Background

HorusEye is a framework that consists of a control plane and data plane elements implemented to outperform the state-of-the-art network intrusion detection model named kitsune; by utilizing the high throughput of a programmable switch, the control plane can offload a major portion of the control plane overhead.

HorusEye is a two-stage anomaly detection model, the first stage is an unsupervised learning model implemented on the data plane called Gulliver tunnel that can filter some suspicious traffic. The second stage is an unsupervised deep learning model implemented on the control plane called Magnifier, that will further analyze the reported suspicious traffic and generate a block list that will be appended to the data plane block list.

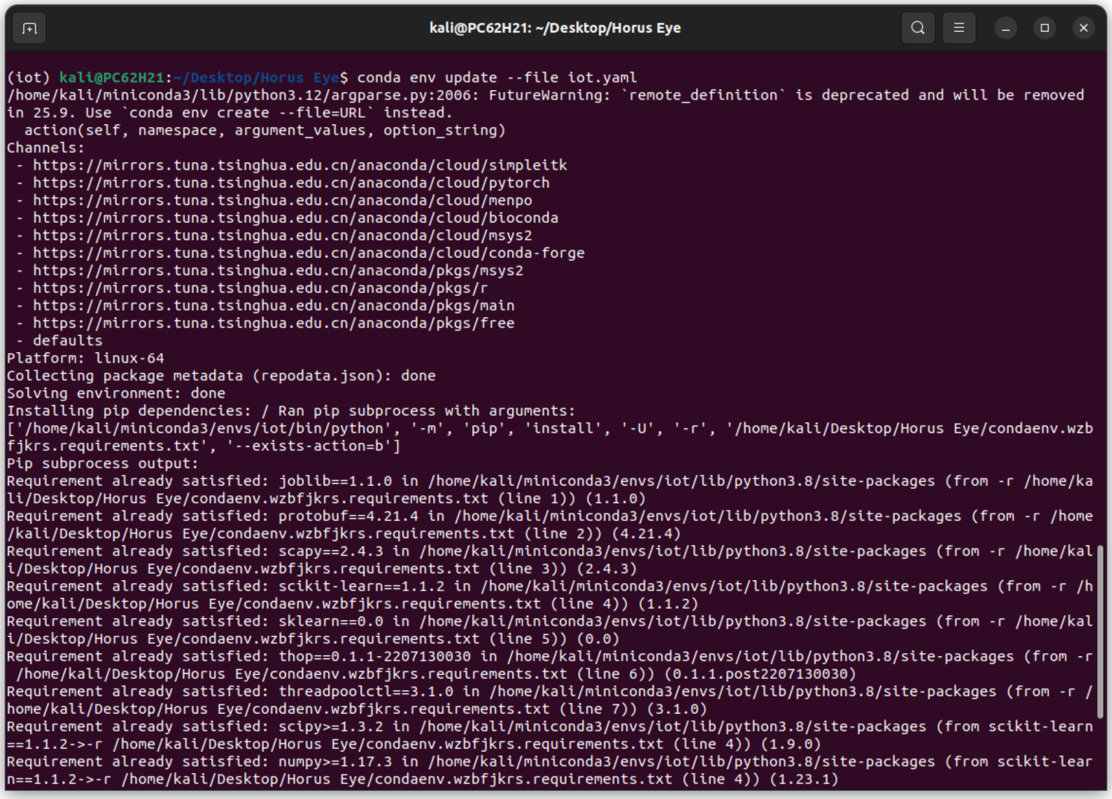
1.1 Setup Documentation

Reproduction page: https://github.com/Muhannad-Alshareef/HorusEye-reproduction/
Original github page: https://github.com/vicTorKd/HorusEye
for a complete setup instructions and experiments details follow the README file and the appendix of the paper.

Hardware Specifications:

OS:	Ubuntu 22.04.5 LTS (Dual Boot)
Processor:	Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz 2.50 GHz
GPU:	NVIDIA GeForce GTX 850 cuda version 12.4

Environment setup:
Following the instruction provided in the README file, the environment was setup successfully using miniconda following the .yaml file in the main project directory as seen in Figure(1).



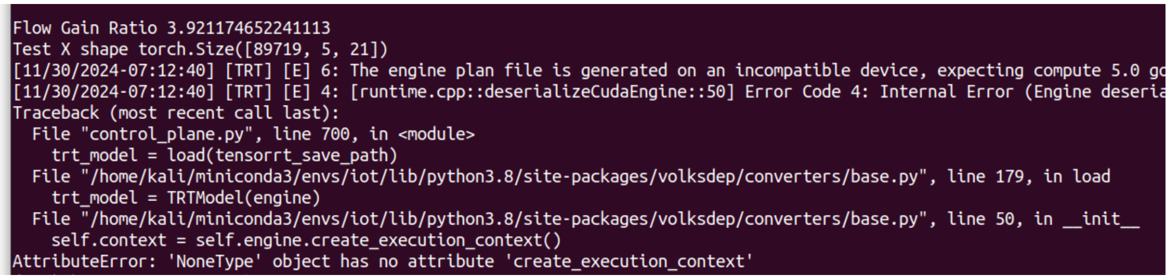
```
(iot) kali@PC62H21:~/Desktop/Horus Eye$ conda env update --file iot.yaml
/home/kali/miniconda3/lib/python3.12/argparse.py:2006: FutureWarning: `remote_definition` is deprecated and will be removed
in 25.9. Use `conda env create --file=URL` instead.
  action(self, namespace, argument_values, option_string)
Channels:
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/simpleitk
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/menpo
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/bioconda
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/msys2
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
 - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done
Installing pip dependencies: / Ran pip subprocess with arguments:
['/home/kali/miniconda3/envs/iot/bin/python', '-m', 'pip', 'install', '-U', '-r', '/home/kali/Desktop/Horus Eye/condaenv.wzb
fjkrs.requirements.txt', '--exists-action=b']
Pip subprocess output:
Requirement already satisfied: joblib==1.1.0 in /home/kali/miniconda3/envs/iot/lib/python3.8/site-packages (from -r /home/ka
li/Desktop/Horus Eye/condaenv.wzbfjkrs.requirements.txt (line 1)) (1.1.0)
Requirement already satisfied: protobuf==4.21.4 in /home/kali/miniconda3/envs/iot/lib/python3.8/site-packages (from -r /home
/kali/Desktop/Horus Eye/condaenv.wzbfjkrs.requirements.txt (line 2)) (4.21.4)
Requirement already satisfied: scapy==2.4.3 in /home/kali/miniconda3/envs/iot/lib/python3.8/site-packages (from -r /home/kal
i/Desktop/Horus Eye/condaenv.wzbfjkrs.requirements.txt (line 3)) (2.4.3)
Requirement already satisfied: scikit-learn==1.1.2 in /home/kali/miniconda3/envs/iot/lib/python3.8/site-packages (from -r /h
ome/kali/Desktop/Horus Eye/condaenv.wzbfjkrs.requirements.txt (line 4)) (1.1.2)
Requirement already satisfied: sklearn==0.0 in /home/kali/miniconda3/envs/iot/lib/python3.8/site-packages (from -r /home/kal
i/Desktop/Horus Eye/condaenv.wzbfjkrs.requirements.txt (line 5)) (0.0)
Requirement already satisfied: thop==0.1.1-2207130030 in /home/kali/miniconda3/envs/iot/lib/python3.8/site-packages (from -r
 /home/kali/Desktop/Horus Eye/condaenv.wzbfjkrs.requirements.txt (line 6)) (0.1.1.post2207130030)
Requirement already satisfied: threadpoolctl==3.1.0 in /home/kali/miniconda3/envs/iot/lib/python3.8/site-packages (from -r /
home/kali/Desktop/Horus Eye/condaenv.wzbfjkrs.requirements.txt (line 7)) (3.1.0)
Requirement already satisfied: scipy>=1.3.2 in /home/kali/miniconda3/envs/iot/lib/python3.8/site-packages (from scikit-learn
==1.1.2->-r /home/kali/Desktop/Horus Eye/condaenv.wzbfjkrs.requirements.txt (line 4)) (1.9.0)
Requirement already satisfied: numpy>=1.17.3 in /home/kali/miniconda3/envs/iot/lib/python3.8/site-packages (from scikit-lear
n==1.1.2->-r /home/kali/Desktop/Horus Eye/condaenv.wzbfjkrs.requirements.txt (line 4)) (1.23.1)
```

Figure 1. successful environment setup using the yaml file included.

Two of the libraries required in Experiment 3 did not successfully run which are:

- TensorRT
- Volksdep

The error displayed can be seen in Figure(2), no solution is found yet.

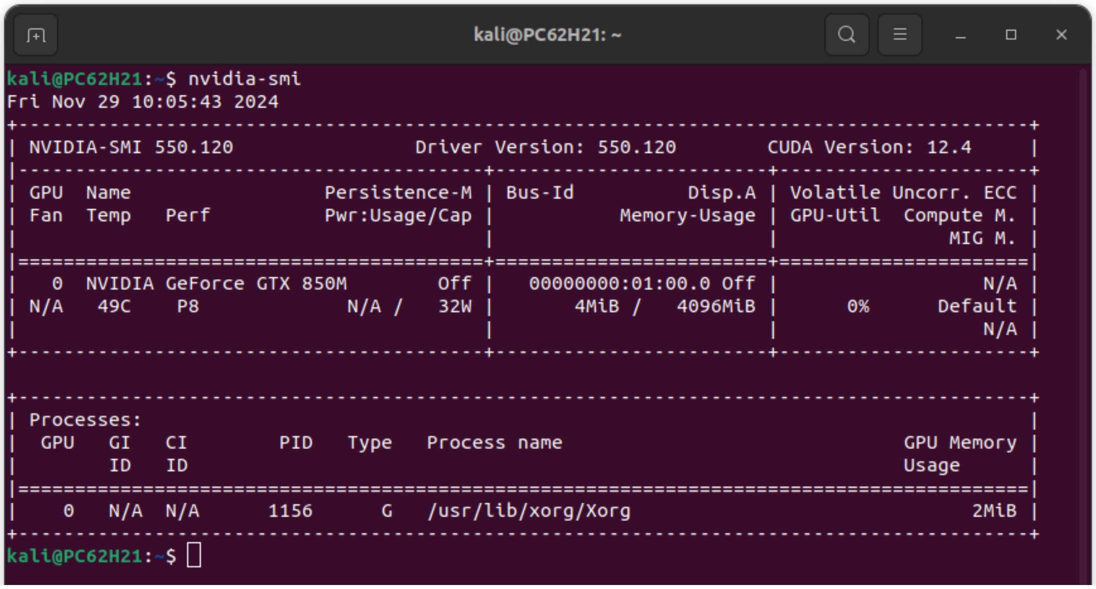


```
Flow Gain Ratio 3.921174652241113
Test X shape torch.Size([89719, 5, 21])
[11/30/2024-07:12:40] [TRT] [E] 6: The engine plan file is generated on an incompatible device, expecting compute 5.0 go
[11/30/2024-07:12:40] [TRT] [E] 4: [runtime.cpp::deserializeCudaEngine::50] Error Code 4: Internal Error (Engine deseria
Traceback (most recent call last):
  File "control_plane.py", line 700, in <module>
    trt_model = load(tensorrt_save_path)
  File "/home/kali/miniconda3/envs/iot/lib/python3.8/site-packages/volksdep/converters/base.py", line 179, in load
    trt_model = TRTModel(engine)
  File "/home/kali/miniconda3/envs/iot/lib/python3.8/site-packages/volksdep/converters/base.py", line 50, in __init__
    self.context = self.engine.create_execution_context()
AttributeError: 'NoneType' object has no attribute 'create_execution_context'
```

Figure 2. error message while running Experiment 3 after installing TensorRT and volksdep.

- Troubleshooting

1. It is important to use a Linux distribution to setup the environment.
   Using a virtual machine will not work since the project uses cuda which is available for NVIDIA GPUs.
   You can check your cuda version by running the command: "nvidia-smi".
   The output will be similar to Figure(3).



```
kali@PC62H21:~$ nvidia-smi
Fri Nov 29 10:05:43 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.120                Driver Version: 550.120        CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce GTX 850M        Off |   00000000:01:00.0 Off |                  N/A |
| N/A   49C    P8             N/A /   32W |       4MiB /   4096MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A      1156      G   /usr/lib/xorg/Xorg                              2MiB |
+-----------------------------------------------------------------------------------------+
kali@PC62H21:~$ 
```

Figure 3. NVIDIA specifications of the testing PC.

2. Create 5 folders under ./result
   a. HorusEye
   b. rmse
   c. Open-Source/HorusEye
   d. Magnifier
   e. kitsune

1.2 Code Understanding

Code file name	Calls to	Purpose
Control_plane.py	iForest_detect.py load_data.py AE.py	Parse arguments e.g. “train False” Loads, trains, and validate the models.
FE.py	netStat.py Afterimage.py	Flow level feature extraction
Convert_model.py	-	Used by TensorRT for model optimization
pcap2csv.py **csv_process.py** **extract_flow_size.py**	-	Flow and burst level feature extraction of raw pcap files.
Iot_dect_waterflow8.p4	-	Data plane iForest implementation in P4 language

The main program is:

- Control_plane.py
  The main function is used to carry experiments 1 through 5 detailed in the paper’s appendix.
  On a high level, the program will do the following:
  1. initialize seeds and parse arguments like "--train False --experiment A".
  2. load model and attack csv paths.
  3. if kitsune is not used, used magnifier.
  4. load data after processing.
  5. save a record of performance metrics per attack type.
  6. if train is True, train the iForest, and AE (including magnifier and kitsune).
  7. if test is True, test the models under several if statements. (open-source, kitsune)
  8. if test_robust is True, the data will be poisoned to test the robustness of the models under certain noise additions.
  9. report the confusion matrix per attack in the command line.
  10. report the performance such as memory usage, throughput, AUC and more in csv files under ./result.

• Insights:

The program fails to create the necessary folders where the results are saved, these folders should be created manually. This is due to the program looking for the main directory ./result only and not the subfolders needed.

Moreover, each run of the program overwrites the results of the previous run. Necessitating copying the files before running the program again.

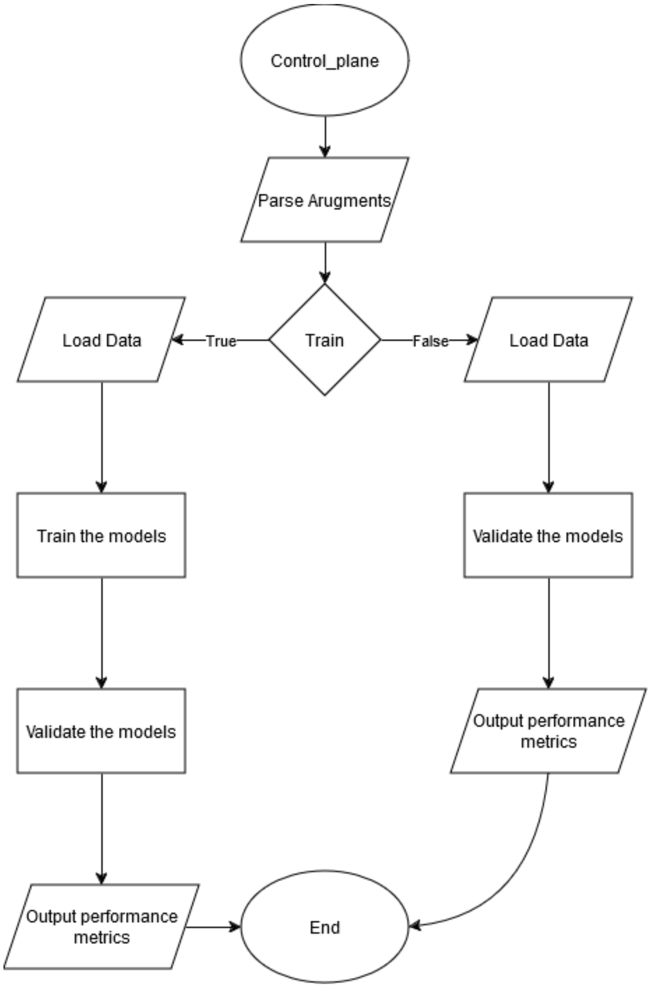


Figure 4. flowchart of the program control plane.

1.3 Input and Output Demonstrations

The following are the inputs and outputs of experiment 1.

All the inputs are under the folder ./DataSets/ Anomaly/attack-flow-level- device/

- Inputs
- data_theft-00.csv
- http_ddos-00.csv
- port_scan_router-00.csv
- bashlite-00.csv
- udp_ddos_router-00.csv
- tcp_ddos_router-00.csv
- os_scan-00.csv
- udp_ddos-00.csv
- mirai-00.csv
- mirai_router_filter-00.csv
- service_scan-00.csv

- aidra-00.csv
- os_scan_router-00.csv
- keylogging-00.csv
- tcp_ddos-00.csv

- Outputs for each attack type:
- raw_rmse csv.
- roc_record_split csv.
- record attack csv containing:
    - false positive rate 1
    - true positive rate 1
    - thershold 1
    - false positive rate 2
    - true positive rate 2
    - thershold 2
    - pr_auc
    - auc_eval

• Test Cases:

- Experiment 1 evaluates the detection performance and throughput of the model on the authors' dataset. Successful reproduction is summarized in table 1.

- Experiment 2 evaluates the detection performance of the model on public dataset. Which can also be seen in table(1).

- Experiment 3 required TensorRT, which produced errors while running.

- Experiment 4 evaluates the detection performance (robustness) of Gulliver Tunnel under three common black-box attacks. Which can be seen in Figure(5).

- Experiment 5 requires a programmable switch, which is not available. Future implementation using open vSwitch will be tested.

• Comparative Analysis:

Minor discrepancies are noted when the model is retrained due to hardware differences and autoencoders used. The differences are seen in table(1).

1.4 Validation

Table 1.The table displays the original performance metrics under (HorusEye), and findings of the model reproduction with and without retraining.

Dataset	Attack	HorusEye			Reproduction			Reproduction after retraining		
		TPR		PR_{AUC}	TPR		PR_{AUC}	TPR		PR_{AUC}
		≤5e-e5	≤5e-e4		≤5e-e5	≤5e-e4		≤5e-e5	≤5e-e4	
Public	Aidra	0.383	0.469	0.657	0.383	0.469	0.657	0.366	0.478	0.664
	Bashlite	0.713	0.735	0.817	0.713	0.735	0.817	0.726	0.744	0.819
	Mirai	0.964	0.966	0.980	0.963	0.965	0.980	0.962	0.965	0.980
	Keylogging	0.527	0.528	0.806	0.527	0.528	0.806	0.527	0.537	0.787
	Data theft	0.508	0.510	0.810	0.508	0.510	0.810	0.508	0.541	0.798
	Service scan	0.334	0.363	0.934	0.334	0.363	0.934	0.343	0.392	0.947
	OS scan	0.498	0.577	0.946	0.498	0.577	0.946	0.547	0.616	0.950
	HTTP DDoS	0.285	0.408	0.942	0.285	0.408	0.942	0.304	0.450	0.931
	TCP DDoS	0.903	0.912	0.929	0.903	0.912	0.929	0.902	0.910	0.927
	UDP DDoS	0.965	0.973	0.990	0.965	0.973	0.990	0.964	0.973	0.990
HorusEye	Mirai	0.303	0.424	0.868	0.303	0.424	0.868	0.284	0.428	0.872
	Service scan	0.991	0.996	1.000	0.991	0.996	1.000	0.994	0.996	1.000
	OS scan	0.968	0.985	0.999	0.968	0.985	0.999	0.969	0.988	1.000
	TCP DDoS	0.997	0.998	1.000	0.997	0.998	1.000	0.997	0.998	1.000
	UDP DDoS	0.998	0.998	1.000	0.998	0.998	1.000	0.998	0.998	1.000

Reproduction results using the model's hyperparameters show an exact match rounding to 3 decimal places. While the retrained model has slight discrepancies due to hardware differences and the use of neural networks, metrics for attacks such as Bashlite and OS scan show slight improvement while other attacks such as Aidra and Mirai have lower TPR. Note that the retraining process will take around 3 hours.

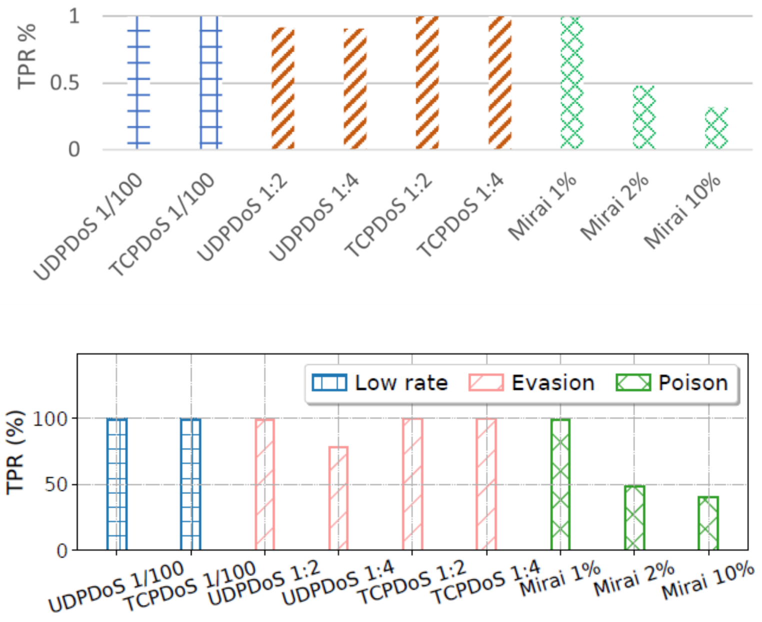


Figure 5. robustness of Gulliver tunnel, the top graph shows the reproduction results.

Slight differences can be seen especially in UDPDoS 1:2 which might be due to the poisoning algorithm used to test it.