# DongTing: A Large-scale Dataset for Anomaly Detection of the Linux Kernel

This file is DongTing Dataset Development Documentation, which records the entire process of dataset development and benchmark establishment. It consists of four parts: operation requirements, database, program structure and working steps, model training and evaluation (including training and evaluation). The following describes the work of each part in detail.

## 1 Require

### 1.1 Environmental requirements

The operating environment of the virtual machine is built on the Esxi platform. The collection tool chain is developed in Python, and MongoDB is used to store data. Multiple libraries are used in the development process. The library names and versions are shown in the table.

```
# Platform
# Esxi         6.7

# Environment
# Python       3.9.X
# MongoDB      4.4

# Library
# paramiko     2.10.3
# tqdm         4.63.1
# pyvim        3.0.2
# pysphere     0.1.7
# pysphere3    0.1.8
# pyvmomi      7.0.3
```

### 1.2 Environment installation

#### (1) Virtualized server

- Please install VMexsi6.7 on the experimental platform, configure the VM virtual machine according to the physical server, and install Ubuntu21.04 in the virtual machine.
- Replace the Linux kernel of the corresponding version of the POCs according to the collected POCs target environment.

- After installing ESXI, ensure that the management platform can log in normally through HTTPS.
- Confirm that the strace tool in the virtual machine can work normally.
- In the virtual machine with the replaced kernel, enable SSH login, and set the management information according to the content of the labserverinfo_input.txt template.

### (2) Collection server

- Make sure that the Ptyhon environment is 3.9.x or above, and python3.10 is recommended.
- Install the MongoDB database, 4.4 is recommended.
- Install the required modules using pip.

```
pip install paramiko==2.10.3
pip install tqdm==4.63.1
pip install pyvim==3.0.2
pip install pysphere==0.1.7
pip install pysphere3==0.1.8
pip install pyvmomi==7.0.3
```

- Import backup data.
- Run the main program according to the command parameters.

### (3) POC and test suite

- POC is collected through the syzbot publishing platform.
- The test suites are LTP (Linux Testing Project), Kselftests (Linux Kernel Selftests), Open Posix Tests and Glibc Testsuite, and they are compiled in Ubuntu21.04 after downloading.

## 2 Database

### (1) Data import

The data of the database in the source code is backed up from MongoDB. When using it, please install the MongoDB database of version 4.4 and import it.

```
mongorestore -h IP:27017 -d syzbot_DB $cp_local_src_dir >/dev/null
```

### (2) Collection description in the database

- Collection kernel_convert_baseline: benchmark generation results, where kcb_master_line_ver is the kernel mainline version. kcb_seq_lables are data labels, which are divided into two categories: attack and normal. kcb_seq_class is divided into data subsets.
- Collection kernel_syscall_x64tbl: According to the syscall_64.tbl (kernel version 5.17) file, the system call list is converted into a database for calling.
- Collection kernel_syscall_normal_strace: Sequence of system calls extracted from the LOG collected by the kernel test suite. Where kns_normal_seq_list is the system call sequence, separated by "|". kns_normal_mlseq_list is the encoded data of the system call number, separated by "|".
- Collection kernel_syscallhook_bugpoc_trace_sum: The sequence of system calls extracted from the LOG of the POC attack kernel. Among them, kshs_poclog_name is the bug name. kshs_bugpoc_syscall_list is the system call sequence, separated by "|". kshs_bugpoc_syscall_mlcode is the encoded data of the system call number, separated by "|". For reference, the system call sequence in kshs_bugpoc_syscall_list refers to the result of converting the set kernel_syscall_x64tbl.
- Collection kernel_trace_dstserver: virtualization platform and virtual machine management information, serving the distributed collection framework. When using, add server and virtual machine information according to the fields provided in the labserverinfo_input.txt template file in the analysis directory in the source code (please use @ to separate multiple server data), gather_syzfixpoc_trace.py will be imported during initialization. e.g.:

```
@
name: aimlab-osslab-03-6     # Virtual Machine tag, e.g.:aimlab-osslab-03-6
ip:                          # Virtual Machine IP
username:                    # Virtual Machine general user username
userpwd:                     # Virtual Machine general user password
rootpwd:                     # Virtual Machine root password
ssh_port: 22                 # SSH port
kernel_ver: 5.1              # Linux Kernel branch version, including next-level
branch version
tool_name: strace            # Collection tool name
tool_method: rtools          # Tools work form
lot: 30                      # Collection lot
esxi_vm_name: lab-210        # Virtual machine name in esxi server
esxi_pvip:                   # Esxi Server IP
esxi_pvusername: root        # ESXi Server root name
esxi_pvpwd:                  # ESXi Server root password
esxi_pvport: 443             # ESXi Server manage service port
```

## 3 Program structure and working steps

### 3.1 Program structure

The source code contains three folders, Source Code, Documents and DB, where Documents stores the development documents, DB stores the sample data, and Source Code is the source code, and its file structure and the description of each file are as follows.

```
Sourcd Code                       # DongTing source code directory
│  argparses.py                   # Command parameter execution file
│  main.py                        # Main file
│  readme.md                      # Source code documentation
├─analysis                        # Log Collection and Analysis Program Directory
│  │  analy_poclogbug_strace.py   # Extract system calls from abnormal LOG and 
generate system call sequence
│  │  analy_poclognor_strace.py   # Extract system calls from normal LOG, generate 
system call sequence
│  │  convert_syscall_toml_select.py  # Automatic partitioning of training, 
validation, and test sets, labeling procedures
│  │  gather_normal_strace.py     # Normal program execution log collector
│  │  gather_switch_vmkernel.py   # VM monitoring and hypervisor
│  │  gather_syzfixpoc_trace.py   # DTLACM method main program
│  │  sum_write_logtosyscall_db.py    # Extract analysis process information from the 
result of analyzing abnormal LOG
│  │  sun_diff_db-sumfb.py        # Sequence length statistical analysis program
│  ├─normal_log                   # LOG generated by running the test suite
│  ├─poc_log                      # LOG generated after running POC
│  │      labserverinfo_input.txt     # Import the management information template of 
the VM
│  │      labserverinfo_input_temp.txt  # VM Management Information Template Field 
Descriptions
│  ├─poc_out_nor-step3            # Normal system call sequence storage directory
│  ├─poc_out_syz-step3            # Abnormal system call sequence storage 
directory
│  ├─srcdata                      # The normal LOG storage directory to be 
analyzed
│  └─srcdataattk                  # The abnormal LOG storage directory to be 
analyzed
├─conn                            # Database related file directory
│      dbconn.py                  # Database connection file
│      dbcopy.sh                  # Database data import and export script
├─ml                              # Machine Learning Program Directory
│  │  Refer_to_instructions.txt   # Citation Notes
│  ├─scripts
│  │   run_trials.py              # Machine Learning Master File
│  │   submitter.py               # Responsible for generating slurm task 
submission files
│  └─src
│       adfa_preprocessing.py     # Adfa data preprocessing
│       data_processing.py        # Model input data processing
│       models.py                 # Model implementation
│       plaid_preprocessing.py    # PLAID data preprocessing
│       save_scores.py            # Model evaluation
│       training_utils.py         # Training tool
│      train_ensemble.py              # Model training
├─public                              # Public file directory
│  │  pub_fun_analysis.py             # Centrally define some commonly used functions
in the analysis process
│  │  pub_network_test.py             # VM connection test program
│  │  pub_syscall_x64tbl.py           # Extract the system calls from the TLB file and
store the system calls in the database tbl_db_list
│  └─pubfile
│         syscall_64.tbl              # Linux Kernel System Call List of X86_64
（5.17.0）
└─tools
      clean_couttarlog.py             # Directory Version Kernel VM Work Process File
Cleaner
      clean_ps_20m.py                 # Periodically scan the program when the POC is
executed
```

## 3.2 Work steps

### (1) Parameter Description

The main program is main.py, which uses parameters to control its work at runtime. The parameters are as follows:

```
main.py -command

# -i,init              # Initialize the database
# -g,gather            # Enable LOG collection
#   -add               # Add experimental server
#   -syz               # Collect the LOG of the POC in syz
#       -lot num       # Open the batch for collecting LOG, num can be multiple
integers, separated by spaces
#   -nor               # Collect LOG in suit
# -a,analysis          # Enable analysis of syscalls
#   -syz               # Analyze syz source data
#   -nor               # Analyze the normal data of strace
# -q,quit              # quit program
# -v,version           # Version information
```

Tips:

- Before running, the Esxi platform, database, and modules must be built as required.
- According to requirements, the collection and analysis programs can be run independently.

### (2) Steps

- Run main.py to initialize the various environments.
- Collect POC, compile test suite.

- Use Strace to collect POC and system call logs when the test suite is running.
- Analysis, statistics LOG extraction system call sequence.
- Dividing benchmarks and marking data

## 4 Model training and evaluation

#### (1) Machine learning environment

For the machine learning files in the ML directory in Section 3.1, we need to do some prep work:

- Build slurm and conda environments.
- Model training or evaluation via run_trials.py.

#### (2) Steps

- Follow the previous step to build a machine learning environment.
- Create a new data directory and put the DongTing and comparison dataset npz files into it.
- Modify the slurm partition on line 119 of submitter.py. Change "#!/bin/bash\n#SBATCH --partition=gpu\n" to the partition name of the new slurm.
- Modify the conda environment path on line 142 of submitter.py. Modify "source activate /home/***/uvm_env\n" to the path of the newly created conda environment.
- Model training is done via main() in run_trials.py.
- Model evaluation via save_scores() in run_trials.py.