Spoki: Unveiling a New Wave of Scanners through a Reactive Network Telescope

Raphael Hiesgen, *HAW Hamburg;* Marcin Nawrocki, *Freie Universität Berlin;* Alistair King, *Kentik;* Alberto Dainotti, *CAIDA, UC San Diego and Georgia Institute of Technology;* Thomas C. Schmidt, *HAW Hamburg;* Matthias Wählisch, *Freie Universität Berlin*

https://www.usenix.org/conference/usenixsecurity22/presentation/hiesgen

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

Open access to the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

A Artifact Appendix

A.1 Abstract

Spoki is a real-time reactive network telescope. It is written in C++ and based on the actor model to achieve high scalability. The artifacts also include Python tools to analyze Spoki log files, identify downloaders distributed by attackers, and fetch files referenced by the downloaders.

We used Spoki to collect the data for our paper over the course of three months. The artifact contains the source code. It can be used to collect the same information (given a suitable setup) and get started with the evaluation.

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

Compilation: C++-17 compiler, Python 3.

Run-time environment Linux. Capabilities to capture network traffic (telescope, root access).

Hardware Depends on your traffic volume. Processing traffic from a /24 should work with 4 cores of common server hardware.

Output Spoki writes observed events to log files, which can be analyzed by our Python tools.

Experiments We collected data over three months. The data sets we used contain sensitive data and are thus not available.

Publicly available? Yes, see below.

Code licenses MIT License

A.3 Description

A.3.1 How to access

All artifacts and future results are available via `https://spoki.secnow.net/`. We also archived the artifact on Zenodo: `https://zenodo.org/record/5702603`.

A.3.2 Software dependencies

Spoki runs on Linux and uses the following open-source software. The repository contains a script to build them.

- The C++ Actor Framework (CAF)
- Scamper
- libtrace

The Python tools depend on Kafka. Python-related dependencies can be installed via `pip` (see below).

A.4 Installation

Spoki The source code is located in the `spoki/` folder. On Ubuntu 20.04, you first need to install the following dependencies:

```
$ sudo apt install gcc g++ cmake git curl make
    libtool-bin automake libpcap0.8-dev
    libbison-dev flex
```

Now you can build the required libraries via a script in the repository. It installs the libraries into a local folder.

```
$ ./setup.sh
```

Finally, you need to configure and build Spoki:

```
$ ./configure
$ make -C build
```

The Spoki binary will be located at `./build/tools/spoki/`.

Evaluation Tools to download the malware linked in payloads are located in `evaluation`. First, setup a local virtual environment. Inside the environment you can setup the tools via the makefile (run it twice the first time):

```
$ make update
$ make update
```

This will link the following tools into the virtual environment:

assemble Reads Spoki logs and assembles events.

filter Identifies events with downloaders.

clean Extracts and clean links from events.

download Follows links and downloads executables.

Additional tools to analyze port statistics, contact types, query malware hashes in VirusTotal, and annotate data are located in the same project. Please check the `README.md` file for details.

A.5 Experiment workflow

Spoki can be run directly from the command line and accepts configuration via a `caf-application.conf` file. Please check the `README` of Spoki for the configuration details. It is necessary to configure the data source (e.g., the interface to read packets from), a folder for the logs, and a tag for your datasource. Spoki writes two types of logs: event logs that contain information on observed events alongside Spoki's probe reply and scamper logs that list the probe confirmations received from scamper.

The malware processing tools require a running Kafka instance for communication. They further accept CLI options to configure the Kafka topic they use between them. The `assemble` further accepts the location of the Spoki tools and a date and hour to start processing. These tools build a processing pipeline. In the final step, Spoki logs all observed URLs and the data it downloads alongside some meta information.

A.6 Evaluation and expected results

We cannot provide our data sets because they include personally identifiable information such as IP addresses. We provide Spoki's code to allow others to repeat our experiments.