How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic

Mingshi Wu, *GFW Report;* Jackson Sippe, *University of Colorado Boulder;* Danesh Sivakumar and Jack Burg, *University of Maryland;* Peter Anderson, *Independent researcher;* Xiaokang Wang, *V2Ray Project;* Kevin Bock, *University of Maryland;* Amir Houmansadr, *University of Massachusetts Amherst;* Dave Levin, *University of Maryland;* Eric Wustrow, *University of Colorado Boulder*

https://www.usenix.org/conference/usenixsecurity23/presentation/wu-mingshi

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

USENIX'23 Artifact Appendix: How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic

Mingshi Wu
GFW Report

Jackson Sippe
University of Colorado Boulder

Danesh Sivakumar
University of Maryland

Jack Burg
University of Maryland

Peter Anderson
Independent researcher

Xiaokang Wang
V2Ray Project

Kevin Bock
University of Maryland

Amir Houmansadr
University of Massachusetts Amherst

Dave Levin
University of Maryland

Eric Wustrow
University of Colorado Boulder

April 20, 2023, v1.0

A Artifact Appendix

A.1 Abstract

As introduced in the paper, we measure and characterize the GFW's new system for censoring fully encrypted traffic. We find that, instead of directly defining what fully encrypted traffic is, the censor applies crude but efficient heuristics to exempt traffic that is unlikely to be fully encrypted traffic; it then blocks the remaining non-exempted traffic. These heuristics are based on the fingerprints of common protocols, the fraction of set bits, and the number, fraction, and position of printable ASCII characters. In this artifact, we provide the data and code to support our major claims. Additionally, we conducted a follow-up experiment to confirm that the GFW had stopped blocking fully encrypted traffic dynamically as of Wednesday, March 15, 2023.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

As detailed in the ethics section of our paper, our measurement tools have already employed the best practices by default.

A.2.2 How to access

The artifact is available on GitHub: https://github.com/gfw-report/usenixsecurity23-artifact/commit/ad45e63b4a708bda5ce39f48fc25ebbae013ee51.

A.2.3 Hardware dependencies

We have prepared a VPS in China and a VPS in the US, on which the AE reviewers can perform remote experiments. The VPS in China is located in AlibabaCloud Beijing Datacenter (AS37963), which uses one core of Intel Xeon Platinum 8163 and 1GB RAM. The VPS in the US is located in DigitalOcean San Francisco Datacenter (AS14061), which uses one core of Intel DO-Regular and 1GB RAM. To SSH into the VPSes, reviewers need to install the provided credentials, as detailed in `artifacts/setup-vps/README.md`.

For people other than the AE reviewers who want to perform experiments, they need to prepare a VPS in China and a VPS outside of China themselves.

A.2.4 Software dependencies

The VPS in China runs Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-56-generic x86_64). The VPS in the US runs Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-88-generic x86_64). The following tools and environment are required:

- GNU make utility
- Go 1.17+
- Python 3.8+

In particular, the two VPSes do not require Go environment. As detailed in `README`, reviewers may compile the Go code on their local machines and copy the binaries to the VPSes.

A.2.5 Benchmarks

None.

A.3 Set-up

As detailed in `artifacts/setup-vps/README.md`, we have prepared a VPS in China and a VPS in the US, on which the AE reviewers can perform remote experiments. Since we have installed the dependencies and required tools on the VPSes, all reviewers need to do is to use the provided credentials to SSH into the VPSes to run experiments. We also provide one-click scripts, allowing reviewers to initialize test-ready VPSes themselves. Be very cautious that running setup scripts will disrupt other reviewers' ongoing experiments.

A.3.1 Installation

- Set up the Go environment: `https://go.dev/dl/`.
- Retrieve the artifacts: `git clone https://github.com/gfw-report/usenixsecurity23-artifact`.
- Compile the client-side experiment tools and install them to the CN VPS: `cd artifacts/setup-vps && ./setup-client/to_alibaba_server.sh`.
- Compile the sink server and install it to the US VPS: `cd artifacts/setup-vps && ./setup-server/to_digitalocean_server.sh`.

A.3.2 Basic Test

First login to the VPS in China using the provided credentials: `ssh usenix-ae-client-china`.

Then send random probes to the port 80 of the $*serverIP* with the following command: `echo $serverIP | ./utils/affected-norand -p 80 -log /dev/null`.

The program outputs in CSV format. If the `affected` field is `True`, the blocking is successfully triggered. If the `affected` field is `False`, the blocking is not triggered.

A.4 Evaluation workflow

A.4.1 Major Claims

(C0): As of Tuesday, March 7, 2023, the GFW was still blocking random traffic. This is supported by the experiment (E0).

(C1): As of Wednesday, March 15, 2023, the GFW had stopped blocking random traffic dynamically. This is supported by the experiment (E1).

(C2): The GFW exempts a connection if the first TCP packet `pkt` satisfies: $\frac{popcount(\texttt{pkt})}{len(\texttt{pkt})} \leq 3.4$ or $\frac{popcount(\texttt{pkt})}{len(\texttt{pkt})} \geq 4.6$. This is supported by the experiment (E2) described in Section 4.1 of the paper. This detection rule is introduced in Algorithm 1 (Ex1) and illustrated in Figure 1.d.

(C3): The GFW exempts a connection if the first six (or more) bytes of the first TCP data packet `pkt` are [`0x20`, `0x7e`]. This is supported by the experiment (E3) described in Section 4.2 of the paper. This detection rule is introduced in Algorithm 1 (Ex2) and illustrated in Figure 1.a.

(C4): The GFW exempts a connection if the first TCP data packet `pkt` has more than 50% of `pkt`'s bytes in [`0x20`, `0x7e`]. This is supported by the experiment (E4) described in Section 4.2 of the paper. This detection rule is introduced in Algorithm 1 (Ex3) and illustrated in Figure 1.b.

(C5): The GFW exempts a connection if the first TCP data packet `pkt` has more than 20 contiguous bytes in [`0x20`, `0x7e`]. This is supported by the experiment (E5) described in Section 4.2 of the paper. This detection rule is introduced in Algorithm 1 (Ex4) and illustrated in Figure 1.c.

(C6): The GFW exempts a connection if the first few bytes of the first TCP data packet `pkt` match the protocol fingerprint for TLS or HTTP. This is supported by the experiment (E6) described in Section 4.3 of the paper. This detection rule is introduced in Algorithm 1 (Ex5) and illustrated in Figure 1.e.

A.4.2 Experiments

Experiment E0 tests if the GFW still blocks random traffic dynamically by sending random probes from China to a single server in US. If the reviewers can trigger the blocking in experiment E0, they can proceed to test experiments (E1-E6); otherwise, they only need to run experiment E1 to further confirm the GFW has stopped blocking random traffic.

Experiments E0 and E2-E6 follow the same testing logic: we craft different payloads that will be either exempted or blocked by the GFW. We send them, from VPS in China to the VPS in US, through the GFW, to observe whether each payload can trigger the blocking or not. If the blocking or exemption results match with what Algorithm 1 predicts, it shows that the GFW is indeed using the detection rule described in Algorithm 1. For reviewers' convenience, we implement Algorithm 1 as `utils/detect.py`, which reads a list of given payloads in hex format and writes if each payload will be exempted by any of the detection rules.

Unless explicitly specified, all operations described below are performed on the VPS in China. And $*serverIP* corresponds to the IP address of the VPS in US.

(E0): *[test-random] [5 human-minutes + 5 compute-minutes]:* This experiment tests if the GFW blocks random traffic. It also familiarizes the reviewers with the testing tools and logic.

Preparation: `cd artifacts`

Execution: Execute this command to generate a random probe of 200 bytes and check if Algorithm 1 thinks it will be blocked by the GFW or not: `head -c200 /dev/urandom | xxd -p -c256 | tee random.txt | ./utils/detect.py`.

Execute this command to repeatedly send the probe to the same port of the US server: `cat random.txt | ./utils/affected-payload -host $serverIP -p $serverPort`.

Results: If the `affected` field in the program output is `True`, it means that your generated probe has triggered the blocking by the GFW. This result should be consistent with what `detect.py` predicts.

(E1): *[confirm-ceased-blocking] [15 human-minutes + 2 compute-days]:* This experiment tests if the GFW has stopped blocking random traffic dynamically. Specifically, it performs an Internet scan from a VPS machine

in China to all 142,827 IP addresses that were previously marked affected as of August 22, 2022. For each IP address, The test uses two types of probes: 50-bytes of random data and 50-bytes of zero (as the control group). For each type of probe, the program makes up to 25 connections, and when five consecutive connections to an IP address fail, the program mark it as possibly affected. We then remove any probes that were also marked as affected in the control group to rule out most of the false positives due to network failure rather than censorship.
Preparation: `cd ceased-dynamic-blocking`
Execution: Execute this command to perform the 2-day test: `make`.
One then compares the results between the two tests using two different types of probes, to find the IP addresses that are marked as blocked (`true`) in the random probe test but marked as not blocked (`false`) in the zero probe test: `make compare`.
Results: The number of affected IP addresses should be as low as around six thousand out of the 142,827 IP addresses tested. One can further test these IP addresses recursively to make sure they are all false positives.

(E2): *[test-entropy] [30 human-minutes + 30 compute-minutes]:* This experiment test if the GFW exempts a connection whose first TCP packet `pkt` satisfies: $\frac{\mathit{popcount}(\texttt{pkt})}{\mathit{len}(\texttt{pkt})} \leq 3.4$ or $\frac{\mathit{popcount}(\texttt{pkt})}{\mathit{len}(\texttt{pkt})} \geq 4.6$.
Preparation: `cd test-entropy`
Execution: Execute this command to generate a list of payloads: `make`. As shown in the output of `detect.py`, some of the probes will be exempted by the GFW; while other probes will not.
Use this command to test if each probe is exempted by the GFW: `make test`.
Use this to compare the blocking results against the results predicted by the `detect.py`: `make compare`.
Results: The testing results should match with what `detect.py` predicts.

(E3): *[test-printable-prefixes] [15 human-minutes + 30 compute-minutes]:* This experiment tests if the GFW exempts a connection whose first six bytes are printable characters.
Preparation: `cd test-printable-prefixes`
Execution: Execute this command to generate a list of payloads: `make`. As shown in the output of `detect.py`, some of the probes will be exempted by the GFW; while other probes will not.
Use this command to test if each probe is exempted by the GFW: `make test`.
Use this to compare the blocking results against the results predicted by the `detect.py`: `make compare`.
Results: The testing results should match with what `detect.py` predicts.

(E4): *[test-printable-fraction] [15 human-minutes + 30 compute-minutes]:* This experiment tests if the GFW exempts a connection whose first TCP data packet has more than 50% of printable characters.
Preparation: `cd test-printable-fraction`
Execution: Execute this command to generate a list of payloads: `make`. As shown in the output of `detect.py`, some of the probes will be exempted by the GFW; while other probes will not.
Use this command to test if each probe is exempted by the GFW: `make test`.
Use this to compare the blocking results against the results predicted by the `detect.py`: `make compare`.
Results: The testing results should match with what `detect.py` predicts.

(E5): *[test-printable-longest-run] [15 human-minutes + 15 compute-minutes]:* This experiment tests if the GFW exempts a connection whose first TCP data packet has more than 20 bytes of contiguous printable characters.
Preparation: `cd test-printable-longest-run`
Execution: Execute this command to generate a list of payloads: `make`. As shown in the output of `detect.py`, some of the probes will be exempted by the GFW; while other probes will not.
Use this command to test if each probe is exempted by the GFW: `make test`.
Use this to compare the blocking results against the results predicted by the `detect.py`: `make compare`.
Results: The testing results should match with what `detect.py` predicts.

(E6): *[test-protocol-fingerprints] [15 human-minutes + 2 compute-hours]:* This experiment tests if the GFW exempts traffic that matches the protocol fingerprints.
Preparation: `cd test-protocol-fingerprints`
Execution: Execute this command to generate a list of payloads: `make`. As shown in the output of `detect.py`, some of the probes start with a fingerprint that will be exempted by the GFW; while other probes do not.
Use this command to test if each probe is exempted by the GFW: `make test`.
Use this to compare the blocking results against the results predicted by the `detect.py`: `make compare`.
Results: The testing results should match with what `detect.py` predicts.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.