Creating a Secure Underlay for the Internet

Henry Birge-Lee, *Princeton University;* Joel Wanner, *ETH Zürich;*
Grace H. Cimaszewski, *Princeton University;* Jonghoon Kwon, *ETH Zürich;*
Liang Wang, *Princeton University;* François Wirz, *ETH Zürich;* Prateek Mittal,
Princeton University; Adrian Perrig, *ETH Zürich;* Yixin Sun, *University of Virginia*

https://www.usenix.org/conference/usenixsecurity22/presentation/birge-lee

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

A Artifact Appendix

A.1 Abstract

Our artifact consists of 1) the SBAS client and node code used to operate the SBAS infrastructure 2) the simulation software used in the security analysis section of the paper and 3) the raw survey results and questions from our network operator survey.

A.2 Artifact check-list (meta-information)

- **Program:** Our artifact contains two programs: 1) the SBAS node and client software that is used to operate the SBAS infrastructure and connect clients respectively and 2) the topology simulation software to run inter-domain topology simulations are perform the security analysis.
- **Compilation:** The SBAS client and node must be compiled and installed as per the instructions in the README.md file. The README.md file (in the usenix22 branch) also contains instructions for setting up a personal SBAS using two nodes that are connected over SCIONLab (and have connectivity to each other but are distinct from our current SBAS production deployment). Additionally, the README.md file contains instructions for how to connect a client this SBAS deployment. The topology simulator is written in python and can be run directly on general-purpose computing hardware and requires no compilation (although the script to graph the results requires several pip and apt dependencies).
- **Binary:** The simulation code is in python (which is interpreted) so there is no binary, but the primary source code file is simulate.py in the root directory of the simulation repo. The SBAS client and node are largely python and bash scripts and the repo contains an install script that installs SBAS as a systemd service.
- **Data set:** Our SBAS client does not require any datasets. The simulation artifact uses the CAIDA AS relationships dataset, RIPE NCC and RouteViews BGP datasets, and PEERING testbed connection data. Our survey result dataset is attached as part of our artifact submission.
- **Run-time environment:** Our simulation requires python3 and the appropriate pip3 modules installed. While our code should run on most Linux environments, **all our testing was done on Ubuntu 22.04** and this was used to generate the required dependencies and install instructions mentioned in the README.md files. We strongly encourage Ubuntu 22.04 as other variants might require different package dependencies and even other Ubuntu versions ship with different versions of python that could potentially impact script behavior.
- **Hardware:** Simulations are run using general purpose hardware.
- **Execution:** The simulation models interdomain routing attacks and outputs statistics about the security of SBAS nodes which can be graphed as a CDF (see the README.md file in the usenix22-simulations branch). The SBAS node and client software install systemd services that manage routing rules related to forwarding SBAS traffic and interact with the other routing services SBAS depends on (e.g., the SCION-IP gateway and BIRD).
- **Security, privacy, and ethical concerns:** Simulations are run on static data/configuration files and thus pose no burden on the ASes and prefixes modeled therein. They are also based entirely on publicly-available datasets. The SBAS node and client software does not violate any networking best practices and only sends IP packets for common well defined protocols.
- **Output:** The simulation outputs result files for standard and ROV SBAS experiments. The SBAS node and client software does not produce any output file per say but configures routing such that secure prefixes and customers can be reached between different SBAS nodes.
- **How much disk space required (approximately)?:** The SBAS client and node software requires only minimal disk space for dependencies to install. Running the abridged version of the simulation requires under 1GB of disk space.
- **How much time is needed to prepare workflow (approximately)?:** Simulation workflow requires only time needed to download the topology simulator repository.
- **How much time is needed to complete experiments (approximately)?:** The simulation workflow requires roughly 1 hour using a general purpose CPU.
- **Publicly available (explicitly provide evolving version reference)?:** The artifact contents are hosted at https://github.com/scion-backbone/sbas/tree/80044509e5ac1681e8d970a09e4b3187439a0938. The client and node software and configuration files are available in the `sbas` submodule; the simulation code and data, in the `sbas-simulation` submodule; and lastly, the survey results in `sbas-survey`.
- **Code licenses (if publicly available)?:** CC Zero.
- **Workflow frameworks used?:** Github.
- **Archived (explicitly provide DOI or stable reference)?:** A stable link to our artifact is available at https://github.com/scion-backbone/sbas/tree/80044509e5ac1681e8d970a09e4b3187439a0938

A.3 Description

Obligatory. For inapplicable subsections (e.g., the "How to access" subsection when not applying for the "Artifacts Available" badge), please specify 'N/A'.

A.3.1 How to access

Artifacts are available online at the URLs listed in A.2

A.3.2 Hardware dependencies

Standard computational hardware.

A.3.3 Software dependencies

Python 3.7 and standard numerical/data science packages (numpy, matplotlib, pandas). See README.md files for more details on dependency installs (which can all be done with standard packages).

A.3.4 Data sets

We rely on the CAIDA AS-relationship data set and BGP update data from RIPE NCC and RouteViews, to generate the policy files and topology used for the simulation.

A.3.5 Models

N.A.

A.3.6 Security, privacy, and ethical concerns

The simulations employ only publicly available datasets and thus do not leak any private information about interdomain connectivity.

A.4 Installation

Obligatory. Describe the setup procedures for your artifact targeting novice users (even if you use a VM image or access to a remote machine).

The full installation instructions for the node and client are included in the `sbas` submodule under the artifact repository at the URL given in A.2 (specifically, https://github.com/scion-backbone/sbas/tree/80044509e5ac1681e8d970a09e4b3187439a0938). The installation for the simulation software is described in the `sbas-simulation` submodule.

A.5 Experiment workflow

Describe the high-level view of your experimental workflow and how it is implemented, invoked and customized (if needed), i.e. some OS scripts, IPython/Jupyter notebook, portable CK workflow, etc. This subsection is optional as long as the experiment workflow can be easily embedded in the next subsection.

The BGP simulation framework takes three main files as input:

1. **CAIDA Topology:** serial-2 AS Relationships topology
2. **policies file:** BGP export/import policies to be applied to BGP announcement points
3. **origins file:** enumerates prefix announcements to be made at BGP announcement points (including hijackers)

The simulation engine runs as a Python script and writes the outcome of each simulation scenario to a text file. Another Python visualization script generates a CDF of the simulation results similar to those presented in Figures 8 & 9 of the main paper. The full experimental workflow for the simulations is described in the sbas-simulation submodule of https://github.com/scion-backbone/sbas/tree/80044509e5ac1681e8d970a09e4b3187439a0938.

For the SBAS node software, the workflow involves joining SCIONLab, connecting two machines to SCIONLab, running SBAS on those two machines and testing connectivity through SBAS and then connecting a client to one of the machines and testing connectivity to the other SBAS node. The full workflow is described in the sbas submodule of https://github.com/scion-backbone/sbas/tree/80044509e5ac1681e8d970a09e4b3187439a0938

A.6 Evaluation and expected results

Obligatory. Start by listing the main claims in your paper. Next, list your key results and detail how they each support the main claims. Finally, detail all the steps to reproduce each of the key results in your paper by running the artifacts. Describe the expected results and the maximum variation of empirical results (particularly important for performance numbers).

We package a subsample of the data used in the Internet-scale simulations (presented in Section 7.2) to model the BGP hijack resiliency gains of SBAS provides over a client making its own BGP announcements to the Internet. These simulations should output textual data showing the proportion of the Internet that will be affected by an adversary's attack for each SBAS announcement. This output is then parsed by the plotting script `plot_artifact_results.py` to plot CDFs illustrating the resilience of SBAS against different adversaries.

There are several main results which can be seen in this CDF plots even with the reduced input files used for the artifact evaluation. First, an SBAS announced-prefix has a significantly higher resilience than a non-SBAS prefix. Second, SBAS performance improves with additional nodes are added to the network of PoPs. Finally, ROV enforcement further improves the resilience of SBAS.

The primary expected result for the SBAS node code is that the pings from one SBAS node's VPN prefix to another SBAS node's VPN prefix (which are routed over SCIONLab) are sent successfully. Furthermore, for the client code, the client connected to the SBAS node should be able to ping the VPN prefix at the other SBAS node securely which implies it could communicate with other clients connected to that node's VPN.

A.7 Experiment customization

A.8 Notes

A.9 Version

Based on the LaTeX template for Artifact Evaluation V20220119.