NEW YORK INSTITUTE OF TECHNOLOGY
Vancouver

INCS 775: Data Center Security

Assignment #2 (Group) - Applying ML to Detect DoS Attacks in Data Center Network

Posted: *June 14, 2024*
Due: *June 28, 2024*

Instructor: Dr. Zhida Li
Summer 2024 INCS, 775-VA3 (1391)
Email: zli74@nyit.edu

In this assignment, we will learn four types of intrusions and use the machine learning (ml) model(s) to detect DoS with **coding (Python) in** the **"*.ipynb*"** file**. DoS** attack and **Regular** data will be detected based on our developed ml model(s).

Tasks:

1. **Download the shared folder from Google Drive (for NYIT login):**
   https://drive.google.com/drive/folders/1L1LU_dQ5hajJd9JvQzkXDFkU8vrsy7yE?usp=sharing
   Folder name:
   ML_in_DCN
   Unzip the folder.

   Folder content:
   - Python code (Jupyter file): *INCS775_Lab2_ MLinDCN.ipynb*
   - NSL-KDD training data (42 columns): *KDDTrain+_20Percent_775.csv*
   - NSL-KDD testing data (42 columns): *KDDTest+_775.csv*
   - Webpage (NSL-KDD) offline: *nsl-kdd_index.html*
   - Paper for NSL-KDD: *A_detailed_analysis_of_the_KDD_CUP_99_data_set.pdf*

   Content of the CSV files:
   Row 1: header (feature names)
   Columns 1-41: features
   Column 42: labels for **Regular = 0, DOS = 1**, R2L = 2, U2R = 3, Probe = 4.

2. **Learn and understand your NSL-KDD data:**
   Cyber attacks are becoming more sophisticated and, hence, more difficult to detect. Using efficient and effective machine learning techniques to detect network anomalies and intrusions is an important aspect of cyber security.

Machine learning algorithms have been evaluated for robustness, high accuracy, and training time when classifying various datasets collected from communication networks. Reliable testing and validation of anomaly and intrusion detection algorithms depend on the quality of datasets such as traffic collected from deployed networks or experimental testbeds. The most widely used benchmark datasets in the literature are Knowledge Discovery in Databases (KDD) Cup 1999 (KDD'99) and NSL-KDD. KDD'99 intrusion dataset is based on the Defense Advanced Research Projects Agency (DARPA) 1998 dataset.

The NSL-KDD dataset is an improved version of the KDD'99 intrusion dataset. Data were captured from an evaluation testbed and included large numbers of virtual hosts and user automata. The KDD'99 dataset was used in various IDSs. NSL-KDD is a randomly selected subset of KDD'99 after redundant data were removed and is a widely used benchmark for evaluating anomaly detection techniques. NSL-KDD dataset captures TCP, UDP, and Internet Control Message Protocol (ICMP) traffic collected using the tcpdump utility. It contains four types of intrusion attacks: DoS, R2L, U2R, and Probe described in Table 1.

Each network connection is represented by 41 features: 38 numerical and 3 categorical ("protocol_type", "service", and "flag") features. Categorical features will be converted to numerical in our experiment.

Table 1: NSL-KDD: Four types of intrusion attacks are listed: DoS, R2L, U2R, and Probe.

Type	Intrusion attacks
DoS	back, land, neptune, pod, smurf, teardrop, mailbomb, processtable, udpstorm, apache2, worm
U2R	buffer-overflow, loadmodule, perl, rootkit, sqlattack, xterm, ps
R2L	fpt-write, guess-passwd, imap, multihop, phf, spy, warezmaster, xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named
Probe	ipsweep, nmap, portsweep, satan, mscan, saint

The features with continuous and discrete types are described in Table 2. We consider binary classification, only considering Regular, DoS. The labels in the CSV files are Regular, DoS, R2L, U2R, and Probe, labeled as 0, 1, 2, 3, and 4, respectively. We process the KDDTrain+_20Percent_775 dataset for training and KDDTest+_775 dataset for testing.

Before beginning the experiment, extract the data points (rows) that are labeled only as Regular (0) and DoS (1). Count the number of data points for Regular (0) and DoS(1). (These tasks are shown in the file *INCS775_Lab2_ MLinDCN.ipynb*)

Table 2: NSL-KDD features: Definitions, types, and descriptions. Each network connection is represented by 41 features: 38 numerical and 3 categorical ("protocol_type", "service", and "flag")

features.

Feature	Definition	Type	Description
1	duration	continuous	length (seconds) of the connection
2	protocol_type	discrete	type of the protocol (TCP, UDP)
3	service	discrete	network service on the destination, (HTTP, telnet)
4	flag	discrete	normal or error status of the connection
5	src_bytes	continuous	no. of data bytes from source to destination
6	dst_bytes	continuous	no. of data bytes from destination to source
7	land	discrete	1 if connection is from/to the same host/port; 0 otherwise
8	wrong_fragment	continuous	no. of "wrong" fragments
9	urgent	continuous	no. of urgent packets
10	hot	continuous	no. of "hot" indicators
11	num_failed_logins	continuous	no. of failed login attempts
12	logged_in	discrete	1 if successfully logged in; 0 otherwise
13	num_compromised	continuous	no. of "compromised" conditions
14	root_shell	discrete	1 if root shell is obtained; 0 otherwise
15	su_attempted	discrete	1 if "su root" command attempted; 0 otherwise
16	num_root	continuous	no. of "root" accesses
17	num_file_creations	continuous	number of file creation operations
18	num_shells	continuous	no. of shell prompts
19	num_access_files	continuous	no. of operations on access control files
20	num_outbound_cmds	continuous	no. of outbound commands in an ftp session
21	is_host_login	discrete	1 if the login belongs to the "hot" list; 0 otherwise
22	is_guest_login	discrete	1 if the login is a "guest"login; 0 otherwise
23	count	continuous	no. of connections to the same host as the current connection in the past 2 s
24	srv_count	continuous	no. of connections to the same service as the current connection in the past 2 s
25-26	serror_rate and srv_serror_rate	continuous	no. of connections that have "SYN" errors
27-28	rerror_rate and srv_rerror_rate	continuous	no. of connections that have "REJ" errors
29	same_srv_rate	continuous	no. of connections to the same service
30	diff_srv_rate	continuous	no. of connections to different services
31	srv_diff_host_rate	continuous	no. of connections to different hosts
32	dst_host_count	continuous	no. of connections to the same service as the current connection in the past 2 s
33	dst_host_srv_count	continuous	no. of connections to the same service as the current connection in the past 2 s
34	dst_host_same_srv_rate	continuous	no. of connections to the same service
35	dst_host_diff_srv_rate	continuous	no. of connections to different services
36	dst_host_same_src_port_rate	continuous	no. of connections to the same source port
37	dst_host_srv_diff_host_rate	continuous	no. of connections to different hosts
38-39	dst_host_serror_rate and dst_host_srv_serror_rate	continuous	no. of connections that have "SYN" errors no. of connections that have "SYN" errors
40-41	dst_host_rerror_rate and dst_host_srv_rerror_rate	continuous	no. of connections that have "REJ" errors

References:
[1] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Ottawa, ON, Canada, July 2009, pp. 1–6.

[2] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln laboratory," *ACM Trans. Inform. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000.

3. **Go back to Google Drive:**
   Create your own copy of the folder:
   ML_in_DCN _teamNumber
   * Note: This is very important so that you do not alter the original code.
   While in your own folder on Google Drive, upload your data, run (double click) on:
   INCS775_Lab2_ MLinDCN.ipynb

   Note:
   We assume that you do not already have the Colab web application.
   To use Colab, select "Open with", search for Application named Colaboratory, and install it. Finally, connect to the Application.

   For advanced users only:

   If you wish to run the exercise locally, you need to install Jupyter's development environment.
   Instructions are posted at:
   https://jupyter-notebook.readthedocs.io/en/stable/

   To install Jupyter, you will also need the "pip" package management system (installer) for Python:
   python get-pip.py
   Instruction is available at:
   https://jupyter.org/install
   ---

4. **You are now ready to follow the steps:**
   INCS775_Lab2_ MLinDCN.ipynb

Scientific Computing and Machine learning libraries may be used:

- https://numpy.org/: numpy (built-in)
- https://scikit-learn.org/stable/index.html: sklearn
- https://pandas.pydata.org/: pandas
- https://pytorch.org/: torch
- https://www.tensorflow.org/: tensorflow

Grading Scheme (total 10 points)

1. Code implementation: (even if not completed) (7 points)
2. Results can be regenerated based on the submission: (1 point)
3. Steps/explanations/comments: (2 points)

Please Submit:

- **Code (*.ipynb*), and please keep the output in the *.ipynb* file.**
  Include code, steps/explanations/comments in the *.ipynb* file.

- Submit the file for your project submission to the Canvas site.
  Additionally, the link to the Colab can also be provided in the comments for better grading.

- Note:
  More credit will be given to completed assignments than to ambitious unfinished work.
  In the case of teamwork, we will make every attempt to evaluate the individual performance of each team member.