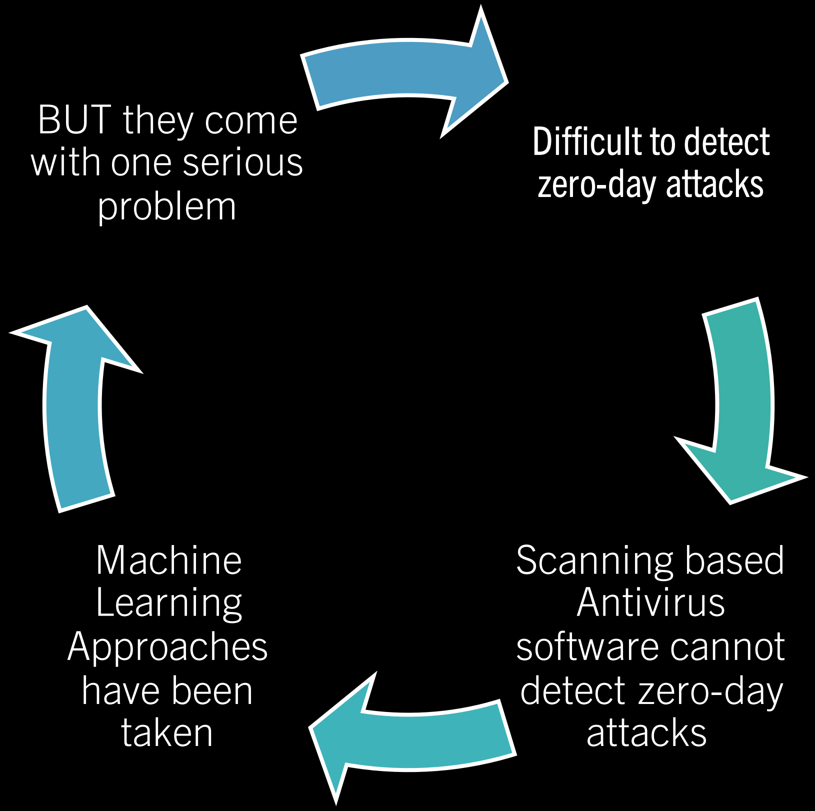
MULTILAYER PERCEPTRON FOR EFFICIENT AND ACCURATE ZERO-DAY ATTACK DETECTION

Ashim Dahal, Prabin Bajgai under the supervision of Dr. Nick Rahimi

THE PROBLEM


Scanning based Antivirus software cannot detect zero-day attacks
Machine Learning Approaches have been taken
BUT they come with one serious problem
Difficult to detect zero-day attacks

HOW CAN MACHINE LEARNING FAIL?

Because of high accuracy

Researchers focus on getting the best accuracy in the KDD99 dataset

But in cases like these, accuracy as a sole metric doesn't suffice

Our research focuses on reduced bias and increased variance

OUR DATASET AND LITERATURE REVIEW

- KDD99: 4.8 Million samples of 23 attack types, 2.8 Million belong to Smurf and 1 Million belong to Neptune
- Out of the 23 classes in the dataset, the sum of number of samples for bottom 20 is less than 50,000.
- 99.98% accuracy = 20 unnoticed classes
- Machine Learning learns from the data and these data make model biased

THE SOLUTION

We worked on a 2-step solution

Step 1: Make the dataset less biased in itself

Step 2: Build a robust ML model that acknowledges the disparity on the data distribution in the dataset

STEP 1: DEBIASING THE DATASET

Probe: back, land, neptune, pod, smurf, teardrop, apache, udpstorm, processtable, worm

DOS: satan, ipsweep, nmap, portsweep, mscan, saint

Unauthorized Access: guess_passwd, ftp_write, imap, phf, multihop, warezmaster, warezclient, spy, xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named, mailbomb, buffer_overflow, loadmodule, rootkit, perl, sqlattack, xterm, ps

Normal: normal

STEP 2: MACHINE LEARNING WITHOUT BIAS

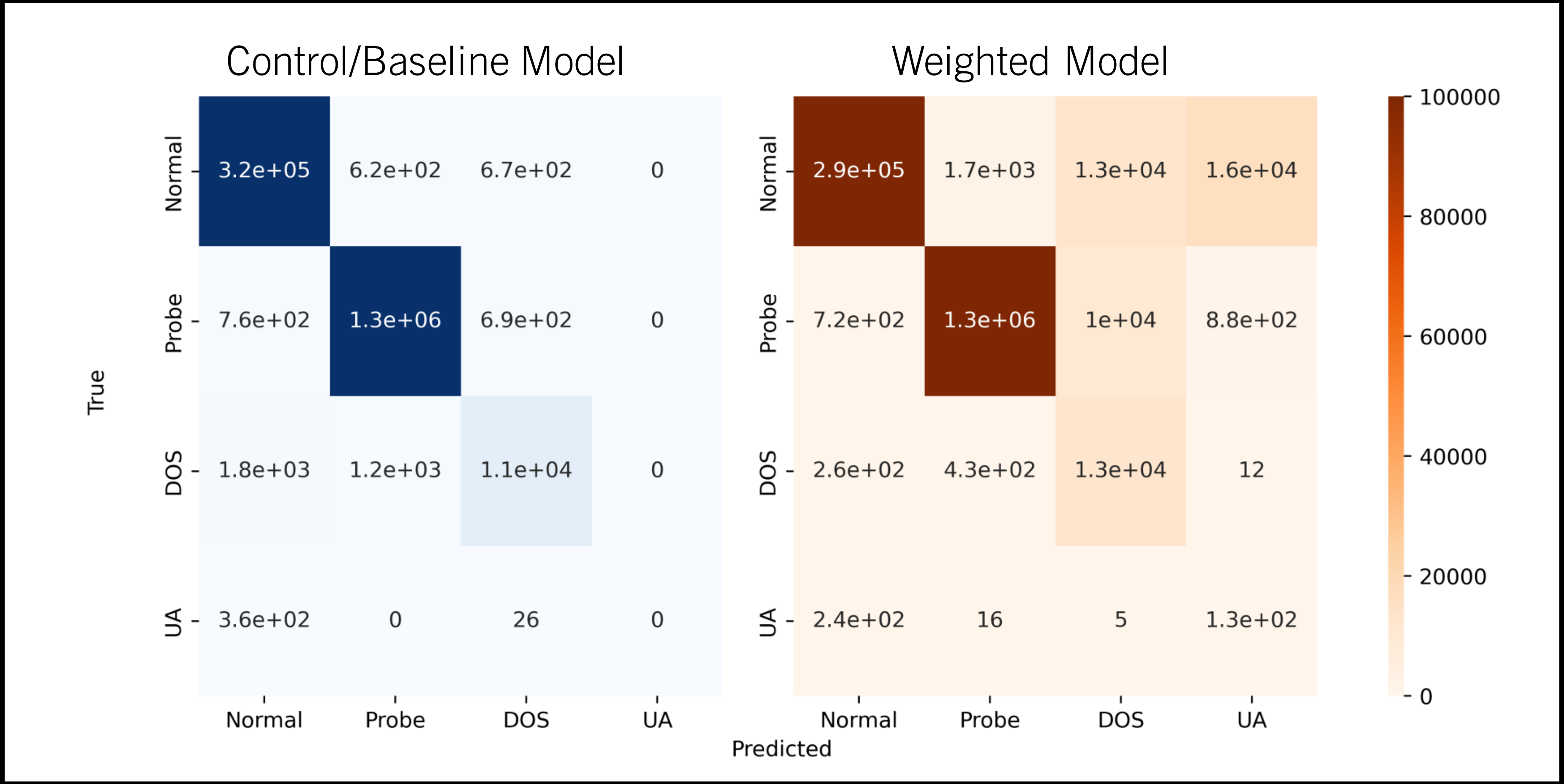
- Used special technique to change the way the model was evaluated
- Weights β were calculated such that the model would have relatively higher value of loss for classes with lower number of samples and vice versa

$$H(t,p) = -\frac{1}{N}\sum_{i=1}^{n} \beta t_i log(p_i) + (1 - t_i)(1 - \beta) log(1 - p_i) \qquad (2)$$

TESTING METHODOLOGY

- Two Machine Learning models were trained for the grouped dataset
- First one used the unweighted loss function
- Second one used the weighted loss function
- A base model on the original unprocessed dataset was also trained as a second control group

RESULTS


Control/Baseline Model
Weighted Model
True
Predicted
Normal
Probe
DOS
UA
3.2e+05
6.2e+02
6.7e+02
0
7.6e+02
1.3e+06
6.9e+02
0
1.8e+03
1.2e+03
1.1e+04
0
3.6e+02
0
26
0
2.9e+05
1.7e+03
1.3e+04
1.6e+04
7.2e+02
1.3e+06
1e+04
8.8e+02
2.6e+02
4.3e+02
1.3e+04
12
2.4e+02
16
5
1.3e+02
100000
80000
60000
40000
20000
0

METRICS EVALUATION

Class	Control Model			Weighted Model			support
	precision	recall	f1-score	precision	recall	f1-score	
Normal	0.9908	0.996	0.9934	0.9958	0.9023	0.9468	321018
probe	0.9986	0.9989	0.9987	0.9983	0.9907	0.9945	1281513
DOS	0.8842	0.7773	0.8273	0.3507	0.9482	0.512	13563
Unauthorized Access	1	0	0	0.0076	0.3368	0.0149	389
accuracy		0.9962			0.9726		0.9726
macro avg	0.9684	0.693	0.7048	0.5881	0.7945	0.617	1616483
weighted avg	0.9961	0.9962	0.996	0.9921	0.9726	0.9807	1616483

CONCLUSIONS AND FUTURE WORK

Accuracy can be deceiving

Weighted loss can be a strong method to tackle a biased dataset

An entire classification report should be preferred above score reports in ML model evaluation

Learn a meta model to analyze the result from both models to produce even stronger Intrusion Detection Systems