Artificially Intelligent Intrusion Detection System

Charles Rizzo
University of Tennessee
crizzo@vols.utk.edu

Nick Skuda
University of Tennessee
nskuda@vols.utk.edu

Austin Saporito
University of Tennessee
dxh594@vols.utk.edu

***Abstract*—Intrusion Detection Systems are software systems that monitor network traffic for malicious activity. Complex systems are multifaceted and have functionalities like deep packet inspection, breach detection, and anomaly detection. The systems respond to suspicious activity by causing alerts and notifying network administrators of the offending behavior.**

Machine learning methodologies have been studied and developed to perform tasks like classification, regression, and even event detection. This paper seeks to apply several different machine learning methodologies to the UNSW-NB15 network intrusion detection data set. By applying and comparing several different models to the data set, the goal is to maximize malicious event classification while minimizing false alarms and determine which models best differentiate between malicious and normal behavior.

***Index Terms*—UNSW-NB15, Intrusion detection system, machine learning**

I. Introduction

The 1998 and 1999 DARPA Intrusion Detection Data Sets are several weeks worth of raw network TCP data that was collected and organized by Lincoln Laboratory MIT. This data was used to study and develop intrusion detection systems that were subsequently delivered to the Air Force Research Laboratory for real-time evaluation [1]. These data sets were considered the standards for developing network intrusion detection systems for over two decades. However, they are indeed two decades outdated and needed to be updated. To accomplish this, the UNSW-NB15 data set was created in 2015 as a hybrid of real modern normal and the contemporary synthesized attack activities of the network traffic [2].

The raw data was about 100 gigabytes of network traffic as pcap files that yielded 2,540,044 connection records. The data was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security. The Argus and Bro-IDS tools were used along with 12 algorithms to transform the the data into connection records comprised of a set of 49 features with label. Each connection record is defined as a sequence of TCP packets that have well-defined starting and ending times as well as discrete source and destination IP addresses. Each connection record is labeled either as normal or as one of 9 types of attacks. [2].

The attacks are labeled according as one of the nine following types: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. In the appendix are Tables I, II, III, IV, V, VI, and VII that list and detail each of the 49 features that comprises a connection record [2]. The facts that the UNSW-NB15 data set has more features per connection record and includes real modern normal attack activities are two main motivations for the usage of this data set compared to the DARPA-derived KDD-Cup data set.

The conversion of the raw TCP data into connection records facilitates the use of machine learning classification techniques to be fit to the data and make predictions as to what kind of attack, if any, a test data set connection record is. This work aims to evaluate the performance of several machine learning models provided by Scikit-Learn in order to determine how certain techniques do across the entire data set. Because two of the authors (Nick and Charlie) are members of the TENNLab Neuromorphic group, this work also leverages and compares the performance of recurrent, spiking neural networks against classical machine learning algorithms. The motivation for doing this is that recurrent, spiking neural networks are typically smaller and therefore more lightweight than conventional neural networks (in the form of Multi-layer perceptrons). Their lightweightedness contributes to their ability to function and produce classifications in real-time, an important faculty for a network intrusion detection system to possess.

II. Methodology

In the spirit of informational reduction, we will most likely reduce the feature set size using Principal Component Analysis. We'll likely opt to retain 90% of the variance in the principal components (or introduce a 10% error) which will reduce our feature space, resulting in models developing more quickly. The trade-off with reducing the amount of features is typically accuracy for training and prediction time. We'll reduce the amount of features across the board as opposed to hand tuning and selecting certain features.

We plan to use only supervised machine learning techniques through Scikit-Learn, aside from the Neuromorphi approach. This decision to not use unsupervised techniques is due to the fact that this is a labeled data set; therefore, it doesn't make sense to us to use an unsupervised machine learning technique for this task. The supervised machine learning models that we will trained and developed were the following: k-nearest neighbors, backpropagation neural networks (multilayer perceptrons), decision tree, random forest, support vector machines (SGD Classifier), and Naive Bayes. Each classifier will have its own dedicated section in the results section detailing how it works at a high level, its performance, and its best parameters. Austin and Charlie will split the workload of Scikit-Learn classifiers evenly since there are 8 classifiers. Nick will focus his efforts on neuromorphic classifier because of how much more complicated it will be to get setup with the data (writing a classification application, selecting a neuroprocessor, choosing input/output encoding schemes, and the hyperparameterization that comes with all of those tasks).

We plan to develop each model to distinguish between all of the different attacks as as well as a binary scheme where the models merely predict either "normal" or "malicious" behavior. Ultimately, a binary classification scheme is a more realistic approach because in the real world, a model would never be able to correctly label an attack by name if it had not been trained to recognize that particular attack. However, it is a realistic expectation for a model to be able to differentiate between normal and malicious behavior based on prior known (and trained on) malicious behavior or attacks.

For evaluation, we will generate training and testing F1-scores for each model across the entire data set. The reason we are going to report F1-scores instead of raw accuracy is because there is a massive imbalance in the distribution of events in the data set. Reporting F1-scores will take into account the difference in different label weights relative to how many samples it occupies in the data set. Just like normal accuracy, however, the best F1-score a model can achieve is 1. While developing the models, we will use 10-fold cross validation to ensure that our best generalizing models' parameter sets are the ones that are reported as being the best instead of merely reporting the parameter combinations with the best performances on isolated instances of the data set. We will also generate confusion matrices for each model and a Receiver Operating Characteristic (ROC) graph to detail each models' separability performance overall. We will also report and highlight both the performance (or true detection) and false alarm/false positive rates per model for the binary classification experiment.

A known issue in machine learning is the problem of hyper-parameter selection. The values of some models' parameters are continuous, which means there are an infinite number of ways that a particular model can be created and trained around the data set. In order to address this, we will define a static list of parameter values that each of the models' could take and randomly select 100 combinations of these parameters to test and generate models from. This random search approach is shown to be about as effective as an exhaustive grid search but saves much in the way of time and computational resources [3].

Hardware-wise, each of the models will be trained and developed on the Neuro-Firewall computer cluster at the University of Tennessee at Knoxville. It is a 1728 core system comprising 36 Dell PowerEdge C6145s. Each node features quad 12 core 64 bit AMD Opteron Processor 6180 SEs and 96 GBs of RAM. In terms of software, all of the code will most likely be written in and run with Python 3.7.4, and all of the models and evaluation metrics will come from the Scikit-Learn 0.21.3 library [4].

III. Timeline and Expected Results

Timeline-wise, we will aim to have the models trained and developed by the end of October. While this is a broad goal, some models will be more difficult to work with than others; this is especially true for the neuromorphic model. The goal is to have most of November to compile the data, generate graphics, and type the final report of our findings.

Currently, we believe that either the MLP or Random Forest classifier will yield the highest accuracy, but that the neuromorphic model will classify the most quickly.

References

[1] "1998 darpa intrusion detection evaluation dataset."

[2] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015.

[3] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research*, vol. 13, no. Feb, pp. 281–305, 2012.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Appendix

	Feature	**Type**	**Description**
1	srcip	nominal	Source IP address
2	sport	integer	Source port number
3	dstip	nominal	Destination IP address
4	dsport	integer	Destination port number
5	proto	nominal	Transaction protocol

TABLE I: Flow Features

	Feature	**Type**	**Description**
6	state	nominal	The state and its dependent protocol, e.g. ACC, CLO, else (-)
7	dur	float	Record total duration
8	sbytes	integer	Source to destination bytes
9	dbytes	integer	Destination to source bytes
10	sttl	integer	Source to destination time to live
11	dttl	integer	Destination to source time to live
12	sloss	integer	Source packets retransmitted or dropped
13	dloss	integer	Destination packets retransmitted or dropped
14	service	nominal	http, ftp, ssh, dns ..,else (-)
15	sload	float	Source bits per second
16	dload	float	Destination bits per second
17	spkts	integer	Source to destination packet count
18	dpkts	integer	Destination to source packet count

TABLE II: Basic Features

	Feature	**Type**	**Description**
19	swin	integer	Source TCP window advertisement
20	dwin	integer	Destination TCP window advertisement
21	stcpb	integer	Source TCP sequence number
22	dtcpb	integer	Destination TCP sequence number
23	smeansz	integer	Mean of the flow packet size transmitted by the src
24	dmeansz	integer	Mean of the flow packet size transmitted by the dst
25	trnas_depth	integer	the depth into the connection of http request/response transaction
26	res_bdy_len	integer	The content size of the data transferred from the server's http service.

TABLE III: Content Features

	Feature	**Type**	**Description**
27	sjit	float	Source jitter (mSec)
28	dhit	float	Destination jitter (mSec)
29	stime	timestamp	record start time
30	ltime	timestamp	record last time
31	sintpkt	float	Source inter-packet arrival time (mSec)
32	dintpkt	float	Destination inter-packet arrival time (mSec)
33	tcprtt	float	The sum of 'synack' and 'ackdat' of the TCP.
34	synack	float	The time between the SYN and the SYN_ACK packets of the TCP.
35	ackdat	float	The time between the SYN_ACK and the ACK packets of the TCP.

TABLE IV: Time Features

	Feature	**Type**	**Description**
36	is_sm_ips_ports	binary	If source (1) equals to destination (3)IP addresses and port numbers (2)(4) are equal, this variable takes value 1 else 0
37	ct_state_ttl	integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct_flw_http_mthd	integer	No. of flows that has methods such as Get and Post in http service.
39	is_ftp_login	binary	If the ftp session is accessed by user and password then 1 else 0.
40	ct_ftp_cmd	integer	No of flows that has a command in ftp session.

TABLE V: General Purpose Features

	Feature	Type	Description
41	ct_srv_src	integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct_srv_dst	integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct_dst_ltm	integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
44	ct_src_ltm	integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).
45	ct_src_dport_ltm	integer	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
46	ct_dst_sport_ltm	integer	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct_dst_src_ltm	integer	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).

TABLE VI: Connection Features

	Feature	Type	Description
48	attack_cat	nominal	The name of each attack category (e.g., Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms)
49	Label	binary	0 for normal and 1 for attack records

TABLE VII: Labelled Features