaked data did not include connection metadata), stealthy adversaries might still use the partial data to help brute-force attacks or just store the data for future reference. Besides, an adversary could proactively trigger desired traffic (e.g., DNS queries) and capture it to launch off-path attacks, as described in §5.1.

The results also implied that the protocol-specific packet injector handled most network traffic going through the GFW, if not all, as hypothesized in [15].

4.4 Process Characteristics

We inferred that the leaked data came from the stack in §4.2, and we can deduce more characteristics of the GFW's processes.

First, we knew from the addresses that the GFW nodes had address space layout randomization (ASLR) and Position Independent Executable (PIE) enabled. ASLR randomizes the addresses of a process's segments on each run to mitigate some memory corruption attacks, and PIE makes ASLR possible for the code segment. In the leaks of the same layouts, the middle bits of the stack and return

Table 2: Patterns of Common Protocols

Regular Expression	Hits	Positivity Rate (‰)	Description
`HTTP\/1\.[01]`	52,851,094	38.30	HTTP/1.x protocol signature
`Authorization:`	984,567	0.71	HTTP header for tokens
`Cookie:`	1,937,274	1.40	HTTP header for cookies
`&password=`	79,090	0.06	Password in URLs/forms
`"password":"`	7,144	0.005	Password in JSON
`AUTH (PLAIN|LOGIN|CRAM-MD5)`	59,326	0.04	Credentials for SMTP & IMAP
`USER .+\r\nPASS .+`	8	/	Credentials for POP3 & FTP

addresses varied in conformity with ASLR and PIE. For example, all of the 16K leaks of the layout in Figure 7 have return addresses starting with `0x00005` and ending with `0x408` and stack addresses starting with `0x00007f` and ending with `0xe82`.

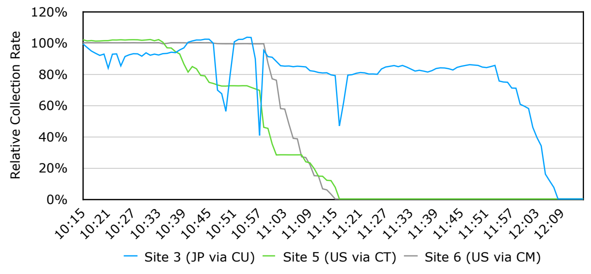
Second, we learned from the address prefixes that the GFW nodes were running x86-64, i.e., stack addresses started with `0x00007f`, and functions in executables started with `0x00005`. In our local experiments, Linux on ARM64 always allocated executable addresses starting with `0x0000aaaa` and stack addresses starting with `0x0000ffff`.

Third, we did not see any stack canaries in leaked stack frames. A stack canary is a random 64-bit integer starting with `\x00` on the stack before the saved RBP to mitigate buffer overflow attacks. We found no leak with a saved RBP and a return address conforming to this pattern. The GFW might not enable it because of performance considerations or outdated compilers. This implies that if buffer overflow vulnerabilities existed in the GFW, they might not be effectively mitigated.

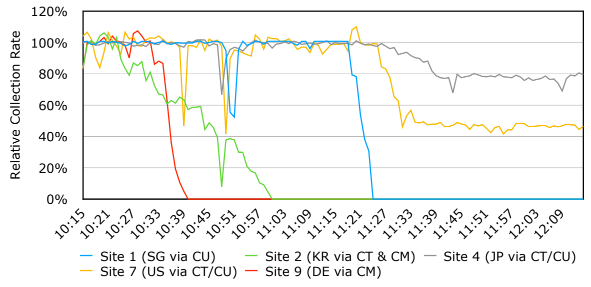
At last, we inferred that each injector process had four threads for packet handling. A unique return address in the same stack frame layout indicated a separate process due to ASLR (as it is globally shared), and one unique stack indicated a thread. We analyzed several stack frame layouts, including the one in Figure 7, and counted the unique stack addresses in the saved RBP. We found that each return address corresponded to up to four groups of stack addresses. The addresses in a group differed within 8 MB (the default stack size on Linux), indicating they were in the stack of one thread. The portion of return addresses with four groups is about one-third. This implied that every injector process might have four handler threads for network packets.

4.5 Maintenance and Update

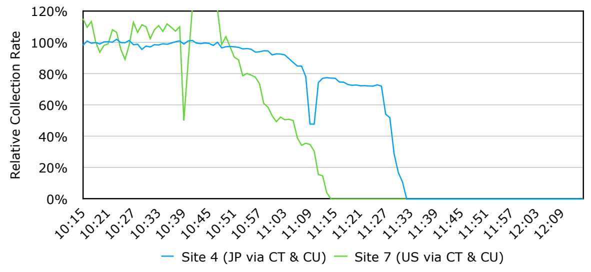
The vulnerability existed in all known paths across the GFW when we identified it in early September 2023, including all major ISPs and landing cities. Then, we started to design the experiments and conducted initial measurements. However, when we were to start the formal measurements in late October, payloads entering China via Beijing no longer worked. The GFW in China Education and Research Network (CERNET) might be patched earlier than the nodes in Beijing as a separate system, though it also resides in Beijing [2]. The nodes in Guangzhou were patched on October 30. The nodes in Shanghai were patched in two steps: on October 31, our observation sites in Singapore, South Korea, and Germany stopped working, and the collection rate in Japan and the USA declined; on November 1, the sites in Japan and the USA stopped working completely. All times and dates in this section are in China Standard Time (UTC+8).



(a) Affected Sites When Guangzhou Was Patched on 10/30



(b) Affected Sites When Shanghai Was Partially Patched on 10/31



(c) Affected Sites When Shanghai Was Completely Patched on 11/01

Figure 9: Collection Rates during Patching

We summarize the timeline as follows:

- Sometime between 10/20 and 10/26: Nodes in CERNET were patched.
- Sometime between 10/20 and 10/29: Nodes in Beijing were patched.

- October 30: Nodes in Guangzhou were patched.
- October 31: Some nodes in Shanghai were patched.
- November 1: Remaining nodes in Shanghai were patched.

We recorded the collection rates in leaks per second during the patching of Guangzhou and Shanghai, which are presented in Figure 9. The baseline is the average collection rate of the corresponding site during the last hour before it was affected by patching, and the minor fluctuations were likely due to packet loss. Per the diagrams, the updates all happened around 11 a.m. The update was divided into phases even within the same city on the same day. Each update phase affected our sites independently, except the sites in Japan and the USA lost 20% and 50% of their rates, respectively, on 10/31.

Beijing, Shanghai, and Guangzhou are the cities where international IXPs in China reside [21]. According to public information on *Submarine Cable Map* [9], Beijing has the fewest international submarine cables while Shanghai has the most. We hence hypothesize the GFW's maintenance process as follows: the GFW is updated from the city with the least international traffic to the city with the most, and the nodes in Shanghai are updated in two steps on two days to observe and minimize potential side effects; the nodes involved in each update phase are largely related to the combination of the ISP and country (possibly the corresponding cable); CERNET may be the GFW's experimental field as it has the least international bandwidth and was patched earliest.

5 POTENTIAL ATTACKS

In §4.3, we analyzed the patterns of leaked traffic and posed the possibility of a stealthy adversary passively collecting sensitive information by exploiting this vulnerability. However, it was feasible for adversaries to launch more proactive attacks leveraging it.

In this section, we describe the potential attacks induced by this vulnerability from two perspectives: off-path eavesdropping in §5.1 and the leakage itself in §5.2.

5.1 Off-Path Attacks

This vulnerability enabled any adversary on the Internet to intercept parts of the transnational packets, hence allowing off-path attacks. A proactive adversary could trigger desired traffic by initiating requests and leak sensitive data when two systems communicated across the GFW. Our experiment in §4.2.1 already showed one could leak one out of 1,000 packets with merely 5,000 exploits, while a malicious adversary could send payloads much faster to increase the chance. We have also seen cases where sensitive information was leaked despite the small leak size and inability to retrieve the first few bytes in §4.3.

One feasible attack is DNS cookie theft. DNS is vulnerable to various attacks due to a lack of encryption and authentication [29], and DNS cookies [10] were introduced as a lightweight security mechanism to mitigate DNS cache poisoning and amplification attacks. In short, a client includes an 8-byte cookie in requests, and the server echoes it in the responses to eliminate source IP spoofing; the server may also generate a cookie of 8 to 32 bytes for this client and require the client to include it in every request for verification. The adoption of DNS cookies can effectively prevent cache poisoning and amplification attacks from off-path adversaries.

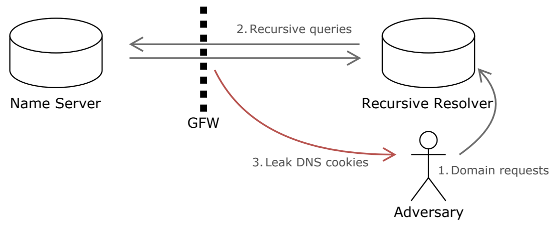


Figure 10: Threat Model of Off-Path DNS Cookie Theft

However, with our vulnerability, it was feasible to intercept the DNS cookies since they are at the end of the DNS packets. Off-path adversaries could send many queries to a recursive resolver and try to capture the traffic between it and a name server across the GFW, as shown in Figure 10. In fact, after scanning a small sample of 160K leaks, we already found 30 DNS cookie signatures in the wild and confirmed they all contained full DNS cookies. The default lifetime of DNS cookies is one day per RFC 7873. So after the adversary acquired the cookies, they could perform DNS cache poisoning by stacking previous vulnerabilities or techniques like [5, 8, 17, 22, 25, 26]. The adversary might also spoof queries from a victim and try to capture the server cookies in the error responses to exploit resolvers enforcing server cookies for amplification attacks.

Another example is attacks on transnational database connections, where the database was located in China, but there was an overseas application server for local users. An adversary could initiate operations like logins to make the server fetch sensitive information and try to eavesdrop on it, as most database protocols are not encrypted by default.

5.2 Reflective Amplification Attack

Middleboxes for censorship can be used as amplifiers for DDoS attacks as they send packets on detection of certain traffic. However, the GFW was considered a weak one with an amplification factor of only 1.45× in [14].

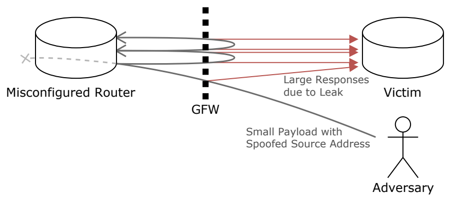

Misconfigured Router
GFW
Large Responses due to Leak
Victim
Small Payload with Spoofed Source Address
Adversary

Figure 11: Threat Model of Reflective Amplification Attacks

Even so, with the out-of-bounds read vulnerability in this work, we could increase the amplification factor significantly. Our shortest payload packet had only 46 bytes in total, and the GFW would respond with 186 bytes, bringing the amplification factor to 4.04×. We could further increase it with routing loops described in [14], where the packets crossed the GFW until their TTL expired or even indefinitely. The threat model is illustrated in Figure 11. In our experiment, we sent queries for blocked domains to IP addresses in China and found around 1,000 destinations looping packets across the GFW over 30 times, 159 of which still existed after two days. The highest number of loop iterations is 119 packets in return for only one query with a TTL of 64. This would result in an amplification factor of 481.17×. With factors at this level, one could achieve an attack volume of 100 Gbps with only a little more than 200 Mbps of initial traffic, higher than 99.9% of the DDoS attacks in 2022 Q3 [30]. The factor might be even larger if we specified the TTL to the maximum value of 255.

There are many benefits for adversaries to leverage a national censor to launch DDoS attacks. First, the GFW is located beside international IXPs with very high capacity. Despite the relatively low amplification factor, the upper limit of the achievable throughput is high. Second, the adversaries can use a large number of IP addresses to launch attacks. An adversary outside the GFW can inflict attack traffic by sending traffic to any IP address in China, making the reflection come from a variety of source IP addresses and, therefore, difficult to block. Third, the attacks can be stealthy as all traffic comes from the backbone networks of China, which is hard to trace and potentially bypasses inbound source address validation.

6 DISCUSSION

The vulnerability presented in this work reveals a new attack surface, i.e., even on-path censors can be compromised due to their implementation flaws. In this section, we discuss other possible attack vectors and the derived threats to data security.

6.1 Potential Attack Vectors

The GFW takes every measure to reinforce its censorship, including stacking multiple systems for the same type of traffic. For instance, the GFW has three packet injectors for DNS poisoning [3, 13] and two middleboxes for blocking HTTPS [16] running in parallel. There are other co-located systems for active probing [11], Great Cannon [27], etc. This complex combo increases the success rate of censorship but also introduces potential attack vectors, which await further research.

This attack surface applies to other on-path censors besides the GFW as well, like other national censors. Some ISPs in China also deploy their own censors at the provincial level [33], which may have independent attack vectors. We hope this work can spur more in-depth studies on similar issues in the censors worldwide.

6.2 Data Threats of Flawed Censors

The censors operated by nations or ISPs are usually closed-source black boxes developed in-house. Due to the lack of external audits, they may be less secure than open-source or commercial products. In the meantime, these censors filter a large volume of traffic on critical paths at all times, a large amount of which is still plaintext, despite the increasing popularity of encryption. It could cause disastrous consequences if malicious adversaries managed to compromise them. Just as we demonstrated in this work, anyone on the Internet could easily collect millions of sensitive data or facilitate DNS cache poisoning attacks with merely one vulnerability of the GFW.

Previous research mainly focused on characterizing the censors' behaviors and evading the censorship. We hope this work can draw attention to the risk of data security induced by defective censors.

7 ETHICS

Censorship research carries an element of risk, especially when it might involve sensitive information or overwhelm the networks in this work. We take the responsibility seriously and try to minimize the side effects and repercussions to the greatest extent possible.

First, the leaked data we collected from the GFW contained other users' Internet traffic, which is sensitive. To mitigate the risk of exposing such data, we rented a dedicated server from a reputable provider to store and process it and adopted TLS for the transmission from the observation sites. The observation sites did not store the data. We programmed automatic tools to analyze the data to minimize our direct access. After the analysis, all the disks were wiped to prevent a secondary leak.

Second, to minimize the impact on the other hosts and networks, we excluded the known online hosts and limited the TTL of our outgoing packets during our measurements. We controlled the sending rate to each routable IP address under 40 PPS and 2 KB/s so it would not cause stress to the host even if the packets did reach it. While scanning the Internet for routing loops, we kept the rate low and sent less than 60 bytes each time. All sending and receiving rates were negligible compared to the available bandwidth of both our host providers and our targets.

At last, the vulnerability has been completely fixed before the completion of this work. Therefore, nobody can further exploit it to launch attacks or harvest sensitive information.

8 CONCLUSION

In this work, we identified and studied an out-of-bounds read vulnerability in the GFW's packet injector. We localized the leak source, evaluated the data risk induced by it, and inferred information about the GFW's processes. We also proposed several feasible attacks leveraging the vulnerability. In the end, we discussed the possibilities of similar attack vectors and threats to data security induced by large censors.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments and constructive feedback. We also would like to express our special thanks to gfwrev and GFW.Report for their great work serving as the foundation for our research. Sakamoto dedicates this work to his friends.

REFERENCES

[1] 1987. Domain names - implementation and specification. RFC 1035. https://doi.org/10.17487/RFC1035
[2] 2006. Management Structure. https://web.archive.org/web/20171002170430/http://www.edu.cn/structure_1380/20060323/t20060323_158673.shtml. (Accessed on 11/02/2023).
[3] 2014. Towards a Comprehensive Picture of the Great Firewall's DNS Censorship. In *4th USENIX Workshop on Free and Open Communications on the Internet (FOCI 14)*. USENIX Association, San Diego, CA. https://www.usenix.org/conference/foci14/workshop-program/presentation/anonymous
[4] 2014. Internet outage in China on Jan 21 | GreatFire.org. https://zh.greatfire.org/blog/2014/jan/internet-outage-china-jan-21. (Accessed on 11/01/2023).
[5] 2020. CVE-2020-25684. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-25684
[6] 2020. CVE-2020-27737. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-27737
[7] 2020. GFW Archaeology: gfw-looking-glass.sh. https://gfw.report/blog/gfw_looking_glass/en/. (Accessed on 11/01/2023).

[8] 2021. CVE-2021-3448. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-3448

[9] 2023. Submarine Cable Map. https://www.submarinecablemap.com/

[10] Donald E. Eastlake 3rd and Mark P. Andrews. 2016. Domain Name System (DNS) Cookies. RFC 7873. https://doi.org/10.17487/RFC7873

[11] Alice, Bob, Carol, Jan Beznazwy, and Amir Houmansadr. 2020. How China Detects and Blocks Shadowsocks. In *Proceedings of the ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC '20)*. Association for Computing Machinery, New York, NY, USA, 111–124. https://doi.org/10.1145/3419394.3423644

[12] Daniel Anderson. 2012. Splinternet Behind the Great Firewall of China: Once China Opened Its Door to the World, It Could Not Close It Again. *Queue* 10, 11 (nov 2012), 40–49. https://doi.org/10.1145/2390756.2405036

[13] 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

[14] Kevin Bock, Abdulrahman Alaraj, Yair Fax, Kyle Hurley, Eric Wustrow, and Dave Levin. 2021. Weaponizing Middleboxes for TCP Reflected Amplification. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 3345–3361. https://www.usenix.org/conference/usenixsecurity21/presentation/bock

[15] 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 *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (Virtual Event, USA) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 586–598. https://doi.org/10.1145/3387514.3405889

[16] Kevin Bock, Gabriel Naval, Kyle Reese, and Dave Levin. 2021. Even Censors Have a Backup: Examining China's Double HTTPS Censorship Middleboxes. In *Proceedings of the ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet* (Virtual Event, USA) *(FOCI '21)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3473604.3474559

[17] Tianxiang Dai, Haya Shulman, and Michael Waidner. 2021. DNS-over-TCP Considered Vulnerable. In *Proceedings of the Applied Networking Research Workshop* (Virtual Event, USA) *(ANRW '21)*. Association for Computing Machinery, New York, NY, USA, 76–81. https://doi.org/10.1145/3472305.3472884

[18] Stanislav Dashevskyi, Daniel dos Santos, Jos Wetzels, and Amine Amri. 2022. Common Implementation Anti-Patterns Related to Domain Name System (DNS) Resource Record (RR) Processing. RFC 9267. https://doi.org/10.17487/RFC9267

[19] gfwrev. 2009. GFW Tech Review: Deeper Understanding of the GFW: DNS Poisoning. https://gfwrev.blogspot.com/2009/11/gfwdns.html. (Accessed on 11/06/2023).

[20] gfwrev. 2010. GFW Tech Review: Deeper Understanding of the GFW: Internal Structure. https://gfwrev.blogspot.com/2010/02/gfw.html. (Accessed on 11/01/2023).

[21] Lisa Hanson. 2015. The Chinese Internet Gets A Stronger Backbone. https://www.forbes.com/sites/lisachanson/2015/02/24/the-chinese-internet-gets-a-stronger-backbone/?sh=35d9a5d91ff4. (Accessed on 11/3/2023).

[22] Amir Herzberg and Haya Shulman. 2013. Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org. In *2013 IEEE Conference on Communications and Network Security (CNS)*. 224–232. https://doi.org/10.1109/CNS.2013.6682711

[23] Erik Hjelmvik. 2013. Forensics of Chinese MITM on GitHub. https://www.netresec.com/?page=Blog&month=2013-02&post=Forensics-of-Chinese-MITM-on-GitHub. (Accessed on 11/01/2023).

[24] Nguyen Phong Hoang, Arian Akhavan Niaki, Jakub Dalek, Jeffrey Knockel, Pellaeon Lin, Bill Marczak, Masashi Crete-Nishihata, Phillipa Gill, and Michalis Polychronakis. 2021. How Great is the Great Firewall? Measuring China's DNS Censorship. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 3381–3398. https://www.usenix.org/conference/usenixsecurity21/presentation/hoang

[25] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. 2020. DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1337–1350. https://doi.org/10.1145/3372297.3417280

[26] Keyu Man, Xin'an Zhou, and Zhiyun Qian. 2021. DNS Cache Poisoning Attack: Resurrections with Side Channels. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 3400–3414. https://doi.org/10.1145/3460120.3486219

[27] Bill Marczak, Nicholas Weaver, Jakub Dalek, Roya Ensafi, David Fifield, Sarah McKune, Arn Rey, John Scott-Railton, Ron Deibert, and Vern Paxson. 2015. An Analysis of China's "Great Cannon". In *5th USENIX Workshop on Free and Open Communications on the Internet (FOCI 15)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/foci15/workshop-program/presentation/marczak

[28] Robert McMillan. 2010. After DNS problem, Chinese root server is shut down | Computerworld. https://www.computerworld.com/article/2755924/after-dns-problem--chinese-root-server-is-shut-down.html. (Accessed on 11/01/2023).

[29] Anju Ramdas and Ramakrishnan Muthukrishnan. 2019. A Survey on DNS Security Issues and Mitigation Techniques. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*. 781–784. https://doi.org/10.1109/ICCS45141.2019.9065354

[30] Cloudflare DDoS Protection Team. 2022. DDoS Attack Trends for 2022 Q3. https://radar.cloudflare.com/reports/ddos-2022-q3. (Accessed on 11/04/2023).

[31] 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 *Proceedings of the 2017 Internet Measurement Conference* (London, United Kingdom) *(IMC '17)*. Association for Computing Machinery, New York, NY, USA, 114–127. https://doi.org/10.1145/3131365.3131374

[32] Mingshi Wu, Jackson Sippe, Danesh Sivakumar, Jack Burg, Peter Anderson, Xiaokang Wang, Kevin Bock, Amir Houmansadr, Dave Levin, and Eric Wustrow. 2023. How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 2653–2670. https://www.usenix.org/conference/usenixsecurity23/presentation/wu-mingshi

[33] 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.

[34] Pengxiong Zhu, Keyu Man, Zhongjie Wang, Zhiyun Qian, Roya Ensafi, J. Alex Halderman, and Haixin Duan. 2020. Characterizing Transnational Internet Performance and the Great Bottleneck of China. *Proc. ACM Meas. Anal. Comput. Syst.* 4, 1, Article 13 (may 2020), 23 pages. https://doi.org/10.1145/3379479