USENIX Security '25 Artifact Appendix: BGP Vortex: Update Message Floods Can Create Internet Instabilities

Felix Stöger, Henry Birge-Lee, Giacomo Giuliari, Jordi Subira-Nieto, Adrian Perrig

A Artifact Appendix

A.1 Abstract

We provide a comprehensive artifact containing all the resources necessary to reproduce our results from sections 4 and 5 of our work. It is available to download as a ZIP archive via a persistent URL (appendix A.2.2).

A.2 Description & Requirements

The artifact submission contains all scripts and configurations required for evaluation. The ZIP archive contains two folders for replicating results from sections 4 and 5 respectively. Each folder includes a comprehensive README with step-by-step execution instructions and requirements. The artifact provides complete attack simulation components: router configurations, orchestration scripts, and notebooks for data analysis.

Replication of our experiments requires 8 virtual machines. We used DigitalOcean for evaluation. The experiments generate several hundred gigabytes of data, compressed to approximately 50GB. The setup runs on standard VMs using common software packages (e.g., Jupyter Lab), requiring no specialized hardware or software. For the functional and reproduced artifact evaluation, we will provide you with 8 VM instances to ensure the setup is indeed identical to the one used for the experiments. Please contact us, so we can set the VMs up and give you access. To analyze the results, we recommend a machine running Ubuntu 24.04 LTS with a modern CPU, sufficient RAM (16GB minimum) and 100-200GB disk space.

A.2.1 Security, privacy, and ethical concerns

The experiments are self-contained and isolated from any public routing infrastructure. The amount of traffic generated is normal, so our experiments also do not negatively affect the cloud provider.

A.2.2 How to access

You can access the artifacts via the following stable link to Zenodo: https://doi.org/10.5281/zenodo.15612433.

A.2.3 Hardware dependencies

Our experimental setup uses DigitalOcean VMs to eliminate hardware dependencies and ensure reproducibility. For local replication, we recommend matching the following VM specifications:

- **3x CPU-Optimized / 8 GB / 4 vCPUs:** VMs hosting the three ASes not directly part of the BGP Vortex: *bystander-as*, *upstream-as*, and *downstream-as*. Each of these ASes runs the FRR router.
- **4x Memory-Optimized / 16 GB / 2 vCPUs:** VMs hosting the four ASes of the BGP Vortex: *attacker-as*, and its three providers *as-1*, *as-2*, and *as-3*. Each of these ASes runs the BIRD router.
- **1x Storage-Optimized / 16 GB / 2 vCPUs:** VM used for orchestration and data gathering.

The machine on which you intend to evaluate the results of the experiments should be equipped with a modern CPU, sufficient RAM (16GB minimum) and 100-200GB disk space.

A.2.4 Software dependencies

Our experiments run on Ubuntu 24.04 LTS and require BIRD and FRR software routers, iperf3, and tcpdump. Data analysis uses Tshark and Jupyter-Lab, with the following Python packages: `matplotlib`, `mrtparse`, `pandas`, `numpy`, and `seaborn`.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

Step 1: Set up local environment Extract the artifact to the machine on which you also intend to perform the analysis of the experiment data. All subsequent directory references are relative to the artifact root unless specified otherwise.

Step 1: Configure VMs. First, provision eight virtual machines with the aforementioned specifications. Ensure co-location within a single data center and, when utilizing cloud infrastructure, verify virtual private cloud (VPC) support. Establish SSH connectivity to all instances and confirm inter-VM communication via private IP addresses.

Afterwards, copy the public and private IP addresses for each VM into `artifact/section_5/generation/ips_map.json` as detailed in the Section 5 README.

Lastly, install FRR, BIRD, and tcpdump on all VMs. On the machine on which you intend to analyze the results, also install tshark and the following Python packages: matplotlib, mrtparse, pandas, numpy, and seaborn.

Step 2: Generate Configuration Files and Scripts. Execute the `artifact/section_5/generation/generate_output.py` script from within the generation folder on your local machine. This populates the output folder with configuration files specific for your deployment.

Step 3: Deploy Configuration to VMs. Execute the `artifact/section_5/generation/push_configs.py` script from within the generation folder on your local machine to deploy the generated configuration files to the VMs.

Step 4: Establish Network Topology. Configure GRE tunnels connecting the upstream-AS, bystander-AS, and downstream-AS. To that end, SSH into the fetcher VM and execute `bash set_up_gretaptunnels.sh`. Subsequently, SSH into the upstream-AS and bind the IP address `192.168.10.1` to its loopback interface. Repeat this step for the downstream-AS with IP `192.168.11.1`.

A.3.2 Basic Test

SSH into the fetcher VM, navigate into the `utils` folder and execute `bash trigger_sending_K_mins.sh 0 0 2 200 8000`. The expected output is:

```
Initializing experiment with 0 prefixes,
200% CPU,and 8000MB memory for 0 minutes.
Setting resource constraints on bystander-as
DEBUG: killing existing instance of
assignResources.sh on the bystander-as
DEBUG: Executing command:
ssh bystander-as "nohup bash ~/assignResources.sh
${cpuCores} ${cpuShare} ${memory} ${swapMemory}
>/dev/null 2>&1 &"
Setting resource constraints on downstream-as
DEBUG: killing existing instance of
assignResources.sh on the bystander-as
DEBUG: Executing command
ssh bystander-as "nohup bash ~/assignResources.sh
${cpuCores} ${cpuShare} ${memory} ${swapMemory}
>/dev/null 2>&1 &"
Setting resource constraints on upstream-as
DEBUG: killing existing instance of
assignResources.sh on the bystander-as
DEBUG: Executing command:
ssh bystander-as "nohup bash ~/assignResources.sh
${cpuCores} ${cpuShare} ${memory} ${swapMemory}
>/dev/null 2>&1 &"
bird_attacker.conf
Started experiment!
Experiment finished, stopping!
DEBUG: killing existing instance of
assignResources.sh on the bystander-as
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): BGP Vortices are widespread on today's Internet (Section 4). We demonstrate this by identifying BGP Vortices in the CAIDA AS-relationship based on known vulnerable ASes.

(C2): BGP Routers are susceptible to BGP Vortices (Section 5.2). We demonstrate the vulnerability of widely deployed, industry-standard router implementations.

(C3): BGP Vortices delay network convergence (Section 5.3). We create a BGP Vortex in our testbed and measure route update propagation delay, demonstrating that critical routing information dissemination is delayed, thus prolonging convergence.

(C4): BGP Vortices overload routers (Section 5.4). We show that our experimental setup, routers fail to keep up with the rate at which route updates would be sent, as evidenced by TCP receive window sizes reaching zero.

(C5): BGP Vortices cause data plane outages (Section 5.5). We create a BGP Vortex in our testbed, measure data plane connectivity using iperf3, simulate route changes, and quantify data plane outage duration.

A.4.2 Experiments

In this section, we describe how to replicate our experiments and how to analyze the results. We encourage the reader to consult the READMEs provided inside the artifact, which include more detailed step-by-step instructions.

(E1): BGP Vortices are widespread on today's Internet (Section 4). [5 human-minutes + 5 compute-seconds]:
Preparation: No preparation required, assuming prior instructions were followed and dependencies installed.
Execution: Execute the following commands from

within the `artifact/section_4/evaluation/` directory:

```
# Count analysis
python run.py -t data/20240301.as-rel.txt
-a data/sico_ases.txt count

# Downstream analysis:
python run.py -t data/20240301.as-rel.txt
-a data/sico_ases.txt downstream
```

Results: The count analysis computes the number of BGP Vortices (gadgets) each of the 21 vulnerable ASes is included in. The downstream analysis computes the number of updates seen by every AS in the customer cone of the 21 vulnerable ASes. Both results together correspond to insights from section 4 of our paper, and thus prove claim C1.

(E2): BGP Routers are susceptible to BGP Vortices (Section 5.2).: We do not provide an explicit experiment, since the subsequent experiments generate BGP Vortices and thus also prove that BGP routers are susceptible.

(E3): BGP Vortices delay network convergence (Section 5.3). [15 human-minutes + > 1 compute-day]

Preparation: No preparation required, assuming prior instructions were followed and dependencies installed.

Execution: To initiate the experiment, SSH into the fetcher VM and execute `bash cp_experiment.sh 2 tcpdump`. The experiment requires substantial time; we recommend execution within a `tmux` instance (on the fetcher VM) to maintain session persistence. Record the experiment identifier as it is needed for subsequent evaluation.

Results: To analyze the experiment, first retrieve data from the fetcher VM by executing the following command from the `artifact/section_5/evaluation/` directory:

```
scp -r root@<fetcher VM IP>:
~/data/dataplane_experiment/<exp. id>
./experiment_data/<ex. id>
```

Each experiment generates a separate file for 100% and 200% CPU share (1 and 2 CPU cores) and for 0, 50, 100, 600, 1000, 3000, and 5000 attacker prefixes. Processing each file sequentially is highly inefficient, so instead we provide a script that should be executed in parallel for each combination of CPU share and attacker prefix count (14 combinations total). Execute the following commands for each combination from the `artifact/output/extraction/` directory:

```
python3 parallel_tcpdump_extraction.py
<exp. id> bystander-as
<cpu share> <num attack pfx> arrival_times

python3 parallel_tcpdump_extraction.py
<exp. id> upstream-as
<cpu share> <num attack pfx> arrival_times
```

To analyze data and generate plots comparable to those presented in the paper, launch a jupyter-lab instance in the `artifact/section_5/evaluation` directory and execute all cells in `section_5_3.ipynb`. Ensure the experiment identifier is correctly configured in the designated notebook cell.

(E4): BGP Vortices overload routers (Section 5.4). [15 human-minutes + several compute-hours]

Preparation: No preparation required, assuming prior instructions were followed and dependencies installed.

Execution: No execution required; this experiment reuses results from experiment E3.

Results: Each experiment contains data for 100% and 200% CPU share (1 and 2 CPU cores) and for 0, 50, 100, 600, 1000, 3000, and 5000 attacker prefixes. Given the computational complexity of processing each file, we provide a script for parallel execution across all combinations of CPU share and attacker prefix counts (14 combinations total). Execute the following command for each combination from the `artifact/output/extraction/` directory:

```
python3 parallel_tcpdump_extraction.py
<exp. id> bystander-as <cpu share>
<num attack pfx> zerowindows
```

To analyze the data and generate plots corresponding to those in the paper, launch a jupyter-lab instance in the `artifact/evaluation` folder and execute all cells in `section_5_4.ipynb`. Verify the experiment identifier is correctly configured in the designated notebook cell.

(E5): BGP Vortices cause data plane outages (Section 5.5). [15 human-minutes + >1 compute-day]

Preparation: No preparation required, assuming prior instructions were followed and dependencies installed.

Execution: To initiate the experiment, SSH into the fetcher VM and execute `bash dp_experiment.sh 2`. The experiment requires substantial time; we recommend execution within a `tmux` instance (on the fetcher VM) to maintain session persistence. Record the experiment identifier as it is needed for subsequent evaluation.

Results: To analyze the experiment, you first have to retrieve the data from the fetcher VM. This is achieved by executing the following command from within the `artifact/section_5/evaluation/` directory:

```
scp -r root@<fetcher VM IP>:
~/data/dataplane_experiment/<exp. id>
./experiment_data/<ex. id>
```

Each experiment holds data for 100% and 200% CPU share (1 and 2 CPU cores), and for 0, 50, 100, 600, 1000, 3000, and 5000 attacker prefixes. Since process-

ing each file takes a long time, we provide a script which we encourage to execute in parallel for each combination of CPU share and number of attack prefixes (14 times in total). Do so by executing the following line for each such combination, from within the `artifact/output/extraction/` directory:
To analyze data and generate plots comparable to those presented in the paper, launch a jupyter-lab instance in the `artifact/section_5/evaluation` directory and execute all cells in `section_5_5.ipynb`. Ensure the experiment identifier is correctly configured in the designated notebook cell.

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/usenixsec2025/.