USENIX WOOT'25 Artifact Appendix
DeepRed: A Deep Learning–Powered Command and Control Framework for Multi-Stage Red Teaming Against ML-based Network Intrusion Detection Systems

Mehrdad Hajizadeh[†], Pegah Golchin[†], Ehsan Nowroozi[*], Maria Rigaki[‡], Veronica Valeros[‡], Sebastian García[‡], Mauro Conti[§], Thomas Bauschert[†]
[†]Technische Universität Chemnitz, Germany
[*]Centre for Sustainable Cyber Security (CS2), University of Greenwich, UK
[‡]Czech Technical University in Prague, Prague
[§]University of Padua, Italy
mehrdad.hajizadeh@etit.tu-chemnitz.de

14.06.2025

A Artifact Appendix

A.1 Abstract

This document describes the artifacts accompanying the USENIX Security '25 publication DeepRed: A Deep Learning-Powered Multi-stage Red Teaming Operations to Reveal Vulnerabilities of ML-based Network Anomaly Detection.

The artifact includes a novel ML-based Network Intrusion Detection System (ML-NIDS) benchmarking dataset, detailed in Sections 4.6 and Appendix A.1. The dataset is generated upon 30 diverse red team exercises covering various attack objectives mapped to MITRE ATT&CK stages, with concurrently generated benign traffic (summarized in Table 8). We also provide the code for the DeepRed Command-and-Control (C2) framework, which demonstrates successful bot-to-C2 communication and execution of RCE for system discovery and data exfiltration.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The DeepRed artifact involves a WebSocket-based communication framework between a Bot and a C2 server that requires binding to local interfaces or opening specific ports. While the artifact does not contain destructive payloads, it simulates malicious behavior, such as remote command execution and data exfiltration (from bot to C2), for evaluation purposes. Using tools like Wireshark traffic analysis is encouraged to inspect and validate the communication between components. Evaluators should avoid executing the artifact with sensitive data on production systems or machines (recommending isolated tests).

A.2.2 How to access

The artifacts are uploaded on GitHub repository[1]. This repository contains all the necessary files and instructions to run the DeepRed C2 framework. Multiple bots can connect concurrently to the C2 server, and network traffic can be captured during their execution. In addition, the TUC-RedTeam30 dataset—which includes 30 diverse red teaming scenarios with both benign and malicious activity mapped to the MITRE ATT&CK framework—is available on Zenodo DOI `10.5281/zenodo.15668685`[2].

A.2.3 Hardware dependencies

- OS: Linux (Ubuntu +20.04)
- 2GB of disk space

A.2.4 Software dependencies

To execute the framework, Python 3.11 or later is required, along with core dependencies including Scapy and tcpdump. For a comprehensive list of required Python packages and their versions, please consult the `requirements.txt` file.

[1] https://github.com/Mehrdad-hajizadeh/DeepRed-C2/tree/v1.0
[2] https://zenodo.org/records/15668685

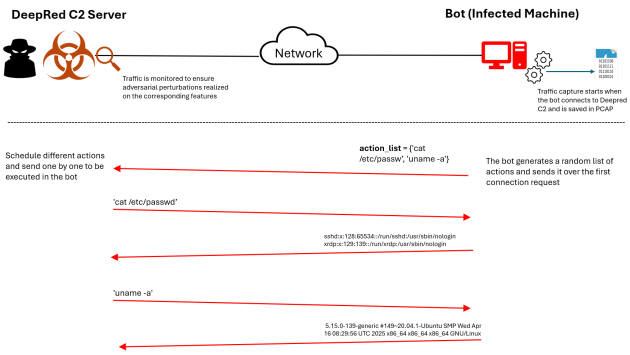

Figure 1: A sample DeepRed C2 operation

A.2.5 Benchmarks

The TUC-RedTeam30 dataset included in this artifact was used to evaluate several ML-based NIDS models, including SSCL-IDS, FlowTransformer, CNN, KNN, and RF. These models were tested on the dataset as part of the experiments reported in the paper.

A.3 Set-up

The environment was set up using Ubuntu 20.04 as the operating system.

A.3.1 Installation

The DeepRed artifact is implemented in `Python3.11` and requires setting up a virtual environment with dependencies listed in requirements.txt. After creating and activating the environment, install the required packages using `pip install -r requirements.txt`.

A.3.2 Basic Test

To verify the artifact operates as intended, a basic test can be performed by running the two main components: the DeepRed C2 server and the bot client. The system is designed to run on `Ubuntu 20.04` with `Python 3.11` in a virtual environment. After setting up the environment (detailed in the README), the user launches `c2-server.py`, which interactively prompts for network configuration. If no custom input is provided, the server defaults to 127.0.0.1 and port 5000. It also allows enabling or disabling a runtime termination condition checker. In a separate terminal, the user starts `bot.py`, which interactively requests the server IP and port, whether adversarial perturbation is required, the number of automated flow iterations, and whether to capture traffic into a PCAP file. If adversarial perturbation is enabled, the bot either loads a `adversarial_perturbation.yaml` configuration (if present) or allows the user to input feature names and their values manually (e.g., src2dst_max_ps: [150, 300]). For traffic capture, the bot auto-detects the system's available interfaces and selects the one matching the server's IP range. Invalid inputs (e.g., unavailable interfaces or used ports) are rejected with feedback prompts. This interactive approach enables flexible configuration while ensuring valid inputs, and is sufficient to test core functionality and produce observable outputs including log files and optional PCAP traces, see Figure 1.

A.4 Evaluation workflow

A.4.1 Major Claims

- **(C1):** The DeepRed C2 framework successfully emulates bot-to-C2 communication, in which a randomly generated action list representing malicious activities is executed during the session. When configured accordingly, DeepRed can apply adversarial perturbations to selected network flow features by injecting crafted packets or padding messages with random strings to increase the byte size—enabling the simulation of evasive behavior against ML-based NIDS.

- **(C2):** A network traffic dataset, named TUC-RedTeam30, was generated during a series of red team attack scenarios, alongside concurrently generated benign traffic. The NFStream library was used to extract flow-level features from the raw PCAP files. These flows were then labeled and enriched with metadata to meet the requirements detailed in Appendix A of the main paper.

A.4.2 Experiments

- **(E1):** In addition to the discussion in Section A.3.2, we provide complete instructions for users to perform a sample experiment on the target benchmark machine. Further details are included in the README file, accompanied by a GIF demonstrating a sample interactive implementation.
  How to: Run the communication between the bot and the DeepRed C2 server to observe automated remote command execution and data exfiltration.
  Execution: First, launch `c2-server.py`, then run `bot.py`, ensuring both are correctly configured interactively.
  Output: While the C2 server is running, all logs are stored in the `deepred-c2/log` directory. If data capture is enabled during `bot.py` execution, the corresponding PCAP files will be saved in `deepred-c2/pcap`. Additionally, when data exfiltration is selected as part of the bot's action list, sample files are randomly chosen from `deepred-c2/sample_file_to_exfil` on the bot side and transferred to the DeepRed C2 server under `deepred-c2/exfiltrated_data`.

- **(E2):** The dataset is available in the GitHub repository under the `dataset` directory and has also been published on Zenodo with the DOI: 10.5281/zenodo.15662906. It includes CSV files containing extracted flow features, along with a selection of corresponding PCAP files. The CSV file is labeled and include metadata that allows users to identify and distinguish between the 30 individual scenarios, as detailed in Appendix A of the main paper.

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/.