Anomaly based IDS using transformer Model

Instructor : Dr. Abhishek Vaish

INDEX

1. Problem Statement
2. Objective
3. Technical Challenges
4. State of the Art
5. Scope of the Project
6. Result & Discussion
7. References

Problem statement

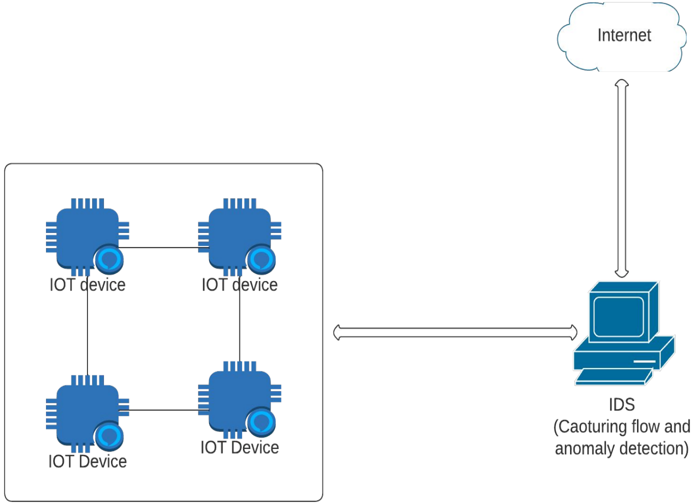
Design an IDS for based on shallow transformer model.

Objective

- To create an IDS(Intrusion Detection System) based on shallow Transformer Model.
- The placement method for our ids is centralised.
- It is lightweight as compared to other DL models, without much compromising with accuracy.

Use Case Diagram :

Centralised IDS for detecting attacks on IOT networks.



Literature review

Paper	Description	Advantages	Disadvantages	Result
Meftah, S., Rachidi, T., & Assem, N. (2019). Network based intrusion detection using the UNSW-NB15 dataset	In two stages, we detect network intrusions using UNSW-NB15. Optimizing with Recursive Feature Elimination and Random Forests, we classify intrusive traffic with Logistic Regression, Gradient Boost Machine, and Support Vector Machine. Support Vector Machine achieves the highest	This model uses machine learning models, which are lightweight as compared to deep learning models.	Model accuracy is only about 80 - 85%.	The model gives an accuracy of 86% with decision tree, 83% with SVM

Paper	Description	Advantages	Disadvantages	Result
M. Hassan, A. Gumaei, A. Alsanad, M. Alrubaian, and G. Fortino, “A Hybrid Deep Learning Model for Efficient Intrusion Detection in Big Data Environment”, Information Sciences, vol. 513, 2019.	This paper includes a NIDS (Network Intrusion Detection System) that combines Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) architectures using deep learning techniques.	The CNN layers are utilised to efficiently extract significant features from network data because of their weight-sharing property, resulting in faster processing speed.	The computational load is very high.	The hybrid model achieved an overall classification accuracy of 97.1% for binary classification and 98.43% for the multi-class case when tested on the UNSW-NB15 dataset
Loc Gia Nguyen Kohei Watabe Flow-based Network Intrusion Detection Based on BERT Masked Language Model	This study utilized BERT model for the representation of flow sequences and an MLP classifier to discriminate between benign and malicious flows. Early experimental results showed that the proposed method is capable of achieving good and consistent results across different domains	Promising results across domains	Poor domain adaptation capability of conventional ML-based classifiers	All classifiers excelled on CIDDS-001 internal tests, outperforming other domains. Proposed method consistently beat EFC. Smaller balanced datasets showed

State of the Art

Paper	Description	Results
Learning autoencoder ensembles for detecting malware hidden communications in IoT ecosystems	The dataset used here contains the traffic flows exchanged by 28 different IoT nodes(JIIS23). The work of this papers is to reveal covert communications hidden in especially in the TTL of IPv4 traffic, which is done by auto-encoder-decoder model.	The hybrid model achieved an overall classification recall of 91% and precision of 96% for the JIIS dataset.

Learning autoencoder ensembles for detecting malware hidden communications in IoT ecosystems Nunziato Cassavia1·Luca Caviglione2·Massimo Guarascio1·Angelica Liguori1,3 ·Marco Zuppelli2Received: 23 May 2023 / Revised: 12 September 2023 / Accepted: 26 September 2023© The Author(s) 2023

Technical Challenges

- **Low Resource Utilisation :** Anomaly based IDS takes a lot of resources in its operation. Our goal is to minimize it without compromising much on accuracy of detection.
- **Mitigating False Positives** : Mitigating false positives in Intrusion Detection Systems (IDS) presents a multifaceted challenge. To solve this issue, we have to fine-tune the model.
- **Large-scale attacks :** Large no. of attacks might be possible. Our model should be able to detect such type of attacks.

Dataset Description :

1. UNSW-NF-v2 :
   a. The dataset contains attacks like Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms.
   b. It contains 95,053 attack samples and 2,295,222 benign samples.
   c. It contains 43 netflow features like IPV4_SRC_ADDR , L4_SRC_PORT, IPV4_DST_ADDR L4_DST_PORT, PROTOCOL, L7_PROTO, IN_BYTES, IN_PKTS, OUT_BYTES, etc.

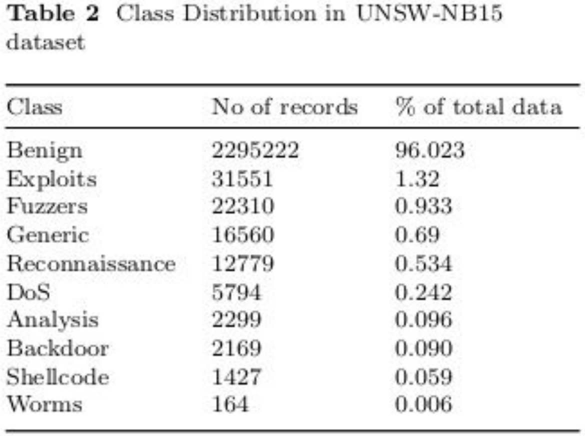


Table 2 Class Distribution in UNSW-NB15 dataset

Class	No of records	% of total data
Benign	2295222	96.023
Exploits	31551	1.32
Fuzzers	22310	0.933
Generic	16560	0.69
Reconnaissance	12779	0.534
DoS	5794	0.242
Analysis	2299	0.096
Backdoor	2169	0.090
Shellcode	1427	0.059
Worms	164	0.006

Dataset Description

2. JIIS23 :

    a. It contain the statistics of the traffic containing covert channels in the TTL field of 3 days considering the Naive Encoding Scheme.

    b. It contains features : timestamp, num_pkts, avg_ttl, median_ttl, 10_percentil_ttl, 25_percentil_ttl, 75_percentil_ttl, 90_percentil_ttl, max_ttl,min_ttl

Implementation

1) Dataset Collection:

- Packets are collected and flows are made using nProbe.
- Data is stored in sql.

2) Dataset Ingestion:

- Data is read from sql, and passed in model.
- Flow Transformer ingests tabular datasets, handling missing values.
- Requires a dataset specification for features, class column, and benign traffic label.

3) Pre-processing:

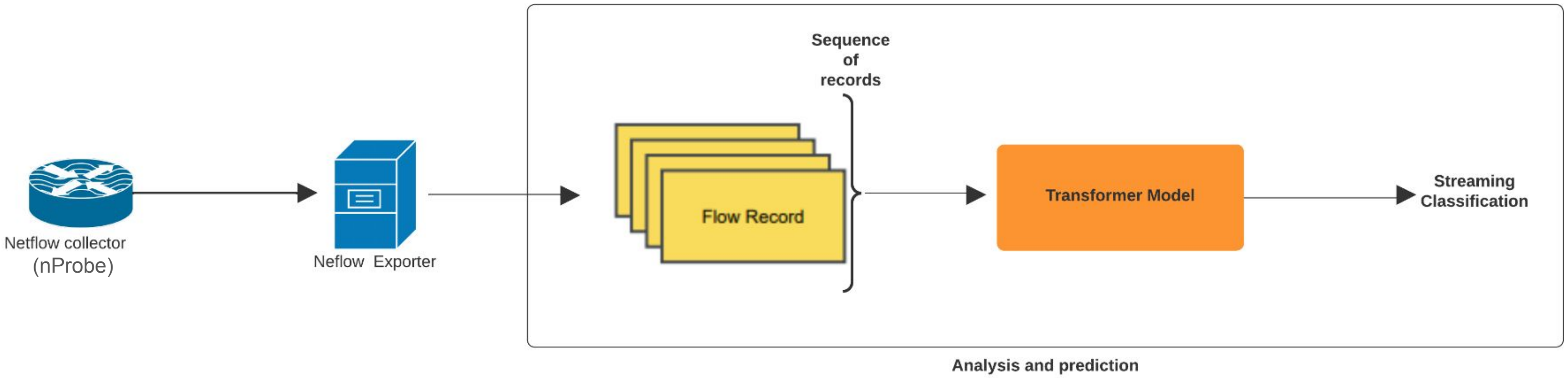
- Adapts to categorical formats and scales numerical features.

4) Model Pipeline:

- Construct a transformer model with interchangeable components (encoder, transformer, classification head).

5)Output:

- Gives output after processing in transformer model.



Framework for NetFlow processing using a transformer model

TRANSFORMER FRAMEWORK

- Transformer is a framework for fast development of transformer-based Network Intrusion Detection Systems (NIDS).

- The framework includes options for encoding categorical and numerical fields in flow data. Flow data is a type of tabular data, consisting of two types of fields:

  numerical fields - like number of packets sent, etc.
  categorical fields - like port no., etc.

- Pre-processed flow records are encoded into fixed-length vectors by the input encoder for the transformer

- The framework supports different transformer components like input encodings, transformer block and classification heads.
- The data of encoder block is passed to classification head, which is an MLP using adam optimizer and binary cross entropy as as loss function.
- Classification head used is "Last Token" classification, in which last layer of output of encoder is passed to classification head.
- It was evaluated on UNFW-NF-v2 and JIIS23 dataset, and the performance of different models were compared.

Shallow transformer models were found to be sufficient for certain NIDS tasks due to their size and higher throughput as compared to other big models.

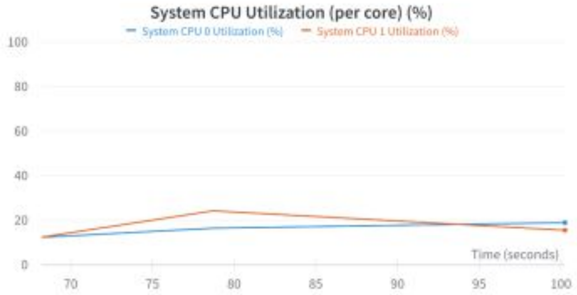
RESULT(UNSW-NF-v2)

Techniques	LSTM model	Transformer Model	Remarks
Accuracy	96.3	97.0	Here, we can see Transformer Model is performing with better accuracy and than the LSTM Model because of the choices in classification heads,architecture we propose in Transformer Model.
F-measure	89.1	90.54	Better F1 score than LSTM model

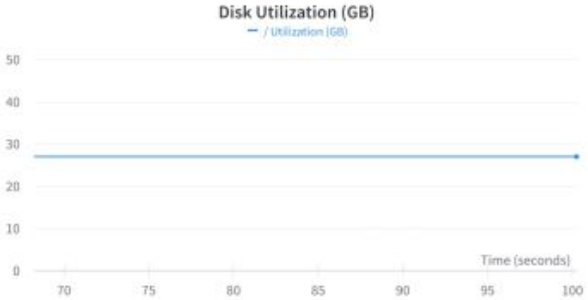
RESULT(JIIS23)

Techniques	Ensemble Auto Encoder	Transformer Model	Remarks
Accuracy	0.90	0.89	Our transformer model is giving similar accuracy as ensemble auto encoder model.
Precision	0.80	0.86	Our transformer model is giving better precision than ensemble auto encoder model.
Recall	0.96	0.85	It is giving low recall values as compared to ensemble auto encoder model.
F-measure	0.87	0.85	It is giving similar f-measure as compared to ensemble auto encoder model.

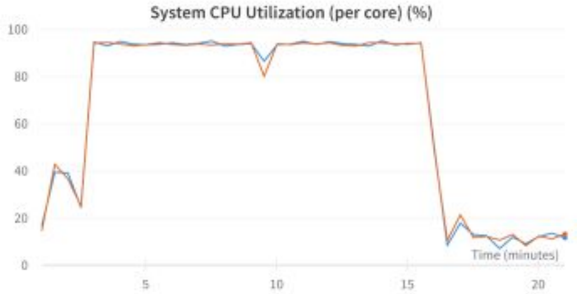
Transformer model



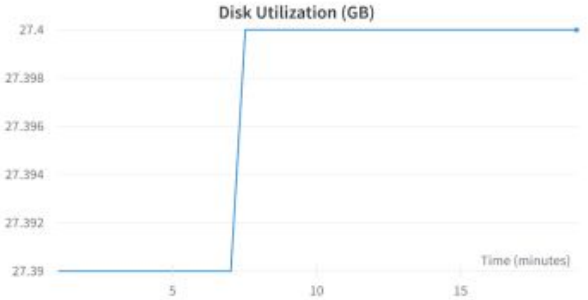
Transformer model



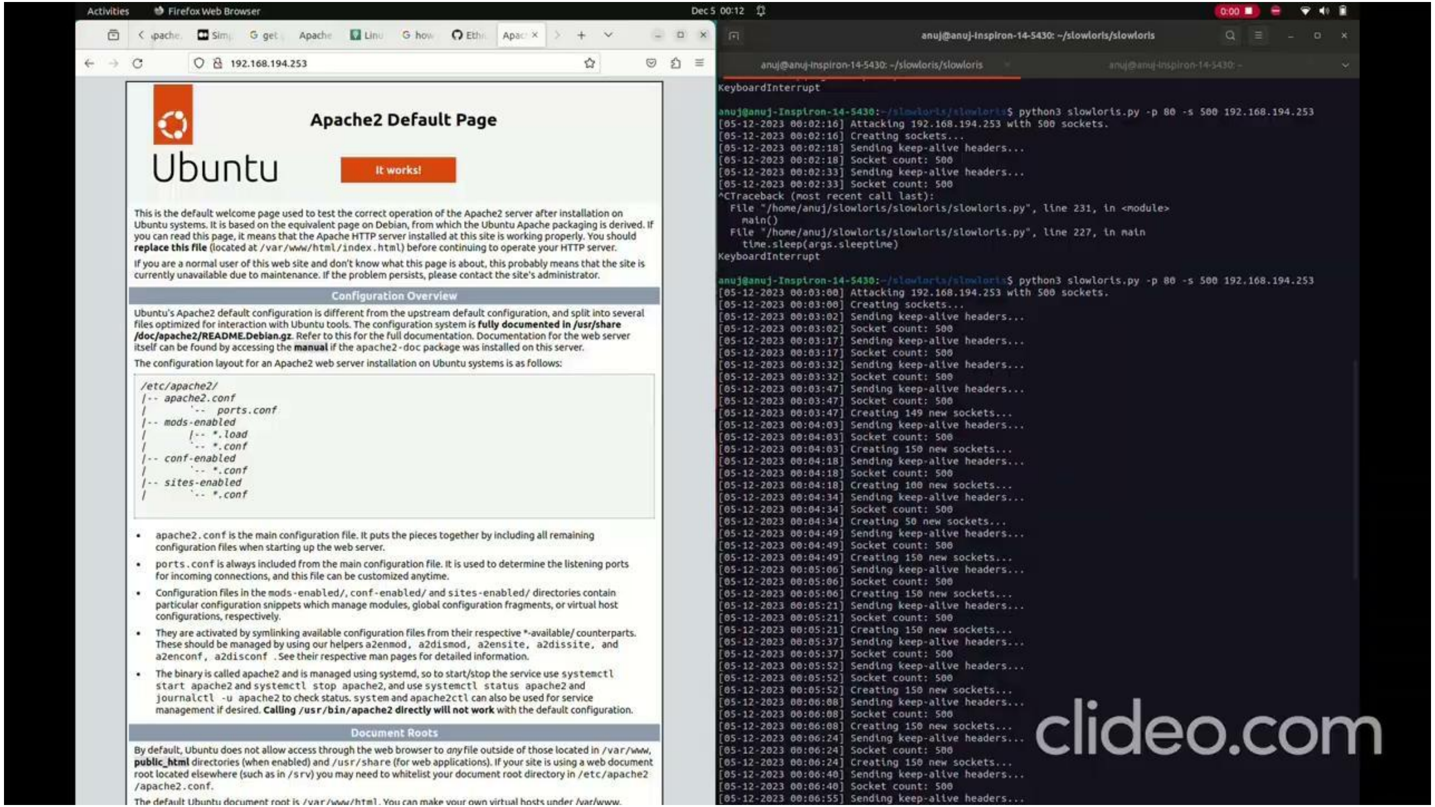
LSTM model



LSTM model



Demo Video



Activities
Firefox Web Browser
Dec 5 00:12
0:00
192.168.194.253
anuj@anuj-Inspiron-14-5430: ~/slowloris/slowloris
anuj@anuj-Inspiron-14-5430: ~
clideo.com

Apache2 Default Page

Ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf` . See their respective man pages for detailed information.
- The binary is called apache2 and is managed using systemd, so to start/stop the service use `systemctl start apache2` and `systemctl stop apache2`, and use `systemctl status apache2` and `journalctl -u apache2` to check status. `system` and `apache2ctl` can also be used for service management if desired. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file outside of those located in `/var/www`, **public_html** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under /var/www.

```
KeyboardInterrupt

anuj@anuj-Inspiron-14-5430:~/slowloris/slowloris$ python3 slowloris.py -p 80 -s 500 192.168.194.253
[05-12-2023 00:02:16] Attacking 192.168.194.253 with 500 sockets.
[05-12-2023 00:02:16] Creating sockets...
[05-12-2023 00:02:18] Sending keep-alive headers...
[05-12-2023 00:02:18] Socket count: 500
[05-12-2023 00:02:33] Sending keep-alive headers...
[05-12-2023 00:02:33] Socket count: 500
^CTraceback (most recent call last):
  File "/home/anuj/slowloris/slowloris/slowloris.py", line 231, in <module>
    main()
  File "/home/anuj/slowloris/slowloris/slowloris.py", line 227, in main
    time.sleep(args.sleeptime)
KeyboardInterrupt

anuj@anuj-Inspiron-14-5430:~/slowloris/slowloris$ python3 slowloris.py -p 80 -s 500 192.168.194.253
[05-12-2023 00:03:00] Attacking 192.168.194.253 with 500 sockets.
[05-12-2023 00:03:00] Creating sockets...
[05-12-2023 00:03:02] Sending keep-alive headers...
[05-12-2023 00:03:02] Socket count: 500
[05-12-2023 00:03:17] Sending keep-alive headers...
[05-12-2023 00:03:17] Socket count: 500
[05-12-2023 00:03:32] Sending keep-alive headers...
[05-12-2023 00:03:32] Socket count: 500
[05-12-2023 00:03:47] Sending keep-alive headers...
[05-12-2023 00:03:47] Socket count: 500
[05-12-2023 00:03:47] Creating 149 new sockets...
[05-12-2023 00:04:03] Sending keep-alive headers...
[05-12-2023 00:04:03] Socket count: 500
[05-12-2023 00:04:03] Creating 150 new sockets...
[05-12-2023 00:04:18] Sending keep-alive headers...
[05-12-2023 00:04:18] Socket count: 500
[05-12-2023 00:04:18] Creating 100 new sockets...
[05-12-2023 00:04:34] Sending keep-alive headers...
[05-12-2023 00:04:34] Socket count: 500
[05-12-2023 00:04:34] Creating 50 new sockets...
[05-12-2023 00:04:49] Sending keep-alive headers...
[05-12-2023 00:04:49] Socket count: 500
[05-12-2023 00:04:49] Creating 150 new sockets...
[05-12-2023 00:05:06] Sending keep-alive headers...
[05-12-2023 00:05:06] Socket count: 500
[05-12-2023 00:05:06] Creating 150 new sockets...
[05-12-2023 00:05:21] Sending keep-alive headers...
[05-12-2023 00:05:21] Socket count: 500
[05-12-2023 00:05:21] Creating 150 new sockets...
[05-12-2023 00:05:37] Sending keep-alive headers...
[05-12-2023 00:05:37] Socket count: 500
[05-12-2023 00:05:52] Sending keep-alive headers...
[05-12-2023 00:05:52] Socket count: 500
[05-12-2023 00:05:52] Creating 150 new sockets...
[05-12-2023 00:06:08] Sending keep-alive headers...
[05-12-2023 00:06:08] Socket count: 500
[05-12-2023 00:06:08] Creating 150 new sockets...
[05-12-2023 00:06:24] Sending keep-alive headers...
[05-12-2023 00:06:24] Socket count: 500
[05-12-2023 00:06:24] Creating 150 new sockets...
[05-12-2023 00:06:40] Sending keep-alive headers...
[05-12-2023 00:06:40] Socket count: 500
[05-12-2023 00:06:55] Sending keep-alive headers...
```

References

1. An effective model for anomaly IDS to improve the efficiency | IEEE Conference Publication | IEEE Xplore
2. Ultra-Lightweight Deep Packet Anomaly Detection for Internet of Things Devices (ipccc.org)
3. A Host-Based Intrusion Detection and Mitigation Framework for Smart Home IoT Using OpenFlow | IEEE Conference Publication | IEEE Xplore
4. 1805.03409.pdf (arxiv.org)
5. A hybrid deep learning model for efficient intrusion detection in big data environment - ScienceDirect
6. Flow-based Network Intrusion Detection Based on BERT Masked Language Model (arxiv.org)

Thank You!