USENIX Security '26 Artifact Appendix: Revelio: A Global Study of VoIP Blocking

Arjun Gupta
IIT Bombay

Roya Ensafi
University of Michigan

Piyush Kumar
IIT Delhi

Devashish Gosain
IIT Bombay

A Artifact Appendix

A.1 Abstract

We present *Revelio*, a tool for detecting VoIP blocking at a global scale. *Revelio* uses interference in STUN traffic as a signal to detect VoIP blocking. *Revelio* supports scraping BGP prefix information, large-scale STUN probing using replayed VoIP App packets, STUN response deviation analysis, and traceroute-based middlebox inference.

Our artifact consists of two main components, (1) The codebase, which can be used for running the entire *Revelio* scans worldwide, and (2) The data we collected during our study, which can be used to reproduce the graphs from the paper.

A.2 Description & Requirements

Below, we detail the modules from both our codebase and data that are in accordance with the open science policy outlined in the accepted paper.

Codebase

- Scraping: it is used to web-scrape country ASNs and prefix data, merge and clean prefixes for further scanning.
- Zstun: perform protocol-level STUN probes using Zmap-assisted pipelines.
- Analysis & STUN Trace: post-process Zstun output and to produce per-country, analysis-ready graphs, filtered destination IPs and middleboxes.

Data

- Hyperquack data: remote measurement data for website blocking regarding VoIP apps (*graphs_from_paper/input/dataset_4*).
- Stun server IPs: censored STUN server IPs released after anonymizing the first two octets (*graphs_from_paper/input/dataset_i_anon*) where "i" can be 1, 2 or 3. This data is used to reproduce the graphs from the paper.

A.2.1 Security, privacy, and ethical concerns for the evaluators

Running *Revelio* poses no security, privacy, or ethical concerns for the evaluators in the uncensored region. However, if the evaluator is residing in a censored region (especially where VoIP is blocked), we recommend exercising caution when running the tool, as it involves sending potentially blocked STUN probes.

A.2.2 How to access

- The repository containing the code and data can be cloned from the following Github link.
- Archived versions of the artifact are available under the following Zenodo DOI.

A.2.3 Hardware dependencies

We recommend scanning machines provisioned with atleast 8 GB of RAM, 2 Intel vCPUs, and a 160 GB SSD, running Ubuntu 22.04 LTS (64-bit). Furthermore, since *Revelio* scans the IPv4 address space using `Zstun` (which is built on `Zmap` version 4.3.4), it requires a high network bandwidth for sending probes (we used 300 Mbps).

A.2.4 Software dependencies

We package all our dependencies as part of the code. Refer to section A.3 for detailed steps.

A.2.5 Benchmarks

None

A.3 Set-up

The repository provides a helper setup script (*setup.sh*) in the root directory to install the required dependencies and prepare the environment. At the end of execution, the script also verifies whether the installation was successful.

A.3.1 Installation

Run ***bash setup.sh*** to install the following dependencies and verify that each package has been installed successfully:

- Python dependencies
- Scapy and plotting libraries
- ZStun dependencies for compiling from source
- Required system packages
- Virtual environment setup components

A.3.2 Basic Test

There is a "Fast Mode" designed to perform an end-to-end run of *Revelio* with limited scope. To launch Fast Mode, please run ***bash run_fast.sh***.

As a representative example, the script only scans for the UAE IP space for WhatsApp VoIP call filtering. The output is a text file `filtered_ips_WhatsApp.txt` that contains the UAE IPs along which *Revelio* detected STUN filtering.

Next, the script attempts to estimate the location of the middleboxes. From this set of filtered IPs, the code randomly selects 10 IPs for which vanilla STUN probes yielded no response, but WhatsApp-specific STUN probes returned ICMP responses (see "No response censorship" in Section 5.3.2 of the paper).

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): `Zstun` discovers usable STUN infrastructure that cannot be reliably identified using standard Internet-wide scanning approaches.

(C2): *Revelio* detects VoIP censorship through differences in responses for vanilla STUN, application-specific STUN, and modified application-specific STUN probes.

(C3): `StunTrace` can identify and characterize censorship middleboxes from filtered destination IPs and reproduce representative examples of the middlebox behaviors in a target country.

A.4.2 Experiments

(E1): *Zstun Discovery Evaluation [5 human-minutes + 15 compute-minutes (Demo Run) OR 21 compute-hour (Comprehensive Run)]* Linked to Claim C1.
Preparation: None
Execution: Navigate to the `claims/C1/` directory. Select either `run_demo.sh` to evaluate the claim on the United Arab Emirates or `run_comprehensive.sh` to evaluate the claim on all 19 countries reported in the paper.
Results: The scripts will result in three output text files corresponding to a target country. The files are `tcp_3478.txt` (contains the Zmap TCP output), `udp_3478.txt` (contains the raw Zmap UDP output), and `stun_a.txt` (contains the ZSTUN output). Finally, the script produces `claims/C1/c1_summary.csv`, which contains only the number of responsive IPs from each of the three files for easy comparison.
The reader can observe a typical trend: Number of UDP 3478 Open IPs < Number of STUN responsive IPs < Number of TCP 3478 Open IPs. This trend indicates that scanning UDP port 3478 discovers only a small fraction of usable STUN servers, whereas scanning TCP port 3478 identifies many hosts that do not actually provide STUN services. The key difference lies in the validation criterion: `Zstun` sends standard STUN probes and classifies an IP address as an operational STUN server only if it returns a valid STUN response. In contrast, ZMap's UDP/3478 and TCP/3478 scans merely indicate that the corresponding port is open and do not verify STUN functionality. Therefore, in our analysis, we treat `Zstun`-responsive IP addresses as operational STUN servers. By requiring successful STUN exchanges for identification, `Zstun` consistently discovers a stable set of genuine STUN servers, providing empirical support for the claim made in C1.

(E2): *VoIP Censorship Detection Evaluation [5 human-minutes + 18 compute-minutes]*. Linked to Claim C2.
Preparation: None
Execution: Navigate to the `claims/C2/` directory. Use `run_demo.sh` to evaluate the claim on the complete IP space of the target country. The country name and the app filtered in that country need to be passed as arguments to the script. By default, we selected the country "UAE" and the app "WhatsApp" whose STUN packets are subject to censorship in the UAE. Execute `run_comprehensive.sh` to evaluate the claim on all 19 countries reported in the paper. (Note that this should incur about 50 hours to complete the comprehensive scans from a single measurement machine.)
Results: For a target country, the `run.sh` code will generate VoIP censorship graphs. For example, if the target country is the UAE, the code will generate graphs similar to Figures 5 (UDP deviation), 7 (No response deviation), and 9–10 (ICMP deviation). Similar graphs will be created for different countries as an output after executing `run_comprehensive.sh`.
Interpretation of graphs: In the absence of filtering, we expect all three probe types (P1, P2, and P3) to show similar response behavior. However, for networks that perform VoIP filtering, P2 probes may exhibit noticeable deviations from P1 and P3 in UDP responses, ICMP errors, and non-responding IPs.

(E3): *Middlebox Localization and Characterization Evaluation [5 human-minutes + 30 compute-minutes]*. Linked to Claim C3.

Preparation: None

Execution: Navigate to the `claims/C3/` directory. Use `run.sh` to evaluate the claim on a representative set (10 IPs) of filtered destinations from each deviation category (UDP, ICMP, and No response) identified during the `Zstun` scan. The country name and the app filtered in that country need to be passed as arguments to the script. By default, we selected the country "UAE" and the app "WhatsApp" whose STUN packets are subject to censorship in the UAE. Execute `run_comprehensive.sh` to evaluate the claim on all 19 countries reported in the paper.

Results: The `run.sh` script will generate three output files for the specified target country, i.e., `<app_name>_middleboxes.txt`, `<app_name>_guessed_asn.txt`, and `<app_name>_likely_filtered_dest_IP.txt` for each deviation category. These files contain potential middlebox IPs (if visible in traces), potential ASes that perform filtering, and filtered destination IPs, respectively. In addition to these files, the vanilla and censored traces are stored in the trace_output directory. `run_comprehensive.sh` will generate such files, and the trace_output directory will be created for each of the 19 countries. (Note that this should incur about 30 hours to complete the comprehensive scans from a single measurement machine.)

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.