USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

TORCHLIGHT: Shedding LIGHT on Real-World Attacks on Cloudless IoT Devices Concealed within the Tor Network

Yumingzhi Pan and Zhen Ling, *Southeast University;* Yue Zhang, *Drexel University;*
Hongze Wang, Guangchi Liu, and Junzhou Luo, *Southeast University;*
Xinwen Fu, *University of Massachusetts Lowell*

https://www.usenix.org/conference/usenixsecurity25/presentation/pan-yumingzhi

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

USENIX Security '25 Artifact Appendix: TORCHLIGHT: Shedding LIGHT on Real-World Attacks on Cloudless IoT Devices Concealed within the Tor Network

Yumingzhi Pan[†], Zhen Ling[†,*], Yue Zhang[‡], Hongze Wang[†], Guangchi Liu[†], Junzhou Luo[†], Xinwen Fu[§]
[†]Southeast University, Email: {pymz, zhenling, wanghongze, gc-liu, jluo}@seu.edu.cn
[‡]Drexel University, Email: yz899@drexel.edu
[§]University of Massachusetts Lowell, Email: xinwen_fu@uml.edu

A Artifact Appendix

A.1 Abstract

This artifact is the public source release of TORCHLIGHT which is designed to detect both known and unknown threats targeting cloudless IoT devices by analyzing Tor traffic. TORCHLIGHT consists of three components:

- Tor Exit Traffic Collector: Real-time capture and storage of external traffic on a resource-constrained VPS, with offline filtering on a campus NAS. *Artifacts released:* node deployment guidelines, offline traffic filtering code.
- Deployment Planner: Optimized VPS resource allocation for effective traffic monitoring. *Artifacts released:* Tor weighted bandwidth algorithm, Tor exit deployment strategy code.
- LLM-based IoT Traffic Analyzer: IoT traffic identification and detection of four attack types using a Large Language Model. *Artifacts released:* IoT traffic identification and attack detection code, sample data.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Executing this artifact does not involve security, privacy, or ethical concerns. For ethical reasons, we will not make the Tor exit traffic we collected publicly available. However, in order to help evaluators in testing the IoT traffic identification and attack detection functionalities, we have released a portion of sample data in accordance with the Menlo Report:

1. This data set excludes any information that could potentially reveal personal or identifiable details—including the Host field, cookies, and software version fingerprints.
2. For attack detection sample data, these malicious requests solely exploit N-day vulnerabilities that have already been patched.

Please refer to these samples in the `./sample_data` directory in the artifact.

Furthermore, to facilitate researchers in validating the functionality of the data preprocessing scripts, we have downloaded several test PCAP files from publicly accessible websites and repositories (including SampleCaptures and PCAPS). Please refer to the `./test_pcaps` folder to access the test PCAP files.

A.2.2 How to access

The code, Tor exits deployment guidelines, and test data are available on Zenodo: https://zenodo.org/records/14742809.

A.2.3 Hardware dependencies

GPU. For our LLM-based IoT traffic analyzer, GPUs with at least 48GB of memory (such as NVIDIA A40, A6000, and A100) are required to meet the memory demands of the quantized Llama 70B model.

A.2.4 Software dependencies

Python Package. All necessary Python packages for our LLM-based IoT traffic analyzer are listed in the corresponding `requirements_<dir>.txt` files.

- **IoT Traffic Identification**
  - **Python version**: Python 3.9.17
  - Install dependencies in the `./iot_identification` directory
  - Install and extract the exllama repository. After extraction:

[*]Corresponding author: Prof. Zhen Ling of Southeast University, China.

    - Copy these Python files to `iot_identification` folder: `model.py`, `generator.py`, `cuda_ext.py`, `lora.py`, and `tokenizer.py`
    - Copy the entire `exllama_ext` directory

- **Attack Detection**
  - **Python version**: Python 3.9.19
  - Install dependencies in the `./attack_detection` directory
- **Preprocessing**
  - Install dependencies in the `./preprocessing` directory

Environment Recommendation Due to version discrepancies in Python (3.9.17 vs 3.9.19) and potential dependency conflicts between components, we strongly recommend using separate virtual environments through Conda environments or Python virtualenv.

Open-Source LLM Models. For IoT Traffic Identification, it is necessary to download a quantized Llama 2 70B model(released at `https://huggingface.co/TheBloke/Llama-2-70B-Chat-GPTQ`). For Attack Detection, a quantized Llama 3.1 70B model is required for download (released at `https://huggingface.co/hugging-quants/Meta-Llama-3.1-70B-Instruct-AWQ-INT4`).

A.2.5 Benchmarks

Tor Exit Sample Data. Due to the sensitivity of Tor exit traffic, the raw data we collected will not be made publicly available. However, in order to help evaluators in testing the IoT traffic identification and attack detection functionalities, we have released a portion of sample data in accordance with the Menlo Report. Please refer to these samples in the `./sample_data` directory in the artifact.

Test Pcaps. To facilitate researchers in validating the functionality of the data preprocessing scripts, we have downloaded several test PCAP files from publicly accessible websites and repositories (including SampleCaptures and PCAPS). Please refer to the `./test_pcaps` folder to access the test PCAP files.

A.3 Set-up

A.3.1 Installation

The installation steps are explained in detail in the `README.md` file. Specifically,

Tor Exit Traffic Collector. For detailed instructions on deploying Tor exits and collecting Tor external traffic data, please refer to `deployment_guideline.md` in `./deployment` directory.

Deployment Planner. Download the latest Tor consensus file. One optional source for downloading is `https://torstatus.rueckgr.at/`, which converts the consensus file into a CSV format, making it easier to process.

LLM-based IoT Traffic Analyzer. Install the dependencies using the `requirements_<dir>.txt` files in the `./iot_identification` and `./attack_detection` directories.

A.3.2 Basic Test

The execution steps are explained in detail in the `README.md` file.

Tor Exit Traffic Collector. Users can run Suricata, Tor, lsyncd, iptables, and ipset to test if the software components are functioning fine.

Deployment Planner. Users can execute `calculate_weights.py` to verify the correct download of the consensus file.

LLM-based IoT Traffic Analyzer. Users can run the following commands to see if the Python packages have been installed successfully.

```
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer,
↪ AwqConfig
import pandas as pd
from model import ExLlama, ExLlamaCache, ExLlamaConfig
from tokenizer import ExLlamaTokenizer
from generator import ExLlamaGenerator
import httplib2
from googleapiclient.discovery import build
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Tor Exit Traffic Collector captures and storages of external traffic in real-time on a resource-constrained VPS. (§6.2 "Traffic Distribution")

(C2): Deployment Planner optimizes VPS resource allocation for effective traffic monitoring. (§6.2 "Traffic Distribution", §6.2 "Effectiveness" and Figure 4)

(C3): LLM-based IoT Traffic Analyzer leverages a LLM to determine if the response data originates from IoT devices, thereby identifying IoT traffic. (§6.2 "Accuracy", §6.3 and Table 3, 4)

(C3): LLM-based IoT Traffic Analyzer prompts the LLM to detect four types of attacks. (Appendix A.2 "Performance", §6.4 and Table 5, 7)

A.4.2 Experiments

Please refer to the `README.md` file in the repository for detailed steps of our experiments.

(E1): *[Tor Exit Traffic Collector] [30 human-minute + 10 compute-minute + 2GB disk]:*

Preparation & Execution: Please refer to the detailed instructions on deploying Tor exits and collecting Tor external traffic in `deployment_guideline.md` in `./deployment` directory. For offline filtering, please refer to "Attack Detection Preprocessing" in the `README.md` file.

Results: Tor Exit Traffic Collector shoud be able to collect Tor external traffic and preprocess the traffic into appropriate formats for LLMs.

(E2): *[Deployment Planner] [10 human-minute + 2 compute-minute]:*

Preparation: Download the latest Tor consensus file as detailed in A.3.1.

Execution: To compute the bandwidths E and D, and determines weights Wee and Wed based on three distinct network conditions, please execute `deployment/calculate_weights.py`. Then, execute `deployment/deploy_strategy.py` to formulate the deployment plan (There are example node options included in the script).

Results: Deployment Planner shoud be able to generate a deployment plan that optimizes VPS resource allocation for effective traffic monitoring.

(E3): *[IoT Traffic Identification] [2 human-hour + 10 compute-minute + 50GB disk]:*

Preparation: (i) Activate the IoT Traffic Identification virtual environment as specified in the A.2.4 section. (ii) Obtain an API key and search engine ID for Google Custom Search Engine(https://programmablesearchengine.google.com). (iii) Utilize the data provided in the `iot_identification_sample_data.csv` file in `./sample_data` directory.

Execution: After entering `iot_identification` directory, run the following four sequential steps:

- **Step I**: It preliminarily identify IoT names within the response data. (`01a`, `01b`)

```
python 01a_llama_ner_initial.py
python 01b_extract_result_from_initial.py
```

- **Step II**: It re-verifies the IoT entities to address potential hallucinations. (`02a`, `02b`)

```
python 02a_llama_ner_second_round.py
python 02b_extract_result_from_second_round.py
```

- **Step III**: It leverages a search-engine-based retriever to complete potentially missing vendor and type names based on the identified model names. (`03a`, `03b`, `03c`)

```
python 03a_retrieve_google_search.py
python 03b_llama_ner_google_search.py
python 03c_extract_result_from_google_search_llamaed.py
```

- **Step IV**: It ensures that the traffic truly originates from IoT devices. (`04a`, `04b`)

```
python 04a_llama_ner_device_traffic_check.py
python 04b_extract_result_from_device_traffic_check.py
```

Results: Upon successful execution of Step IV, the final `<OUTPUT_FILE>` will contain LLM-identified IoT traffic entries, along with their corresponding predictions for vendor (`pred_vendor`), device type (`pred_type`), and/or model (`pred_model`).

(E4): *[IoT Attack Detection] [2 human-hour + 10 compute-minute + 50GB disk]:*

Preparation: (i) Activate the IoT Traffic Identification virtual environment as specified in the A.2.4 section. (ii)Utilize the data provided in the `attack_detection_sample_data.csv` file in `./sample_data` directory.

Execution:

- **Command injection detection.**

```
python llama_command_injection.py
python extract_llama_command_injection.py
```

- **FTP anomaly detection.**

```
python llama_ftp_anomaly.py
python extract_llama_ftp_anomaly.py
```

- **Path traversal detection.**

```
python llama_path_traversal.py
python extract_llama_path_traversal.py
```

- **Information disclosure detection.**

```
python llama_information_disclosure.py
python extract_llama_information_disclosure.py
```

Results: (i) The output file from `llama_<attack_type>.py` contains the LLM's complete response to this specific attack type. This includes whether the LLM identified the input as an attack and its subsequent analysis, recorded in the `<attack_type>_pred_line` column. (ii) The `extract_llama_<attack_type>.py` script extracts the LLM's core answer (`"yes"` or `"no"`) regarding the attack and stores it in the `<attack_type>_pred_result` column.

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/.