IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 6, June 2024

DOI: 10.17148/IJARCCE.2024.13632

DDoS Attack Detection by Using Practical Lightweight Deep Learning Methods

Goriparthi Meenakshi[1] and L N V Rao[2]

M.Tech Student, V K R V N B & A G K College of Engineering, Gudivada, India.[1]

Associate Professor, Department of CSE, V K R V N B & A G K College of Engineering, Gudivada, India.[2]

Abstract: DDoS assaults, which interrupt the availability of services across the board, are one of the most dangerous forms of cyberattacks now in existence. The complexity of DDoS detection stems from the fact that it must analyse a large amount of real-time traffic as well as a wide variety of attack methods. In this work, we introduce LUCID, a lightweight deep learning distributed denial-of-service (DDoS) detection system that uses Convolutional Neural Networks' (CNNs) inherent features to distinguish between malicious and benign traffic flows. Specifically, we add four things to the literature: (1) a novel application of a convolutional neural network (CNN) to detect DDoS traffic with low processing overhead; (2) a dataset-agnostic pre-processing mechanism to produce traffic observations for online attack detection; (3) an activation analysis to explain LUCID's DDoS classification; and (4) an empirical validation of the solution on a resource-constrained hardware platform. When tested on the most recent data available, LUCID's detection accuracy is on par with that of the state-of-the-art methods, but its processing time is cut in half. Through our evaluations, we show that the suggested method can effectively identify DDoS attacks even in contexts where resources are few.

Keywords: Distributed Denial of Service, Deep Learning, Convolutional Neural Networks, Edge Computing.

I. INTRODUCTION

Due to the proliferation of attack vector combinations, DDoS assaults have gotten more evasive in recent years. The use of numerous protocols in a single DDoS assault, or "multi-vector attacks," is not uncommon. More complex and comprehensive defence strategies are needed to counteract the increasing variety of assault methods. Intrusion detection systems that rely on historical signatures are incapable of responding to novel threats. The current need to establish detection thresholds in statistical anomaly-based systems is a limitation. Network Intrusion Detection Systems (NIDSs) using machine learning techniques are being explored to address the limitations of existing solutions. Here, deep learning (DL) algorithms have been shown to be very successful in separating DDoS activity from benign traffic by deriving high-level feature representations of the traffic from low-level, granular properties of packets [4, 5]. However, many of the DL-based techniques reported in the literature are impractical because to their high training costs and lack of realism. Detection algorithms must handle traffic flows that might be separated over numerous capture time windows, which is a challenge in a real-world network and which existing solutions do not address. The rapid rise in popularity of Convolutional Neural Networks (CNNs) is attributable to the significant advances they have brought to fields as diverse as computer vision [6]– [8], NLP [9], machine vibration analysis [12], and medical signal processing [13]. While research on CNNs is still limited, their implementation has pushed the state of the art forward in a number of specific cybersecurity use cases, including malware detection [14–17], code analysis [18], network traffic analysis [4] and intrusion detection in industrial control systems. Motivated by these accomplishments and the advantages of CNN in terms of less time spent on feature building and better detection accuracy, we have begun using CNNs in our research.

II. LITERATURESURVEY

Low-rate Distributed Denial-of-Service attacks are the most harmful kind of cyberattack. We explore two novel methods for detecting low-rate DDOS attacks: the generalised entropy and information distance techniques. It is difficult to maintain network security due to threats such as distributed denial of service (DDoS) assaults and OpenFlow DDOS flooding attack. Armies are prepared to launch an assault in this scenario. However, many of the DL-based techniques reported in the literature are impractical because to their high training costs and lack of realism. Detection algorithms must handle traffic flows that might be separated over numerous capture time windows, which is a challenge in a real-world network and which existing solutions do not address.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 6, June 2024
DOI: 10.17148/IJARCCE.2024.13632

The rapid rise in popularity of Convolutional Neural Networks (CNNs) is attributable to the significant advances they've brought to fields as diverse as computer vision, Natural Language Processing, protein binding prediction, machine vibration analysis, and medical signal processing. CNNs have enhanced the state of the art in particular situations, such as malware identification, code analysis, network traffic analysis, and intrusion detection in industrial control systems, but their usage is still under-researched in cybersecurity as a whole. These victories, together with CNN's advantages in terms of less time spent on feature building and better detection accuracy, convince us to use the tool.

III. PROBLEM STATEMENT

Models have a high false positive rate and cover a narrow spectrum of possible assaults. Due to previous research mostly relying on a single dataset to describe the success of the machine learning model, the models cannot be generalised. A DL-based DDoS detection architecture well suited for online resource-constrained situations, which leverages CNNs to learn the behaviour of DDoS and benign traffic flows with minimal processing overhead and attack detection time, has not been explored before. Our model is denoted as LUCID (Lightweight, Usable CNN in DDoS Detection). This paper proposes a dataset-agnostic preprocessing technique that generates traffic observations compatible with those gathered in preexisting live systems, where detection algorithms must deal with segments of traffic flows recorded across predetermined time intervals. In order to decipher and clarify which traits contribute LUCID places significance on DDoS categorization while making decisions.

Existing systems have not widely adopted solutions for network intrusion detection because of difficulties such as high error cost, traffic fluctuation, and others. Over the past few years, there has been a steady rise in the availability of realistic network traffic data sets and an uptick in collaboration between data scientists and network researchers to enhance model explain ability, paving the way for the development of more effective Machine Learning (ML) solutions for network attack detection. In particular, there is a decline in accuracy when trying to differentiate between benign and malicious communications. When evaluating an existing system, neither the detection time nor the datasets utilised are disclosed. The current system's outcomes demonstrate the wide range of ML models' accuracy and performance.

3.1 Algorithm

For the objective of identifying instances of online violence, we feed the results of algorithm 1 into a cnn model. Lucid categorises incoming network activity as either malicious (ddos) or benign. Our objective is to streamline this cnn model so that it can run quickly and efficiently on low-powered devices. The suggested approach utilises a lightweight, supervised detection machine with a cnn, similar to [9] in the field of natural language processing, to accomplish this goal. In contrast to traditional neural networks, where each weight is utilised optimally only once, Cnns contain reusable and shareable parameters for their kernel weights. Our model's storage and memory needs are reduced as a result.

Max pooling layer: For Max pooling, we use a down-pattern along the initial measurement of a to capture the temporal character of the entrance. With a pool length of m, the most relevant m activations of each learnt clear out are included in an output matrix mo of dimension (n - h + 1) = m x okay, with mo = [max(a1)j:::jmax(ak)]. In this way, the version gives more weight to the more significant activations and ignores the less advantageous facts that resulted in lower activations. By doing so, we may reduce the network's complexity and get a more compact characteristic encoding by excluding information regarding the spatial facts of the activation, i.e. where it occurred within the original signal. Rearranged such that the final input to the category layer is a one-dimensional function vector v.

IV. PROPOSEDSYSTEM

Object Oriented Analysis and Design

The line between OOA and OOD blurs when object orientation is employed for both analysis and design. In particular, this is true for approaches that merge the phases of research and development. The essential constructions (i.e., objects and classes) employed in OOA and OOD are similar, which contributes to the fuzziness. While opinions vary on which steps in the object-oriented development process fall within the purview of analysis and which should be left to design, there is consensus on the areas in which each should be focused.

In contrast to OOD, which models the solution to the issue, OOA models the problem domain, which yields an understanding and definition of the problem. The difference between analysis and design is that the latter focuses on the domain of the solution to the issue. However, with OOAD, this is all rolled into a single representation of the solution domain.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 6, June 2024

DOI: 10.17148/IJARCCE.2024.13632

That is, most of the representation built by OOA is often included in the solution domain representation built by OOD. The dividing line is subjective, and its location is open to several interpretations. The "seamless" nature of the transition from analysis to design is often cited as a strength of the object-oriented methodology. That's also why a lot of people choose OOAD approaches, in which development of both the architecture and the software is carried out simultaneously.

Object-oriented analysis and design (OOA/OOD) vary primarily in terms of the kind of objects that result from the process of analysis and design, since the two fields of study use distinct modelling paradigms.

Features of OOAD:
Individuals that utilise it The application's building blocks are objects, not functions.
All things, and their connections to one another, may be visually depicted.
The system's key users, or KPs, will be modelled as actors, with their activities captured in use cases.
Sequence diagrams are useful for depicting the overall flow of a use case, while Activity diagrams and status chart diagrams are useful for depicting the individual steps in the use case's workflow.

The Genesis of UML
The field of software engineering is becoming pervasive. Most of our day-to-day activities rely on software, and as time goes on, this software becomes more complicated and expensive across the board, from washing machines to compact disc players to cash machines and phones.

Development teams face more and more pressure as the demand for complex software grows. Applications, dispersed and heterogeneous settings, programme size, team structure, end-user ergonomic expectations, and other factors are all contributing to a world of increasing complexity for software developers.

Software engineers will need communication skills, both for explaining their own work and for comprehending the work of others, in order to succeed in this field. Because of this, their demand for approaches is growing (and will continue to grow).

From Functional to Object – Oriented Methods
Though object-oriented approaches may be traced back to the 1960s, structured and functional approaches came earlier. Since functional approaches are heavily influenced by computer architecture, this is not surprising (a proven domain well known to computer scientists). Computer scientists learned to conceptualise in terms of system functions by adopting a mindset that translates the physical separation of data and code into the methodologies.

This method makes sense when placed in its historical perspective, but is hopelessly out of date in the present day due to its lack of abstraction. You shouldn't force a software solution to work with certain hardware. Instead of dictating the hardware's architecture, hardware should be the software's servant.

Towards A Unified Modelling Language
The examination of the many ideas offered by previous approaches made the unification of object-oriented modelling methods feasible with the accumulation of expertise. When Jim Rumbaugh and Grady Booch saw that the gaps between the many ways were narrowing and the method wars were no longer advancing object-oriented technology, they decided towards the end of 1994 to consolidate their efforts into a single framework they called the Unified Method. Ivar Jacobson, the creator of use cases, an effective method for eliciting needs, joined the team a year later.

Booch, Rumbaugh and Jacobson adopted four goals:
Aiming to use object-oriented ideas to depict whole systems rather than just the software component.
With the goal of creating a direct connection between ideas and actual code.
In order to factor in the scaling characteristics of complicated and crucial systems.
That is, to develop a modelling language that can be understood and used by computers as well as people.

SDLC Models
For developing modest-sized to moderate-sized database applications, this is the SDLC.

Components of the application are produced in rapid succession using the iterative development lifecycle used for this project. Initial iteration focuses on core functionality, whereas future iterations build upon that work by adding new features or fixing bugs discovered in production.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 6, June 2024
DOI: 10.17148/IJARCCE.2024.13632

The SDLC's six phases are meant to build upon one another, with each phase using the preceding one's outputs to guide its own work and yielding outcomes that both benefit from and can be traced back to those that came before. Additional data is collected or created at each stage, which is then integrated with the inputs to create the stage's outputs. New ideas that might take the project in areas not expected by the original set of high-level requirements or features that are out-of-scope are retained for later consideration, and the extra information is limited in scope.

When both the development team and the customer's staff become excited about the potential of automation, the software development project often goes off the rails. Nice-to-have features are appealing, but they aren't necessary to address the issue at hand, which might lead to the team losing sight of the most important features. This is the major reason the development team employs the iterative approach, since it addresses the underlying cause of a significant proportion of unsuccessful and/or abandoned development endeavours.

Roles and Responsibilities of PDR AND PER

Two crucial responsibilities, defined by the iterative lifecycle, work together to convey project challenges and ideas from the end-user community to the development team in a way that is both understandable and actionable.

Primary End-user Representative (PER)

The major end-user contact and key decision maker is the principle endorser (PER). The PER's other duty is to coordinate the timely execution of end-user reviews by qualified specialists in the field.

PER-PDR Relationship

The PDR and the PER are the development effort's think tank. The PER is well-versed in the application's domain and has strong relationships with other members of the end-user community, allowing them to fully grasp the challenges of the business processes that the application is meant to facilitate. When the PDR and the rest of the development team work together, they serve as hubs for the dissemination and consolidation of information about the application under development.

In essence, this is an extension of the "pair programming" idea from Agile techniques to the end-user community, with the goal of creating the tight connection that is typical of software projects involving a single developer and a single end-user. While it may be challenging to forge strong bonds between the many members of an end-user community and a software development team, doing so amongst the respective groups' primary spokespeople is often a breeze.

As the number of end-users and developers on a project rises, so does the difficulty of communication between the two groups. The PER and PDR are kept in the loop on all communications between the end-user community and the development team. This helps them resolve conflicts that may arise, such as when two different end-users communicate different requirements for the same application feature to different members of the development team.

Input Design

System design, of which input is a component, is an umbrella term. This section outlines the primary goals of the input design process:
To develop an efficient means of input at a low cost.
Optimal precision is desired.
For the sake of user satisfaction and proper comprehension, the input must be correct and understandable.

Input Stages

The primary inputs required before data may be saved in a database:
For instance, under this project, voters' information, both fresh and old, will be recorded in a database.
Information gathering, Information transcription, The Processing and Verification of Data
Command and control, data transfer, data verification, data rectification

Output Design

The primary function of computer system outputs is to provide processed results to end users. The findings may be consulted again in the future, and a permanent copy is provided for this purpose. Different kinds of outputs include:
The products created for consumption outside of the company are called "external outputs."

Internal outputs are those whose final destination is inside the organisation, and they serve as the primary means through which users interact with the system.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 6, June 2024

DOI: 10.17148/IJARCCE.2024.13632

A product of an operation that has little value outside of the IT department.
Outputs from the interface that need user participation in order to convey information.
There was a requirement for both printed reports and online querying of the data. The output format is derived from the outputs that are presently being acquired via laborious human processing to ensure accuracy. Whenever possible, please use a normal printer to produce physical copies.

V. RESULTS

Implementation
The term "implementation" is often used to refer to the steps taken to put into action a newly designed or changed system. The three varieties of Implementations are as follows:

Automation of a formerly manual process via the use of computers. File conversion, user education, and printout integrity verification are all issues that arise.

Replacement of an old computer system with a new one. In most cases, this is a challenging adjustment to make. Issues might arise if preparations are sloppy.

A new version of a programme is installed on the same machine to replace an older one. As long as there are no big modifications to the files, this sort of conversion is straightforward to manage.

All modules of the Generic tool project have been implemented. The first step is to authenticate the user. This section of code is responsible for generating a session for the user and verifying that they are an authorised user of the database. Any and all illegal activity is to be avoided.

The tables made in the Table creation module include fields chosen by the user, and the user may generate many tables simultaneously. The production of tables may be contingent on their stipulating requirements, limits, and computations. The needs of the end-users are consistently met by the Generic code.

A user may modify an existing record, remove it entirely, or add it as a new entry in the database through the Updating module. This is a crucial part of the Generic code project's overall architecture. User input is required for field values, after which the Generic tool will provide complete field values pertinent to the given record.

The reports from the database may be seen in either a two- or three-dimensional format in the Reporting module. When the user chooses the table and enters the condition, a report is automatically created. The following figures shows the screen shots of the results obtained.

```
import numpy as np
import pandas as pd
import os
import gc
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
#import libraries
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc
from sklearn import metrics
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation,Dropout
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.optimizers import SGD,Adam
import keras
# Conv1D + LSTM
from keras.layers.convolutional import Conv1D,MaxPooling1D
from keras.layers import LSTM
from keras.layers import Dense,Dropout
from keras.layers import Flatten
```

Figure 1:

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 6, June 2024

DOI: 10.17148/IJARCCE.2024.13632

	sport	dport	pkts	bytes	state	dur	mean	stddev	sum	min	max	spkts	dpkts	sbytes	dbytes	rate	srate	drate
0	3456	80	10242	9972888	CON	75.066856	4.989200	0.006723	74.837997	4.968420	4.999928	5124	5118	4877519	5095369	136.425049	68.245834	68.165901
1	8080	80	10246	9978140	CON	75.066925	4.989192	0.006743	74.837875	4.968407	4.999997	5122	5124	5153910	4824230	136.478210	68.219124	68.245766
2	80	80	5116	4911913	INT	75.063446	4.990958	0.005609	74.864372	4.985135	4.999931	5116	0	4911913	0	68.142357	68.142357	0.000000
3	80	80	5117	4789010	CON	75.063515	4.990964	0.003873	74.864456	4.985136	4.999931	5117	0	4789010	0	68.155617	68.155617	0.000000
4	365	565	2753	165180	INT	75.240524	4.954834	0.048765	74.322510	4.826530	4.997365	2753	0	165180	0	36.576035	36.576035	0.000000

Figure 2

Feature Scaling

```
[ ] # Normalization OR Standardization
    def standardize(df,col):
        df[col]= (df[col]-df[col].mean())/(df[col].std()) # Standardization
        #df[col]= (df[col]-df[col].min())/(df[col].max()-df[col].min()) #Normalization

    for i in (df_num.iloc[:,:-1].columns):
        standardize (df_num,i)

    df_num.head()
```

	sport	dport	pkts	bytes	state	dur	mean	stddev	sum	min	max	spkts	dpkts	sbytes	dbyte
0	-1.117768	-0.009047	136.295724	140.707255	-6.113447	54.380232	64.382420	-0.607463	82.504517	82.378869	46.009199	102.505273	157.980595	107.722945	162.37027
1	-0.243746	-0.009047	136.348979	140.781363	-6.113447	54.380290	64.382315	-0.606978	82.504382	82.378653	46.009843	102.465236	158.165841	113.828228	153.72966
2	-1.755895	-0.009047	68.050576	69.294412	-3.872036	54.377382	64.405280	-0.634490	82.533627	82.656359	46.009227	102.345125	-0.034879	108.482685	-0.00809
3	-1.755895	-0.009047	68.063890	67.560190	-6.113447	54.377439	64.405358	-0.676607	82.533720	82.656375	46.009227	102.365143	-0.034879	105.767844	-0.00809
4	-1.702025	11.523014	36.590708	2.315678	-3.872036	54.525396	63.935544	0.412512	81.935571	80.023319	45.985281	55.041297	-0.034879	3.630691	-0.00809

Figure 3

Training Curves

```
[ ] plt.figure(figsize=(15,5))
    plt.plot(range(history.epoch[-1]+1),history.history['val_auc'],label='val_auc')
    plt.plot(range(history.epoch[-1]+1),history.history['auc'],label='auc')
    plt.title('auc'); plt.xlabel('Epoch'); plt.ylabel('auc');plt.legend();
    plt.show()
```

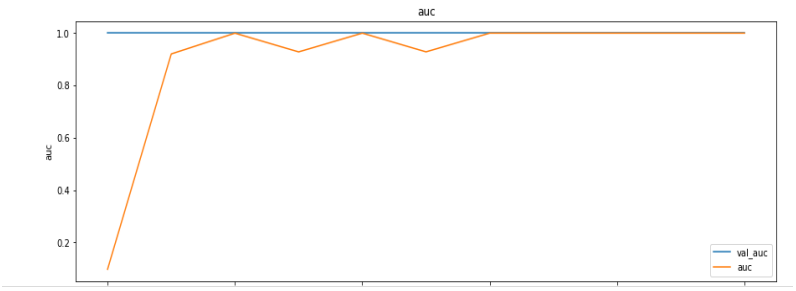

Figure 4

```
[ ] plt.figure(figsize=(15,5))
    plt.plot(range(history.epoch[-1]+1),history.history['val_loss'],label='Val_loss')
    plt.plot(range(history.epoch[-1]+1),history.history['loss'],label='loss')
    plt.title('loss'); plt.xlabel('Epoch'); plt.ylabel('loss');plt.legend();
    plt.show()
```

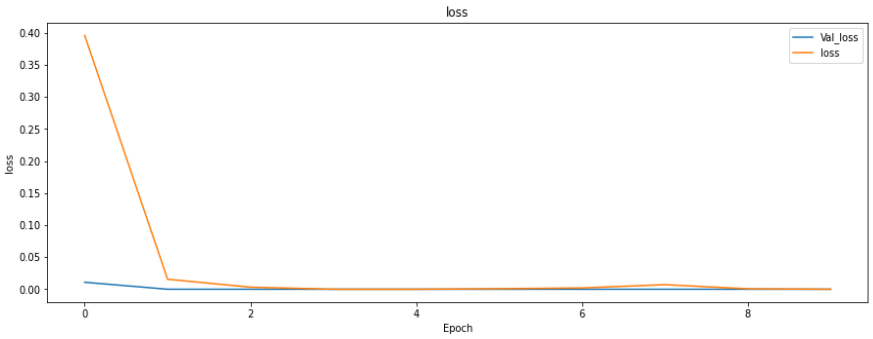

Figure 5

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 6, June 2024

DOI: 10.17148/IJARCCE.2024.13632

Build CNN Model

```
[ ]  learning_rate=0.001
     batch_size=5000
     epochs = 15

     model_save = ModelCheckpoint('./DDoS_TCP.h5',
                                  save_best_only = True,
                                  save_weights_only = True,
                                  monitor = 'val_loss',
                                  mode = 'min', verbose = 1)
     early_stop = EarlyStopping(monitor = 'val_loss', min_delta = 0.0001,
                                patience = 8, mode = 'min', verbose = 1,
                                restore_best_weights = True)
     reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.6,
                                   patience = 4, min_delta = 0.0001,
                                   mode = 'min', verbose = 1)
```

Figure 6

```
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.legend(loc="lower right")
    plt.show()

generate_results(y_test, y_pred)
```

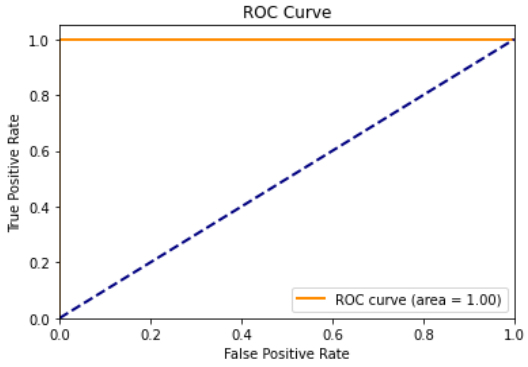

Figure 7

VI. CONCLUSIONS

Distributed denial of service attacks continue to be a major problem that limits network uptime everywhere. As part of this effort, we have developed a scented CNN-based DDoS detection framework. We have aimed for efficiency and speed in both processing and assault detection with our approach. The CNN model has advantages over other ML methods since it eliminates the need for threshold setting, which is necessary for statistical detection approaches, and it requires less feature engineering and dependence on human specialists. To put it another way, this facilitates deployment in a realistic manner. In contrast to other available options, our novel traffic pre-processing approach takes into account the interdependence of various network nodes and is tailored to feed network traffic into a convolutional neural network model for real-time detection of distributed denial of service attacks.

When compared to the current state-of-the-art, LUCID performs just as well, as shown by our evaluation findings. However, unlike prior work, we show that our method is stable by consistently detecting targets across a variety of datasets. In addition, the results of our study on a device with limited resources show that our approach may be successfully deployed even in such circumstances. In particular, we show that our method significantly outperforms comparable state-of-the-art systems in terms of processing time (by a factor of 40). Finally, we have included inactivation analysis, which is absent from prior publications, to explain how LUCID learns to detect traffic.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 6, June 2024
DOI: 10.17148/IJARCCE.2024.13632

VII. FUTURE SCOPE

When compared to the current state-of-the-art, LUCID performs just as well, as shown by our evaluation findings. However, unlike prior work, we show that our method is stable by consistently detecting targets across a variety of datasets. In addition, the results of our study on a device with limited resources show that our approach may be successfully deployed even in such circumstances. In particular, we show that our method significantly outperforms comparable state-of-the-art systems in terms of processing time (by a factor of 40). Finally, we have included inactivation analysis to explain how LUCID learns to identify DDoS traffic, which is absent from prior publications.

REFERENCES

[1]. X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS Attack via Deep Learning," in Proc. of SMARTCOMP, 2017.
[2]. M. Ghanbari and W. Kinsner, "Extracting Features from Both the Input and the Output of a Convolutional Neural Network to Detect Distributed Denial of Service Attacks," in Proc. of ICCI*CC, 2018.
[3]. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015.
[4]. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, 2012, pp. 1097–1105.
[5]. M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deepanomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes," Computer Vision and Image Understanding, vol.172, pp. 88 – 97, 2018.
[6]. Y. Kim, "Convolutional neural networks for sentence classification," in Proc. of EMNLP, 2014.
[7]. B. Alipanahi, A. Delong, M. Weirauch, and B. J Frey, "Predicting the sequence specificities of dna- and rna-binding proteins by deep learning," Nature biotechnology, vol. 33, 07 2015.
[8]. D. Quang and X. Xie, "DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences," Nucleic Acids Research, vol. 44, no. 11, pp. e107–e107, 2016.
[9]. O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccufier, S. Verstockt, R. V. de Walle, and S. V. Hoecke, "Convolutional Neural Network Based Fault Detection for Rotating Machinery," Journal of Sound and Vibration, vol. 377, pp. 331 – 345, 2016.
[10]. A. Vilamala, K. H. Madsen, and L. K. Hansen, "Deep Convolutional Neural Networks for Interpretable Analysis of EEG Sleep Stage Scoring," Proc. of MLSP, 2017.
[11]. N. McLaughlin, J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doup´e, and G. Joon Ahn, "Deep android malware detection," in Proc. of CODASPY, 2017.
[12]. T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 773–788, March 2019.
[13]. Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng, "Malware traffic classification using convolutional neural network for representation learning," in Proc. of ICOIN, 2017.
[14]. M. Yeo, Y. Koo, Y. Yoon, T. Hwang, J. Ryu, J. Song, and C. Park, "Flow-based malware detection using convolutional neural network," in Proc. of International Conference on Information Networking, 2018.
[15]. R. Russell, L. Kim, L. Hamilton, T. Lazovich, J. Harer, O. Ozdemir, P. Ellingwood, and M. McConley, "Automated Vulnerability Detection in Source Code Using Deep Representation Learning," in Proc. of ICMLA,2018.
[16]. K. Wu, Z. Chen, and W. Li, "A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks," IEEE Access, vol. 6, pp. 50 850–50 859, 2018.
[17]. Krebs on Security, "DDoS on Dyn Impacts Twitter, Spotify, Reddit," https://krebsonsecurity.com/2016/10/ddos-on-dyn-impacts-twitterspotify-reddit, 2016.
[18]. Radware, "Memcached DDoS Attacks," https://security.radware.com ddos-threats-attacks/threat-advisories-attack-reports/memcached-underattack/, 2018.