LUPINE PUBLISHERS

Current Trends in Computer Sciences & Applications

DOI: 10.32474/CTCSA.2023.02.000147

ISSN: 2643-6744

Research Article

Real-time Detection Technique for Identifying Encrypted Attacks in TLS Traffic

Asia Othman Aljahdali*, Aisha Saleh, Rawan Alrifaie, Samar Alfaifi and Ghadeer Moqbel

Department of Cybersecurity, University of Jeddah, Saudi Arabia

***Corresponding author:** Asia Othman Aljahdali, Department of Cybersecurity, University of Jeddah, Saudi Arabia

Received: June 03, 2023 **Published:** June 19, 2023

Abstract

Nowadays, most internet traffic is encrypted, and the adversary makes use of this technology to evade detection by encrypting attacks and attack signatures. This study examines multiple machine learning and deep learning models to detect and identify encrypted attacks with the minimum number of false positives and negatives. The study employs deep and machine learning models such as random forest, neural network, logistic regression, linear regression, and C4.5. These models were tested using two sets of features: extracted features and selected features, to determine the best model and set of features for detecting the encrypted attack traffic. To evaluate these models and then select the best model, several metrics have been used, including accuracy, false negative rate, and false positive rate. Finally, we developed an IDS interface and evaluated its viability using the best model and set of features.

Keywords: Network security; encrypted attacks; detection; machine learning; deep learning

Introduction

Cyberattacks happen when a third-party gains unauthorized access to a system or network. A hacker or attacker is someone who conducts a cyberattack. Cyberattacks have a number of detrimental repercussions. When an attack is conducted, it may result in data breaches, which may cause data loss or manipulation. Companies suffer financial losses, a decrease in customer trust, and reputational harm. The purpose of cybersecurity is to prevent cyberattacks, protecting networks, computer systems, and their constituent parts from illegal digital access. In today's digital environment, cyberattacks are more prevalent than ever. They can seriously harm people, companies, and governments. Cyberattacks are carried out for a variety of objectives, including monetary gain, espionage, activism, and sabotage [1]. Hackers may also conduct assaults only as a test of their mettle or for the challenge. Cyberattacks come in a wide variety and are commonplace today. Knowing the different forms of cyberattacks makes it simpler for us to defend our systems and networks against them.

Here, we'll take a detailed look at six cyberattacks that, depending on their scope, could harm either a small business or an individual. Starting with the various categories of cyberattacks on our list: A denial-of-service (Dos) attack poses a serious risk to businesses. In this case, attackers target systems, servers, or networks and bombard them with traffic to deplete their bandwidth and resources. When this occurs, the servers get overburdened with serving incoming requests, which causes the website it hosts to either go down or slow down. As a result, valid service requests go unattended. When attackers employ numerous hacked systems to initiate this attack, it is sometimes referred to as a DDoS (distributed denial-of-service) attack. A botnet, sometimes known as "bots," is a network of compromised devices that is run either by a single attacker or by a group of attackers. These bots are capable of attacking the operating systems of mobile phones and other internet-connected devices.

Next are web attacks, which aim at websites and include file inclusion, SQL injection, and cross-site scripting (XSS). Hackers frequently employ a port scan attack approach to find gaps or weak

spots in a network. Cybercriminals can use a port scan attack to identify open ports and determine if they are accepting or rejecting data. Additionally, it can demonstrate whether active security tools like firewalls are being utilized by an organization. The response that hackers get from a port when they send a message to it tells them whether the port is in use and whether it has any vulnerabilities that might be exploited. With the port scanning technique, businesses can also send packets to particular ports and examine the responses for any potential vulnerabilities. To make sure their network and systems are safe, they can utilize tools like IP scanning, network mapper (Nmap), and Netcat. Port scanning can reveal details like the services that are active consumers of services, if anonymous logins are permitted, and what network services need to be authenticated. Brute Force Attack, where a hacker can gain access to a system without authorization by trying multiple passwords until they find the right one. It has a great deal of potential for weak passwords.

Attacks that employ encryption to evade detection are referred to as encrypted attacks. Malware, ransomware, spear-phishing, zero-day attacks, data exfiltration, rogue websites, and other attack types are among them. There are several encryption methods, just as there are numerous ways for attackers to send encrypted attacks. A certificate vulnerability is one sort of encryption threat in which a certain website's security certification is deficient. This is typically indicated by an alert in your browser. In another attack, the attacker hides itself from detection by enclosing all of its communications in an encrypted tunnel. Another involves intercepting encrypted traffic and using man-in-the-middle attacks to get around encryption. Hackers use this attack type to steal information or intercept emails [2].

Recent Network Traffic Analysis National Testing Agency (NTA) research has begun to produce security intelligence by mining large volumes of network traffic. In terms of cybersecurity, the National Testing Agency (NTA) has become essential for attack detection, quality of service management, data protection, and other functions. A few things that make this problem worse are the rising number of connected devices; accelerating network speeds; and increasingly sophisticated malicious software. Over time, NTA has evolved from port-based methods to strategies based on machine learning. Since modern applications have shifted to using dynamic port allocation, port-based techniques have become unreliable. Secondly, as the volume of encrypted communication grows, payload analysis, or Deep Packet Inspection (DPI), has lost its effectiveness. Additionally, Deep Packet Inspection (DPI) is ineffective for real-time network traffic analysis because it cannot keep up with the network speed while evaluating packet payloads. Therefore, there is an urgent need for a lightweight, quick, and more accurate attack detection system [3]. Attack is created in such a way that it alters itself periodically in order to avoid being easily discovered. Attack authors constantly strive to create programs that are difficult to detect. With the passage of time, they have made successful improvements to the methods used to disguise or transform malicious code. These ideas begin with straightforward encryption before moving on to viruses that are oligomorphic, polymorphic, and metamorphic. However, with the growth of attack generation techniques, attack detectors use a variety more techniques to prevent these software's harmful impacts [3].

All current malware applications tend to encrypt their signatures to avoid detection by any antivirus software; malware detection through standard signature-based methods is becoming increasingly difficult. Recent encrypted malware has been expertly and intricately built to attack the target.

The main purpose of this study is to document a method to detect encrypted malware. Additionally, this research investigates malware detection strategies while examining malware that is encrypted, oligomorphic, polymorphic, or metamorphic in order to evade detection. We will test and try many models and sets of extracted features to finally propose a tool that is effective and accurate in detecting the encrypted malware, as well as validating and evaluating our selected model, presenting key features and limitations, and giving some recommendations.

Obfuscation Techniques

An encrypted attack is one of the most serious threats to the enterprise and can result in monetary losses, reputational harm, service interruptions, and data breaches. The fact that your consumers could download dangerous malware whenever they visit an infected website or open a malicious attachment in a phishing email only serves to exacerbate the issue. Attacks are usually sent through phishing emails in encrypted form, which enables them to pass through your traffic undetected, and inexpensive HTTPS certificates make it simpler for attackers to install attacks. Using encryption, attackers can bypass the majority of inspection devices and deliver attacks inside the network. Additionally, encrypted data exfiltration avoids detection by security systems. Threat analysis reveals that 71% of attacks encrypt their communications with command-and-control centers in order to remain undetected. Furthermore, 95% of phishing sites and 57% of attack websites were only browsed once, and their encryption hinders incident response investigations [4]. Organizations struggle to decrypt and examine traffic, which is something that cybercriminals take advantage of. Encrypted attacks infiltrate users, networks, and apps to steal personal data using attacks including spyware, ransomware, and rootkits.

Oligomorphic

The oligomorphic approach encrypts and decrypts the attack payload using various keys. It is typically employed by computer viruses to create a decryptor for themselves in a manner comparable to a simple polymorphic code. This is accomplished by choosing each decryptor component at random from a range of predetermined options. The components of a decryptor are typically too common to be identified by signatures. However, as the majority of oligomorphic viruses can only produce a few hundred distinct decryptors, it is still feasible to identify them using simple

signatures. An attack that uses oligomorphic techniques is harder to find than an attack that employs encryption [5].

Polymorphic

The polymorphic approach encrypts and decrypts data using different keys. Each copy of the worm has a new code that is generated on the fly utilizing functionally comparable instructions and encryption techniques, in addition to changing their code for each replication to evade discovery. Random number generators are employed in the implementation of polymorphic mechanisms. As a result of the virus body's encryption and the decryption routine's variations with each infection, polymorphic attacks are more challenging to identify [5].

Metamorphic

The metamorphic method does not use encryption. Instead, every time the malicious process is run, they entirely reinvent themselves. Such viruses are intelligent and carry out their operations via metamorphic engines. A metamorphic program changes itself. It is translated into temporary code (a new variant of the same virus but with a different code) and then converted back into the original code. Because every new copy of this attack has a completely unique signature, it is incredibly challenging to identify. Viruses that metamorphose are more potent than viruses that are polymorphic [5].

Stealth

The stealth approach, also known as code protection, employs a series of countermeasures to stop it from being properly analyzed. By actively modifying and corrupting the service call interrupts while it is operating, it attempts to evade detection systems. To avoid being detected by software, some infections present false information. For instance, a stealth virus hides the processes it affects and presents bogus information. As a result, it commandeers parts of the target system and conceals the infection code. A stealth virus avoids detection by antivirus software by altering the file's original size or temporarily inserting a copy of itself on another system drive, which replaces the infected file on the hard disk with the clean version. A stealth virus also conceals the adjustments it makes. It takes control of the system's functions that read files or system sectors. When a different program asks for data that the virus has already altered, the stealth virus instead reports that data to the requesting program. Memory also houses this pathogen. These viruses always hijack system processes and utilize them as a cover for their presence in order to avoid detection. The rootkit is one of the carriers of stealth viruses [6].

Packaging

Packaging is an obfuscation technique to compress attacks in order to avoid discovery, or the actual code can be hidden using encryption. This method makes it simple for attacks to get past firewalls and anti-virus protection. Before analysis, packaged attacks need to be unpacked. Compressors, crypters, protectors, and bundlers comprise the four different categories of packers [7,8].

Description of Attack Detection Techniques

Signature-Based

The fastest and most accurate method of detecting known attacks is signature-based detection. Byte sequences, assembly instructions, strings, opcodes, and lists of Dynamic Link Libraries (DLLs) are just a few of the static features that are used while creating signatures. Since it reduces overhead and execution time, the signature detection schema has been in use for a long time. However, it is unable to recognize new generations of attacks, is susceptible to polymorphism and obfuscation, and neglects to pick up certain features. A feature selection step can be introduced, and it is dynamic. Features can be employed to avoid obfuscation, and new technologies like deep learning, active learning, and machine learning can be leveraged to boost detection rates in order to create an efficient signature-based detection schema [9].

Behavior-Based

Attack functioning is ascertained using a behavior-based detection technique. Therefore, even if the attack instruction sequence and signature change, the functionality will largely remain the same. As a result, it is capable of detecting both new attacks and attacks with multiple variations. It is also effective against polymorphism and obfuscation approaches. However, it generates lots of FPs. Additionally, it might be challenging to group certain behaviors because they occur in both attack and benign samples. Additionally, some attacks do not run in a protected environment and are wrongly labeled as benign. Multiple execution pathways can be acquired using various cloud-based computers in order to accurately characterize all actions. This can reduce the number of attacks that are inadvertently classified as benign [9].

Heuristic-Based

The heuristic-based detection strategy can make use of both static and dynamic information, including Application Programming Interface (API) calls, Opcode, n-grams, lists of Dynamic Link Libraries (DLLs), and hybrid features. This detection method is sophisticated and can detect certain previously unidentified attacks; however, it is susceptible to metamorphic approaches and has a lot of rules and training steps. The performance of the approach can be enhanced by reducing the number of rules and creating a more effective learning phase.

Model Checking-Based

The model checking-based approach is effective, capable of detecting unidentified attacks, and resistant to polymorphic and obfuscation tactics. It can only see a small portion of the attack, is vulnerable to evasion tactics, and is unable to recognize all attacks of the most recent generation. Using an efficient model checker could enhance performance by identifying more precise formulas.

Deep Learning-Based: Deep learning is the foundation for the suggested attack identification for software piracy. Different kinds of source codes are used to record plagiarism-based detection. The original software's logic is carried over into the pirated edition. The

source code is tokenized after the traffic data has been identified as being related to software theft in order to reduce the size of the data. [10] Deep Learning originates from Artificial Neural Networks (ANNs) and is a subfield of ML. Large collections of labeled data are used to train deep learning systems, and neural network designs are used to extract features automatically from the data without the need for manual feature extraction. [11] It is a novel approach that is widely utilized in voice control, autonomous cars, and image processing, but it is not employed enough in attack detection. Although it is very efficient and dramatically reduces feature space, referencing the literature review of Aslan and Samet [12] it is not immune to evasion attacks. Additionally, its implementation requires time, and the model's performance was not the best. Since deep learning-based attack detection is still in its infancy, further research must be done to categorize this approach more accurately.

Mobile Device-Based

Many attack detection techniques, particularly for the Android platform, have been proposed for mobile devices. These techniques typically employ data mining and ML algorithms to find attacks. Despite recent research showing improvements in traditional and new generation attack detection for mobile devices, mobile device detection-based approaches must be improved to detect modern attacks, even if they appear to be effective when detecting old attacks. Additionally, it cannot handle a huge bundle of programs. More research is required in this area to close the gaps because attack detection in mobile space is still in its early stages [12].

Related Works for Encrypted Attack Detection

Jiyuan Liu, et al. [12] proposed MalDetect, a structure for detecting encrypted attack traffic. MalDetect can detect attack traffic before the actions of the attack have a practical impact since it only extracts features from the first 8 packets (the number changes depending on the flow). The Online Random Forest model is used in this model to distinguish between legitimate and malicious flow types. This avoids re-training and re-deploying the classifier when fresh samples are received by training it in online mode. It lowers personnel expenses and raises service standards. utilized Libpcap [Jacobson and McCanne], a C++ package that continuously records network traffic. Effectiveness, timeliness, and performance are three areas where MalDetect has undergone extensive testing. MalDetect can quickly and efficiently learn new attack traffic and increase the detection rate of unidentified threats if additional samples are provided.

MalDetect doesn't serve as a middle box that restricts traffic. It copies the packet content from the network card when a packet arrives so that it can be processed further. As a result, MalDetect is able to identify attack communications without adding extra time to the client-server connection. Libpcap continuously records network traffic. Following the network packets' capture, MalDetect processes them using two largely independent modules, feature extraction and training and detection. The collected packets from Libpcap are required as input for feature extraction. The intended flow is then represented by some informative byte fields that are chosen. The output of this module quantifies the values of these fields into numeric vectors. The following module, Training and Detecting, receives the numerical vectors to either train a machine learning classifier or be detected by a classifier that has already been trained. The 23 most robust features are selected from each flow independently and can be separated into three classes as follows: Packet feature: (inbound bytes, outbound bytes, inbound packets, outbound packets, duration, SPL: sequence of packet length, SPT: sequence of packet time). Transport Layer Security (TLS) protocol features: TLS Version, Offered Cipher Suites (Set of Algorithms), Selected Cipher Suite, Selected Compression Method, Offered Extensions, Selected Extensions, TLS Packet Ratio.

Certificate feature: Certificate Number, Bad Certificate Number, Certificate Version Ratio, Certificate Extension Ratio, Certificate Validity Mean, Certificate Public Key Length Mean, Certificate Public Key Algorithm Ratio, Certificate Signature Algorithm Ratio. Network flows are processed separately in MalDetect. A flow is uniquely identified by five elements: source IP, source port, destination IP, destination port, and protocol. MalDetect captures packets through Libpcap and applies a filter to TLS network traffic. As a result, a flow is represented by the vector [source IP, source port, destination IP, destination port]. MalDetect assigns a separate RAM space, made up of numerous counters and variables, for each flow and saves this vector in an array. Counters and variables are set to their initial values as soon as MalDetect collects the first packet of a new flow.

It changes newly received packets in accordance with the extracted byte field values. The numerical vector produced in the feature extraction module serves as the input for the training and detection modules. This module consists of two modes: training and detecting. In training mode, labels are required. It's crucial to know how to label the network card-captured flows. Since the training dataset is already available, it is possible to identify the different flow types in the flows before training by adding a particular mark. To assist in labeling the network flows, they also created the PcapEditor tool. This tool needs attack traffic flows, which are currently supported in.pcap format, and the appropriate labels. The number of labels is not constrained (legitimate or attack-generated). A label list that is predefined in the configuration file is maintained by PcapEditor. Users are able to update the configuration file and add their own labels. However, it should be noted that since MalDetect uses the label list to identify different flow types, the order and quantity of these labels must match those in MalDetect. In detection mode, MalDetect captures the traffic coming from a designated network card. Instead of training the classifier while extracting the value of random bytes, it will check to see if the value was generated randomly by the client and conduct flow type identification. In order to show detection results, MalDetect prints the flow ID, [source IP, source port, destination IP, destination port, protocol] and the flow type prediction. In this module, MalDetect adopts the online random forest model as the classifier.

Usually, Random Forest is trained offline, which necessitates the preparation of all training data. In practice, though, training data is

continuously produced. New types of network traffic are constantly evolving, particularly for the identification of attack traffic. Online random forests retain prior threat information while simultaneously picking up new threats. MalDetect has been demonstrated to be efficient, timely, and high-throughput in its ability to differentiate attack flows with a low false negative rate (FNR) and false discovery rate (FDR), and it can find the attack traffic before it carries out illegal actions. This is extremely beneficial because we can completely disable attack communication. However, using MalDetect independently is not enough. It is better to incorporate it into a threat-handling system, such as an IDS, and it has to become capable of handling a large number of flows at a time, so it may be deployed on a high-speed network. In this approach, two general datasets were used, i.e., the CTU-13 dataset and the MCFP dataset. CTU-13 dataset [Garcia, Grill, Stiborek et al. (2014)] - This dataset contains thirteen scenarios and was captured at CTU University, Czech Republic, in 2011. In each scenario, a specific attack that used various protocols and performed various tasks was executed.

It consists of three types of traffic, i.e., attack traffic, legitimate traffic, and background traffic. MCFP dataset [Erquiaga, Garca, and Garca Garino (2017)] This dataset has been collected by a group of researchers at CTU University. In their datasets, 76.65% of attacks do not adopt the TLS protocol. Therefore, only use traffic from the rest of the 23.35% attack. The attack generates traffic by running for extended periods of time, up to three weeks or even months, and it stores a range of captures from various attack types, including trojans, adware, botnets, etc. The detection result shows the false negative rates (FNRs) were 0.8%, which means 8 out of 1000 flows generated were considered legitimate flows. At the same time, the false discovery rates (FDRs) were 0.09%, which means that only 9 out of 10,000 flows reported as attacks generated by MalDetect were legitimate. MalDetect received gradual training using suspected attack traffic flows.

The false discover rates were estimated using 100 flows as input (FDRs). With the number of suspicious flows ranging from 0 to 300, the false negative rates (FNRs) substantially decreased from 100% to 3.35 percent. The false negative rates (FNRs) slightly dropped to 3.1% after 1000 Susp flows were input. This demonstrates how MalDetect, when taught with new varieties of attack flows in online mode, can detect new threats effectively [13].

Ferriyan, et al. [14] proposed a method for detecting encrypted attacks called TLS2Vec that creates words from extracted features and uses Long-short-term Memory (LSTM) for inference. It uses deep learning techniques incorporated with Word2Vec, which is a method to efficiently create word embeddings. TLS2Vec analyzes the Transport Layer Security (TLS) session by extracting the features from the payload and the raw PCAP of the TLS handshake, and then it creates a corpus and trains the dataset using LSTM. The authors assert that by utilizing only the TLS handshake information, their approach can distinguish between benign and malicious behavior with an average of 0.999. TLS2Vec does not need to decrypt the traffic, which will impact privacy, and it discovers the malicious traffic before the TLS handshake conversation ends and takes immediate action as soon as it determines that the traffic is malicious.

They use two datasets that use encryption protocols in their traffic for their valuations: the CTU-Attack-Capture dataset and Jason Stroschein's public Github attack samples (https://github.com/jstrosch/attack-samples). They used 4 dataset samples: Zeus, benign, and cobalt from CTU-Attack-Capture, and Trickbot from Jason Stroschein. The key to their research is to analyze the TLS session by identifying the features in the TLS handshake and the payload. Most of the encrypted network traffic has a specific format that differs from the others. Thus, using the knowledge of this format, it is possible to differentiate and identify malicious traffic.

TLS2Vec has three steps: (1) Feature Extraction: extracts features from raw PCAP to build a corpus. (2) Building Vocabulary and Token Parser: uses tokenization techniques to extract words from the training dataset and then applies word embedding techniques to represent words. (3) Training Model: TLS2Vec trains the dataset using LSTM and Bidirectional Long-Short-Term Memory (BiLSTM). It is possible to obtain information in two ways: the unencrypted content that resides in the header and the packet length of the payload. They got information from the unencrypted phase: the initial handshake and its properties, and the authentication information exchange identifier from the client and the server. During the initial TLS handshake, the client and server negotiate with both exchanging cipher suite information and which protocol version is used. The important thing about this behavior is that it distinguishes malicious applications coming from the client.

The authentication exchange information method in [12] considers extensions such as elliptic curves and ec_point_formats as features. The unencrypted phase communication between client and server follows the TLS protocol, which can be treated as one sentence in a TLS session. Each sentence has a specific pattern that can distinguish between malicious and benign behavior. The packet length from the TCP application data is the additional information feature extracted from encrypted traffic. Although some TLS server communication parameters can be changed over time, the application data's size remains constant. They considered using the TCP packet length from the application data. While they cannot see into the encrypted content, the packet length is directly linked to the payload of the traffic and usually follows an application-specific profile. Thus, they used packet length to determine which traffic was benign or malicious. Similar traffic represented by words will have the same vector space, whether benign or malicious. Their method can also detect malicious applications that use their own TLS library instead of the available library on the host. However, it cannot run independently but as an ensemble with the network intrusion detection system (NIDS) based on host behavior. Also, its monitory class does not perform well when the payload is short and binned, so they suggested further analysis in the case of shorter packets.

Pastor, et al. proposed a solution for one type of encrypted attack: cryptomining connections. They deployed a complex and realistic cryptomining scenario for training and testing machines

and deep learning models, in which clients interact with real servers across the Internet and use encrypted connections. They used simple and complex machine and deep learning models like fully connected neural networks, random forests, C4.5 and classification And Regression Trees (CART) decision trees, and logistic regression to train with 4 data sets created by Mouseworld, which is a controlled Network Functions Virtualization (NFV) based infrastructure that sets up and deploys on it the cryptomining attack scenario that would generate the traffic of experiments. They used Tstat, a traffic analysis tool, to analyze the TCP flows, and it allowed them to train and test machine learning components that can identify cryptomining flows even when only a few packets of the flow have been transmitted. They explained the Mouseworld Lab modules. They also explained the properties of the models they used for testing machine and deep learning and their strengths and limitations, and finally compared their performance in three scenarios using two different sets of statistical features obtained from network traffic flows.

They used two different sets of features with the aim of determining their impact on machine learning performance and demonstrating that with the use of more informative features, a significant increase in machine learning performance can be obtained. The first set of extracted features is extracted using the Tstat tool, and the second is derived from the Internet Engineering Task Force (IETF) standard NetFlow/IPFIX metrics. The first set of 51 features, which were derived from a subset of the Tstat tool statistics, contain features like acknowledge sent (ACK), data bytes, maximum segment size observed, and more. The second set of features that were derived from the IETF standard NetFlow/IPFIX metrics were 8 features: inbound and outbound packets/second, inbound and outbound bits/second, inbound and outbound bits/packet, bits-inbound/bits-outbound ratio, and packets-inbound/packets-outbound ratio. Then they evaluated their machine learning models with three sets of features:

a) In scenario A, the NetFlow/IPFIX metrics were utilized.

b) In scenario B only, the features derived from Tstat

c) statistics were used.

d) Finally, in scenario C, both the Tstat derived and

e) NetFlow/IPFIX features were jointly used.

They found that the best results were obtained using the set of features derived from Tstat unlike when they joined the two sets of features (Tstat + Netflow) where they did not obtain any observable advantage. They also concluded that using exhaustive features as input to complex machine learning models allows them to deploy precise, accurate, and stable mechanisms [1].

Carlos Novo, et al. [15] investigated the effects of the gap between generated and constructed hostile samples on learning and evasion. Modern deep learning systems are known to be susceptible to evasion attacks, in which a malicious sample is used to create an adversarial sample that is mistakenly identified as benign. Based on TCP/IP flow properties, the detection of encrypted attack command and control (C2) traffic can be framed as a learning job, making it susceptible to evasion assaults. Their proposed system included examples using white-box adversarial learning, a deep neural network detector trained on a publicly available C2 traffic dataset, and a proxy-based method for creating longer flows. When utilizing well prepared adversarial samples, the high evasion rate obtained by using generated adversarial samples on the detector can be greatly decreased. It's crucial to identify encrypted attack command and control traffic using machine learning and traffic characteristics, especially when attackers often switch IP addresses or server-side certificates that are blacklisted or use zero-day exploits.

But since adversarial attacks are known to be possible against machine learning algorithms, it makes sense that attackers would take advantage of this weakness and alter the behavior of C2 traffic between the victim and the C2 server to evade detection. But creating attack is expensive. Modifying the behavior of sophisticated code to achieve certain hostile traffic can be a chore with a significant influence on the profitability of most enterprises, even with a wide selection of open-source attack frameworks available. It may be more appealing and less expensive to apply traffic proxies or addons to the source code that do not alter the behavior of the infection but can alter traffic features. With mechanisms for replacing outdated payloads with new ones on the victims, deploying a changed attack on the C2 server and even on victims appears to be a comparably easier task. These limitations serve as the driving force behind the proposed system. Although it is generally agreed that attackers are more driven than defenders, the strategy in the study is to attempt to level the playing field between the two.

The proposed system sets out to explore what occurs when an attacker and defender each build their own plan and these methods collide, starting from a common public dataset. The study specifically strives to examine the performance of the attacker and defender in various attacker and defender setups. As they evaluated the effectiveness of hardening detection models and evasion assaults with the presumption that the attack should retain its original behavior. To achieve this, they created a labeled dataset using attack traffic captured from a public source, which both attackers and defenders can use to train and test C2 encrypted traffic detector algorithms. They demonstrated the effects of the practical restrictions on a particular attack they implement on the packet traces, with increasing adversarial sample misclassification. The detector is then strengthened with various feature set adversarial samples.

The proposed system research is for combinations of feasible attacks and models that would either make it impossible for the attacker to win or make it impossible for the defender to win, assuming that neither the attacker nor the defender knows which feature set the detector employs to harden the detector. According to the proposed system research, the best attack does not, however, produce the most hardened models overall because it does not allow for the addition or subtraction of adversarial values from any

original feature. They used the TCP static and analysis tool (Tstat) to extract 86 numerical features from traffic flows, some of which are: maximum and minimum time to live (TTL), flow duration, data segment and byte counts, and more. Last but not least, the study attempts to comprehend the same problems for several iterations of attacking and hardening a model. We discover that it is possible to achieve parity, where no hardened model is able to detect attacks of all iterations for a given number of iterations and a given training set strategy, nor can an attack cause a detector at all iterations to misclassify a significant portion of its adversarial samples.

Anderson & McGrew 2017 [18] used machine learning, which is a slow process, embedded in the network security industry. In their essay, they created and constructed experiments that demonstrate how and when to use six popular machine learning methods. Linear regression is one of the simplest machine learning models. It is efficient to train and test linear regression, and the interactions between the weight vector and the data features make it simple to understand the predictions that result. The result of a logistic Regression is a correct probability, which can be seen as the likelihood that a feature vector belongs to a particular class. As a set of rules can be connected to each output, decision trees are generally efficient to learn and simple to interpret. In a random forest, the average output variance is significantly reduced, which usually leads to better performance. Support-vector machine classifiers have been demonstrated to be reliable and produce the best outcomes for a variety of issues. Given the right kernel function, these models are flexible with little bias and work well in high-dimensional feature spaces.

For several tasks, including speech processing and picture identification, Multi-layer Perceptron (MLP) models with two or more hidden layers have proven to be state-of-the-art. Each algorithm has some limitations as well. Although the data is linearly separable, linear regression frequently performs poorly in a classification scenario and is dependent on the scale of the data components. It is also unable to describe nonlinear functions. When compared to a single decision tree, the random forest ensemble is harder to interpret. Support vector machine models often have very poor interpretability. Multi-layer perceptron models can learn very nonlinear functions and have very low bias. Also, there is a lack of interpretability in multi-layer perceptrons with one or more hidden layers, which is frequently crucial in the field of network security. However, there is a chance of overfitting because there are numerous parameters in these models that must be trained. With the malicious pcap files, they started to collect 10–30,000 new TLS sessions each month.

They monitored a single enterprise network and geographically separated the enterprise network. They used a well-known blacklist to filter the enterprise traffic. The blacklist was regularly updated. Despite this, there were certainly some malicious sessions left in the enterprise datasets. To construct the final datasets, they randomly sampled the data from each company's network without replacing any of it. There are several experiments to assess the strengths and weaknesses of six popular algorithms. They analyzed the effects that the enhanced feature set had on the algorithms. The experiments are: cross-validated accuracy, longitudinal study, distinct network, inaccurate ground truth, adversarial machine learning, and computational complexity. It is considerably more crucial to have a low false positive rate than to be 100% accurate. It displays the precision at a false discovery rate (FDR) of one in 100,000, or.001%. This represents the classifier's accuracy when just one false positive is permitted for every 100,000 true positives.

The reasonably balanced class composition in the testing datasets is not realistic; hence, this stringent metric was utilized as a stand-in for the outcomes that should be anticipated in a real network scenario. When the standard representation was used, the random forest ensemble was the only classifier with a significantly higher accuracy than zero at a.001% FDR. Except for linear regression and the decision tree, all methods performed equally well with the enhanced representation. A random forest ensemble was unquestionably the best-performing algorithm over time, retaining its accuracy on the enterprise dataset, using the standard representation. On the attack dataset, the random forest continued to perform better than most algorithms.

The random forest utilizing the enhanced representation nearly perfectly maintained its cross-validation accuracy for the corporate data throughout the course of the five months. When identifying fraudulent TLS connections over the course of the five months, the random forest algorithm was among the most competitive. It is noteworthy to observe that linear regression readily outperforms all other techniques that employ the traditional representation and is robust when utilizing the upgraded representation. The classification accuracy of these algorithms, which were biased towards attack samples and introduced label noise at a rate of 1.5% to 5.0% on the enterprise dataset, significantly improved, and there was no additional degradation on the attack dataset after 1.5%.

Using the upgraded data, MLP and linear regression were stable for all amplitudes of label noise. The strong bias of linear regression made it exceptionally resilient to label noise, even though it resulted in low accuracy at a fixed FDR. Despite the noisy labels, the random forest ensemble maintained its accuracy with the corporate data. Listing the accuracy of various classifiers when put to the test with the data obtained in the new network, the random forest group had four out of nine accuracy standards (99.53%) and enhanced (99.99%) classifiers, which was quite competitive. Once more, the properties of the revised data went beyond the algorithm's selection. In terms of attack data, it performed far better than all other classifiers with respectable enterprise performance, retaining an accuracy rate of 89% as opposed to 72-74%. The MLP method classified new samples quickly but required a long training period. The random forest ensemble classified new samples quickly after being trained. Traditional network devices can export these features, which are commonly derived from NetFlow [19] or IPFIX [21].

The enhanced features can also be exported by network devices; however, they are less efficient to obtain. The standard set and enhanced set each contain 22 and 319 data features, respectively. The standard set of features includes 22 features, including the client and server packet lengths' minimum, mean, maximum, and standard deviation, client and server packet lengths, server, client packet inter-arrival times, client and server packet inter-arrival times. Additionally they utilized the protocol, the length of the network connection, the quantity of client-server packets and bytes, and the quantity of server–client packets and bytes. By including individual packet lengths and times, which provide a more in-depth picture of the application's behavioral profile, and TLS metadata, which provides details about the library that the application is using for TLS, the enhanced feature set builds on earlier work. Bestafera, a specific attack sample renowned for keylogging and data exfiltration, is offered as an in-depth explanation of the upgraded features to show the intuition that the domain experts used to decide which aspects were useful in the context of TLS attack detection.

Jiayong, et al. [19], mentioned that the primary technique for detecting encrypted attack traffic is the supervised learning approach, which is based on aspects that are unaffected by encryption, such as statistical information retrieved from packets. Statistical features taken from packets are one type of feature that is not impacted by encryption, and they are currently the primary approach for detecting encrypted attack traffic. The three primary steps of the approach are as follows: Traffic concentration uses the Gaussian Mixture Model (GMM) algorithm to extract the center samples of each type of attack in order to address the class imbalance issue, and the Ordering Points to Identify the Clustering Structure (OPTICS) approach to cluster the center samples. They set up a framework using a distance-based approach that makes use of the GMM, an unsupervised learning technique, and ordering points to determine the clustering structure (OPTICS) to calculate the distance between attacks and use the distance to construct a new attack class called FClass.

The XGBoost algorithm then trains several models to create an identification framework based on the FClass. They also used them to compare the performance of the proposed method to that of two other widely used approaches: the multi-classification model and binary models with a single layer, and two more sophisticated approaches, CluClas and MalClassifier. They created a distance to determine how closely attack traffic datasets resembled one another on complicated datasets, which could be used to establish the new attack class FClass. The FClass was built based on the extracted features, which reflect the relationship and distinction between attacks. By using the CICIDS2017 dataset, attack-traffic analysis, and stratospheric IPS, the complete experiments have been carried out. These datasets are all often used in studies to assess the effectiveness of traffic identification. According to the distance-based strategy for developing an encrypted attack traffic identification framework, classification techniques are frequently unrelated to their traffic properties.

Different varieties of attack can have comparable traffic features or entirely different ones. The experimental dataset for the experiments combined three datasets. The University of New Brunswick has included the most frequent and benign attacks, including SQL injection, in the CICIDS2017 dataset. To increase its robustness, they used benign flow from this dataset in the experiment. There are two main kinds of traffic features that can be used to identify encrypted traffic.

One of them is extracted from the unencrypted contents of traffic, like features from the handshake of the TLS/SSL protocol. The other is to ignore the communication content and identify the traffic according to its statistical and numerical features. 83 numerical features are used, which can be separated into four groups. (1) The first group includes TCP and IP header features such as the sum of packet header bits and TCP internal ports; (2) time-based features such as average packet arrival time; (3) length-related features such as payload length; and (4) packet variation features such as the number of TCP window change times and payload length change times. When only the top 30 information-gathering characteristics are used for training, accuracy is only 83.56%; if all features are included, accuracy is 87.28%. The goal of feature selection in this study is to increase efficiency by selecting the feature set with high accuracy and reducing the number of features in the feature set.

The experiment demonstrated that the method could distinguish between various encrypted attack traffic with more accuracy and in less time than the distance-based technique, which could create a framework for distinguishing between various types of encrypted attack transmission. The approach blends supervised and unsupervised learning to provide a framework that is more accurate and efficient depending on the dataset and can also handle unidentified traffic. The major limitations are the need for advance acquisition of many attack traffic samples and feature extraction during detection. This severely restricts how widely this strategy may be used in practice. The class imbalance of a supervised learning approach to traffic identification is another issue. Data imbalance and feature imbalance are two different interpretations of the term "class imbalance".

The real-world environment presents various difficulties. To categorize attacks, there is no single standard. Even the names of attacks vary between different detection systems. It is so challenging to determine which attack class a piece of attack originally belonged to. A very small amount of daily network communication was attack traffic. Additionally, there are significant differences in the dataset sizes of the communications from various attack types. It creates an issue in that the model would be biased towards those classes with large data volumes and unable to recognize the classes with small data volumes if the original dataset were utilized directly for training.

In Table 1, a brief comparison of the articles was displayed. To summarize the comparison: Article 1 employed a dataset that includes three different forms of traffic: background traffic,

legitimate traffic, and attack traffic. They concentrate on datasets without labels and do not adopt the TLS protocol. To aid in labeling the network flows, they created the Pcap Editor tool. MalDetect picks the top 23 features and applies them. Three groups of features-packet feature, TLS protocol feature, and certificate feature-can be distinguished from the features that are independently acquired from each flow but do not offer statistical data for their analysis. MalDetect cannot simply detect attack traffic with low FNR and respond immediately. The reason for this limitation is that MalDetect does not integrate with IDS. In the case of article 2, they used Word2Vec, which makes words to label packets, and after training, it detects the word associated with packets.

Table 1: Comparison Of Related Research Paper.

Articles	Technique	Characteristic	Dataset	Feature extraction	Strength	Limitation	False Positive	False Negative
1) Jiyuan Liu et al.,2019 [13]	MalDetect utilizes Online Random Forest.	MalDetect It is a structure for detecting encrypted attack traffic. Before the attack starts exchanging information, it can recognize a malicious flow, and 23 reliable and accessible features have been chosen. To retrain and deploy methods, MalDetect uses Online Random Forest as its classifie.	CTU-13 and MCFP datasets	The 23 most robust features are selected and used in MalDetec. In the packet, 7 features are listed, 8 features are selected for the TLS protocol, and 8 features are extracted in the certificate packet.	MalDetect has the ability to identify a attack flow before the attack starts exchanging data. 23 reliable and simple to extract features are chosen. MalDetect can also quickly and accurately detect new threats by learning new attack traffic.	Utilizing MalDetect alone is insufficient because it cannot manage numerous streams at once.	0.8%	
2) Ferriyan et al.,2022 [14]	Deep Learning techniques incorporated with Word2Vec	Analyze the TLS session by identifying the features from the TLS handshake and the payload.	CTU-Attack-Capture dataset.	Version, Cipher, Ext_len, Elliptic curves Ec point formats. length	It does not need to decrypt the traffic. It discovers the malicious traffic before the handshake conversation ends and take immediate action.	Cannot run independently but as an ensemble with the NIDS based on host behavior.		
3) Pastor et al. ,2020 [1]	Fully Connected Neural Networks, Random Forest, C4.5 and CART decision trees and Logistic Regression	used a controlled NFV-based infrastructure, to set up the cryptomining attack scenario that generate the traffic of experiments then train a set of machine and deep learning models	Datasets generate from the Mouse world environment	From Tstat tool: acknowledge sent (ACK), data bytes, maximum segment size observed and more From Neyflow: 8 extracted features mentioned in related work	Exhaustive features as input to complex machine learning models allows to deploy precise, accurate and stable mechanisms	L Lack of precision and accuracy when Simpler machine learning models were used	Differ from a Scenario to another but worse false positive was in Scenario A which used Netflow feature selection method (77) FP flows	Low in Scenario B/C But worse false negative was in Scenario A which used Netflow feature selection method (393) FN flows.
4) Carlos Novo & Ricardo Morla, 2020 [15]	Deep learning in Flow-based Detection and Proxy-based Evasion	Examine how learning and evasion are affected by the difference between generated and created hostile examples.	Use data from https://malware-traffic analysis.net (MTA), which contains recent, vast,etc.	86 features extracted, some are: packet counts, Data segment and byte counts,Flow Duration and Max and min TTL	It is possible to frame the analysis of encrypted attack command and control communications as a learning task, leaving it open to evasion attacks	- The need to know how many iterations your opponent decided to do in their attacking hardening loop -The paper is build on several assumptions that may not be the case in real situation	Low	Low

5) Anderson & McGrew 2017 [18]	In this article, concentrate on supervised learning techniques	All machine learning algorithms significantly improved in performance by not depending exclusively on features that were easy to collect and by working with domain experts to iterate on how the data would be best represented.	enterprise dataset, with the malicious pcap files and randomly sampled the data from each company's network without replacing any of it.	These 22 features included the client and server packet lengths' minimum, mean, maximum, and standard deviation. the enhanced set has 319 data features.	Linear Regression it is simple, efficient to train and test linear regression, for classification, Logistic Regression is specifically created. decision trees are generally efficient to learn and simple to interpret. Random Forest The average output variance is significantly reduced, which usually leads to better performance. Support vector machine these models are flexible. Multi-Layer Perceptron (MLP) reliable and produce the best outcomes.	Although the data is linearly separable, linear regression frequently performs poorly in a classification scenario and is dependent on the scale of the data components. When compared to a single decision tree, the random forest ensemble is harder to interpret. Support Vector Machin models often have very poor interpretability. Multi-layer Perceptron models can learn very nonlinear functions and there is a chance of overfitting.		
6) Jiayong et al. , 2019 [19]	The primary technique for detecting encrypted attack traffic is the supervised learning approach.	use of the Gaussian mixture model (GMM), an unsupervised learning technique, and ordering points to determine the clustering structure (OPTICS) to calculate the distance between attacks and use the distance to construct a new attack class called FClass	Using the datasets from CICIDS2017, attack-traffic analysis, and stratospheric IPS, complete experiments have been carried out	83 numerical features are used, which can be separated into four groups. (1) TCP and IP header features (2) time-based features (3) length-related features such as payload length and (4) packet variation features such as the number of TCP window change times and payload length change times.	The suggested approach blends supervised and unsupervised learning to provide a framework that is more accurate and efficient depending on the dataset.	The major limits are the need for advance acquisition of many attacks traffic samples and feature extraction during detection. This severely restricts how widely this strategy may be used in practice. The class imbalance of a supervised learning approach to attack traffic identification is another issue.		

They used an existing unlabeled dataset and extracted only a few features from it. They did not specify the false positive rate of their experiment, and they used a deep learning model on encrypted and unencrypted traffic. Article 3's main focus was to create a new set of static features that can detect cryptomining attacks with high accuracy. They created real traffic between users and servers with the help of Mouseworld Lab. They then tested their new set of extracted features on multiple machine learning and deep learning models to detect cryptomining attacks and compared the results with an existing set of features. They did not create a specific detection tool, nor did they examine their set of extracted features against different attacks. Article 4 gives information on adversaries' success in evading detection and how to detect their encrypted attack through understanding their activity and 86 chosen extracted features on an existing data set using a deep learning model. The difficulty is that they needed to have knowledge of adversary activities that are not constant, and their method is built on assumptions that may not happen in real cases.

Article 5 combined three datasets as our experimental dataset. The CICIDS2017 dataset contains benign and the most common attacks, such as SQL injection, collected by the University of New Brunswick. The following are the primary issues: (1) There are evident differences between different types of traffic in terms of typical traffic identification, such as p2p traffic and HTTP traffic. There will be a considerably smaller variation between samples when identifying malicious traffic. (2) The class imbalance issue is substantially worse in attack traffic than it is in regular traffic. (3) The traffic carried by the attack is encrypted in this test environment, making it harder to identify it. The major issue is the need for advance acquisition of many attack traffic samples and feature extraction during detection. This severely restricts how widely this strategy may be used in practice. In article 6, the enterprise dataset, including malicious pcap files and randomly picked data from each company's network, there is an ethical consideration.

All non-attack network traffic utilized was gathered and analyzed in accordance with the guidelines established by the institution. All personally identifiable data, such as internal IP addresses and usernames found in unencrypted HTTP connections, was anonymized as part of this process. Strict access control measures are upheld to guarantee that all users have the necessary training to handle sensitive material and a legitimate reason for needing to examine it. The proposed method will select sets of features, apply them to an existing dataset using machine learning and deep learning models, and compare their accuracy. Then test a special set of selected features and some extracted feature results to finally come up with the best method for detecting encrypted attacks and integrating it with an intrusion detection system.

Dataset

In this research, we used the CICIDS 2017 dataset [20]. Since its creation, the CICIDS2017 dataset has attracted academics for study and the creation of new models and algorithms. The dataset, which spans eight separate files, contains five days' worth of traffic data from the Canadian Institute of Cybersecurity's normal and attack days. Another intriguing finding was that the dataset satisfied every need for a real intrusion detection dataset, including full network configuration, full traffic, labeled dataset, full interaction, complete capture, various protocols, attack diversity, heterogeneity, feature set, and meta data. The implemented attacks in the dataset include brute force (FTP-Ptator), DoS, web attack, Botnet, PortScan, and DDoS, the distribution is showed in Table 2. In order to construct a trustworthy reference data set, there were eleven features stated in the framework of the most recent dataset review.

Table 2: Experimental Datasets.

Types	Num of samples	Num of kinds
Infected	26267	6
Benign	12994	1

The features are: complete network configuration, complete traffic, labeled dataset, complete interaction, available protocols, and attack diversity. Using CICFlowMeter, more than 80 network flow features were extracted from the generated network traffic. Neither redundant observations nor null or zero values were found in the database. However, there were attributes or features, namely: Fwd Header Length, Flow Bytes, and Packets, that made redundant features, so we removed Flow Bytes at the feature selection phase and cleaned the dataset using Google Colab [17]. In our paper, we will test two sets of extracted features for training and evaluation. The evaluation criterion is the accuracy of detection, the False Positive (FP) rate, and the False Negative (FN). After the testing and comparison, the top-performing feature set is chosen as the final feature set to perform in our tool creation and is used to train our model.

We extracted two sets of features from this data set. The first set of features were 27 unstatical features, as shown in Table 3, which were extracted with reference to Gulab, et al. [17], and the results of the selected features were compared to the results of using all features and showed better accuracy and performance.

Table 3: Unstatical Selected Features [17,18].

Name	Description	Name	Description
1. Destination Port	The destination port	15. Fwd Packet Length Mean	Mean size of packet in forward direction
2. Flow IAT Min	Mean time between two packets sent in the flow	16. Fwd IAT Min	Minimum time between two packets sent in the forward direction
3. Total Bwd Packets	Total packets in the backward direction	17. Fwd Packet Length Max	Maximum size of packet in forward direction
4. Down/Up Ratio	Download and upload ratio	18. Bwd IAT Min	Minimum time between two packets sent in the backward direction
5. subflow Fwd Packets	The average number of packets in a sub flow in the forward direction	19. PSH Flag Count	Number of packets with PUSH
6. Init_Win_bytes forward	The total number of bytes sent in initial window in the forward direction	20. Fwd Header Length	Total bytes used for headers in the forward direction
7. init_Win_bytes backward	The total number of bytes sent in initial window in the backward direction	21. Total Backward Packets	Total packets in the backward direction
8. min_seg_size forward	Minimum segment size in the forward direction	22. Fwd IAT Mean	Mean time between two packets sent in the forward direction
9. Bwd Packet Length Std	Standard deviation size of packet in backward direction	23. Packet Length Variance	Variance length of a packet
10. Bwd IAT Std	Standard deviation time between two packets sent in the backward direction	24. Average Packet Size	Average size of packet
11. Packet Length Mean	Mean length of a packet	25. Bwd Packet Length Max	Maximum size of packet in backward direction
12. URG Flag Count	Number of packets with URG	26. Bwd Packet Length Mean	Mean size of packet in backward direction
13. Active Std	Standard deviation time a flow was active before becoming idle	27. Avg Bwd Segment Size	Average size observed in the backward directio
14. Total Length of Fwd Packets	Total size of packet in forward direction		

The second set of features was extracted from the existing 80 features of the CICIDS2017 dataset using the Independent Component Analysis (ICA) extraction model. Our contribution is to apply two sets of features to a variety of machine and deep learning models, including random forest, C4.5, linear regression, logistic regression, and neural networks. The final analysis will indicate which set of attributes is better at detecting encrypted assaults and which method is better at detecting them.

Before starting the training of the models, we begin by exploring the dataset and getting familiar with it. According to our goal of integrating it with an IDS, we needed to merge all the attacks into one CSV file. After that, we notice an imbalance in the dataset due to the large difference in the number of packets, as DDoS was 128027, whereas the FTP-Patator attack was 7938. We made the number of each kind of packet smaller so it would be more balanced, and the final numbers are as shown in Figure 1.

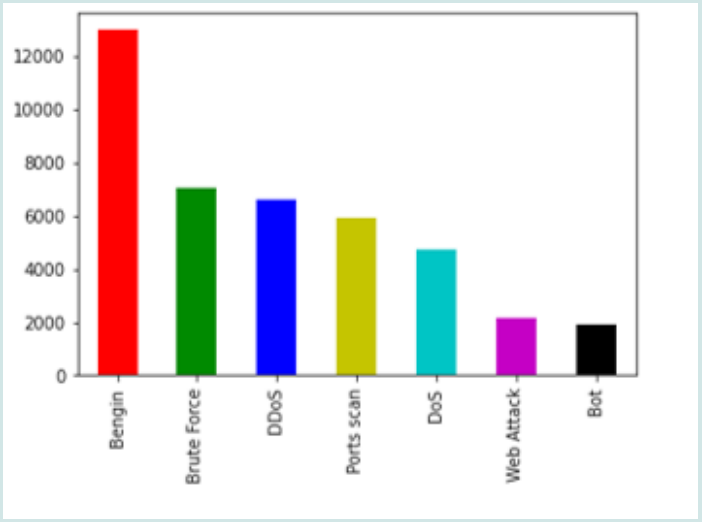

Figure 1: Numbers of Packets of Each Type.

A classification issue known as "unbalanced data" occurs when different data classes are not equally represented. It is a data collection with skewed class proportions, to put it another way. A binary classification problem, for instance, where 90% of the instances are labeled as "YES" (this is the majority class), and only 10% are labeled as "NO" (the minority class). In order to attain

high accuracy, the model in this situation will always forecast the outcome in accordance with the majority class, regardless of the data it is requested to predict. The term "unbalanced data" and its impact on machine learning models will be discussed in this section, and we'll discover how to handle it with the aid of certain widely used strategies, and we'll compare the outcomes at the end.

Resampling (oversampling and undersampling)

Undersampling entails eliminating instances from the majority class until we achieve the number of instances in the minority class, and oversampling is adding new examples (whether copy or artificial) to the minority class so that the number of instances in both classes becomes equal. We must first transform the categorical data into numerical data before using any resampling algorithm so that it can handle it. Several machine learning algorithms are unable to directly operate on categorical data. Numbers must be assigned to the categories. When there is no ordinal link, there can be issues, and letting the representation rely on any such relationship could harm your ability to understand how to tackle the issue. Under these circumstances, it is necessary to give the machine learning models greater expressive capability so that they can learn a probability-like number for each potential label value.

This may make it simpler for the algorithms to simulate the issue and provide more accurate results. By utilizing the Sklearn Library, label encoding in Python may be accomplished. A highly effective method for converting the levels of categorical characteristics into numerical values is offered by Sklearn. Labels having values between 0 and N classes-1, where N is the number of different labels, are encoded using LabelEncoder. The LabelEncoder's approach requires turning each value in a column into a number, which is quite straightforward. We changed the dataset labels to numbers 0-6, as shown in Table 4. That shows that our problem is modelled as a classifier, where the expected output is a multi-class value scaling from 0 to 6.

After merging the five attack days and preparing the data, we could finally start with feature selection and extraction.

System Methodology

Before starting the implementation, we designed our system architecture as shown in Figure 2. The CICIDS2017 dataset was cleaned and normalized, and the feature sets were extracted after that. Then for each model, we split the dataset into two sets: one for training and the other for testing. Each model is trained using the statistical and unstatistical extracted feature sets and then tested to finally compare the results of each model to come up with our tool.

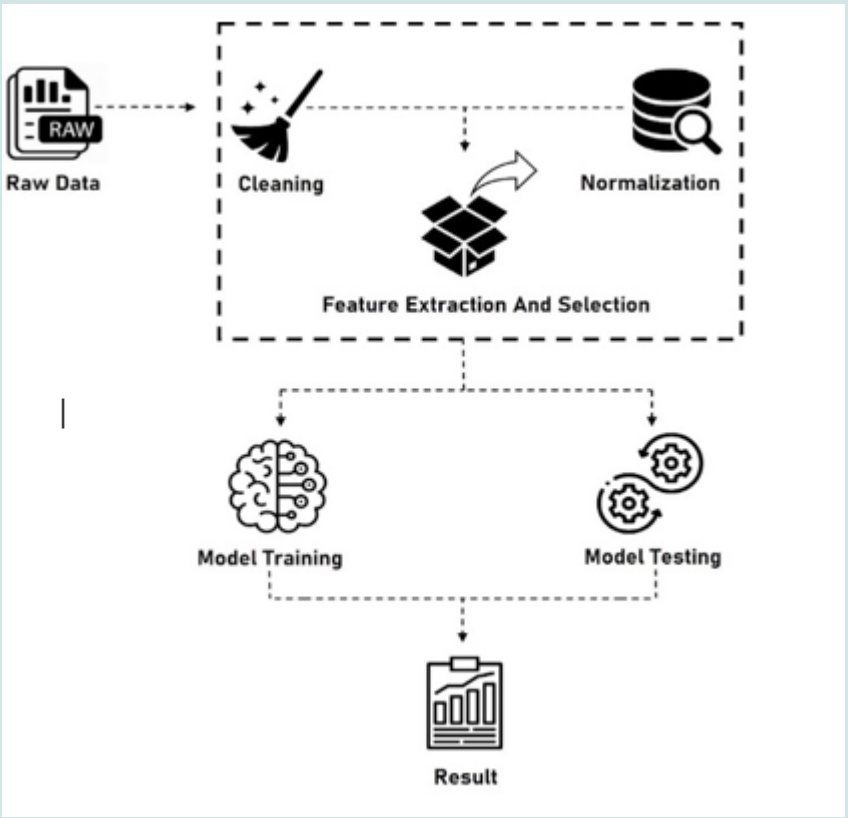

Figure 2: Methodology.

Implementation

Features sets

As mentioned in the previous section, the first set of features were 27 unstatistical selected features. Using Google Colab, we made two copies of the merged dataset and removed from one of them any additional features to get a dataset that contained 28 selected columns (27 selected features plus 1 for the label column) and 41420 rows. This set of features showed good performance in the Gulab, et al. [17] article. The dataset was divided into two training and test sets, one using all features and one using selected features. Comparisons of accuracy, precision, sensitivity, the F1-score, and G-means were made. In addition, it demonstrates the percentage predicted for each category (DOS, botnet, brute force, infiltration, portscan, web attack, and DDoS) and the overall testing and training time needed by the DT model throughout the CICIDS2017 dataset, utilizing both selected features and all features. The results were better and more accurate when the features were selected for each attack in terms of accuracy, precision, sensitivity, the F1-score, and the G-mean. Also, the training and testing time was less compared to training and testing all the features.

In particular, the effectiveness of machine learning classifiers is influenced by the choice of input characteristics as well as model variations and configurations. Then, as a second part of the project, we used the other copy of the merged dataset that contained all 80 features and uploaded it to Google Colab, then imported the Independent Component Analysis (ICA) extraction model to extract statistically independent hidden factors, and we extracted three new statistical features from them. Together with input attributes, we also added information about the output class. Because ICA produces a set of maximally independent component vectors, it is categorized as an unsupervised learning method because it produces a set of vectors with the most independent components.

An extraction method called Principal Component Analysis (PCA) was employed. PCA's goal is to minimize the dimensionality of a data set made up of numerous variables that are connected with one another while preserving as much of the dataset's variation as possible. By converting the variables into a new collection of variables known as the primary components. It is important to remember that PCA finds the rotation in an unsupervised manner, without the use of classifiers. It merely examines the data's correlations. Anyway, it was not as accurate as the ICA method, and it also did not extract statistical features as we wanted.

Training models

Two Google Colab codes were used, one for each set of features. We first imported the necessary model libraries, as well as any additional important libraries, into the selected feature code. Then I uploaded the dataset and made sure it was the 28-feature merged dataset, following which I equated the 27 selected features with the X variable and the last feature column, which was the Label column, as the Y variable. For the extracted features, we imported the FastICA function with the parameter component equal to 3 and extracted new features that were assigned to the X variable, and the Y variable contained the labeled values. In both cases, these variables were used to start splitting the data set into a train dataset with 70% traffic and a test dataset with 30% traffic and were passed to the models.

The Random Forest (RF) model was the first one we trained. Based on random subspace and bagging, Random Forest RF employs CART DTs as its fundamental algorithm. Both categorization and regression are effective with it. It introduces randomness into the training and assessment phases of learning, which causes each tree to differ from the others. Each tree is combined in predictions, which lowers the variance of the prediction and enhances performance [22]. It is a modeling technique that combines many independent classification trees. If the calculation is not greatly increased, the algorithm can boost prediction accuracy. A number of classification trees are used in the random forest classification approach, and each classification tree is built using guided samples of data.

Variables from each division are chosen at random to serve as the candidate variable set for tree building. It should be emphasized that throughout the training and verification procedures, all datasets should be divided at random. Hyperparameter tuning is done in order to get the optimal random forest structure. by applying it to tune four important parameters (the number of trees, known as n estimators; the minimum number of samples needed at a leaf node in RF, known as min samples leaf; the maximum depth of the tree in random forest, known as max depth; and the minimum number of samples needed to split a node, known as min samples spli); using the grid search method. Grid search is a tuning method that seeks to determine the ideal hyperparameter values. It is a thorough search that is done on a model's particular parameter values. An estimator is another name for the model.

We passed the training and testing arguments: the (X_Train/Y_Train) arguments were passed to the model fit method, and the (X_Test/Y_Test) arguments were passed to the model score method, and we calculated the time taken.

One of the simplest and most widely used machine learning techniques is linear regression. It was imported right after RF, but soon after we noticed its unsuitability for our problem because it is a statistical technique for performing predictive analysis for continuous/real/numeric variables like sales, salary, age, and product price. Regression problems are solved using linear regression. In contrast to classification issues, which demand discrete values, linear regression predicts the value of continuous variables. That is why we decided to change it to the logistic regression technique, which was employed to predict the categorical dependent variable using a predetermined set of independent variables. Classification issues are solved using logistic regression. We estimate the values of categorical variables using logistic regression. A categorical value, such as 0 or 1, Yes or No, etc., must be the result of a logistic regression. As we did with the Random Forest (RF) model, we passed the argument into the model and trained it to see how good it was at solving our problem.

As with the previous model, the C4.5 model was applied similarly, and it is used in data mining. The C4.5 method serves as a decision tree classifier that used to produce a decision based on a specific sample of data (univariate or multivariate predictors). We do "import decision tree classifier" then we can determine if this model is C4.5 by making "criterion = entropy", We get them from the SKlearn documentation. After that, we call the C4.5 by a variable assigned to the "Decision Tree Classifier" library, then fit it by "Y train, X train" and we start training the model, and calculate the accuracy and print time, and we found out that the test accuracy was 99.94%. The C4.5 model performed well overall, but the RF model performed better in terms of accuracy, false negatives, and false positives.

Finally, the last model is the Neural Network (NN), which is a subset of machine learning and is at the heart of deep learning algorithms. It is comprised of node layers, each containing an input layer, one or more hidden layers, and an output layer. Although Neural Networks (NNs) are used in many aspects of daily life, their main goal is to simulate how neurons in the human brain process information and data. The human brain is made up of nerve cells and neurotransmitters to handle orders and inputs (data); a neural network follows this structure, processing data and learning from it to anticipate outcomes based on inputs. A neural network works by utilizing a flexible algorithm that absorbs knowledge from the available data and generates predictions in reaction to it [22]. We used it with the sequential model, which ensures that every layer is an input to the next layer and no overlaps happen. We passed the training and testing argument to the model, which resulted in 3 layers and Rectified Linear Units (ReLU) as activation functions.

To improve the training and calculate accuracy and time, we used Adam Optimizer to create batches of the passed rows and divide the dataset. Machine learning models (linear and logistic regression and RF) were trained using the available Python code versions in the well-known scikit-learn library. where C4.5 was imported through the decision tree classifier library and NN was implemented in Python using the Keras framework.

Experimental Results

We conducted a series of experiments to assess how well various machine and deep learning models performed at identifying encrypted attacks and separating them from regular traffic. At the start of the experiments, two complementary aims were set. First, determine which set of features is better at detecting encrypted attacks, and then determine which model is better at detection. To compare their outcomes with those of larger deep learning neural networks, we chose ML models like linear regression and C4.5 trees as examples of simpler models. Also, logistic regression was chosen as an example of a very straightforward methodology, and random forest was chosen as it is usually regarded as one of the finest performing ML techniques. We compared the results of these models using the selected and extracted features to evaluate the best model and the best set of features.

We observed that due to the small size of the merged data set, the simple models performed well and gave good results, unlike the very complex neural network, which resulted in bad results due to our data being small in size and numerical, whereas neural networks deal with millions of rows, pictures, and sounds (Figure 3). Looking at the results in Table 5, neural network and linear regression were excluded due to the low accuracy they provided. The main comparisons were between RF, C4.5, and logistic regression. We computed a set of quality metrics widely used in classification problems to compare and rank the obtained results in validation and testing: accuracy, precision, recall, F1 score, and confusion matrices. The components were primarily about accuracy, False Positive (FP), and False Negative (FN).

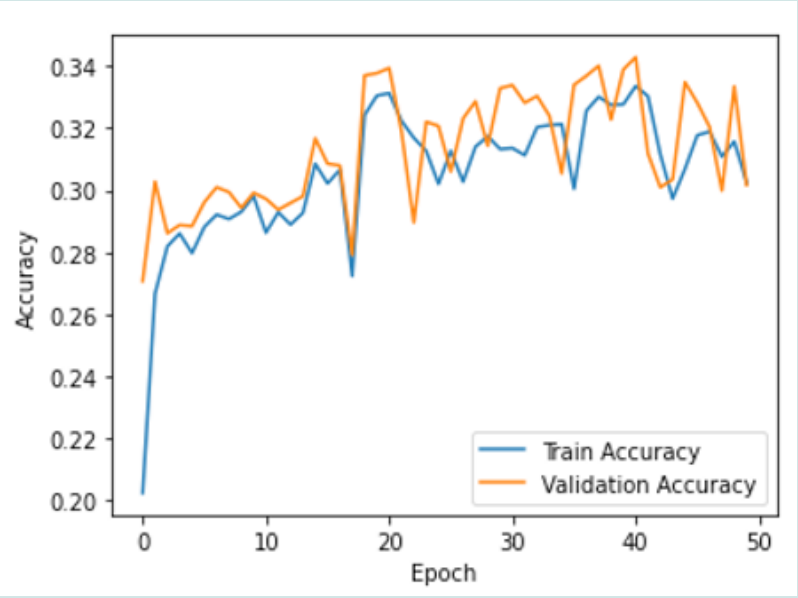

Figure 3: Neural Network Accuracy.

Table 4: Labeling Attacks as Numbers.

Number of Packets	Type of Attack
0	Bengin
1	Botnet
2	Brute Force
3	DDoS
4	DoS
5	Ports Scan
6	Web Attack

Table 5: Results and Discussion.

		Selected Unstatistical Features				Extracted Statistical Features			
Algorithm		Accuracy	Time	False Positive	False Negative	Accuracy	Time	False Positive	False Negative
01	Random Forest	99.79%	16.9 s	0.001	0.0005	89%	15.9 s	0.042	0.04
02	C4.5	99.57%	0.30 s	0.003	0.003	91.69%	0.1 s	0.02	0.039
03	Logistic Regression	93.51%	22.9 s			31.18%	1.55 s		
04	Linear Regression	45.82%	0.08 s	-	-	5.83%	0.01 s	-	-
05	Neural Network	30.17%	160 s	-	-	4.4%	31.5 s	-	-

However, in the selected feature case, RF performed best, with an accuracy of 99.79%, a FN of 0.0005, and a FP of 0.001, see Figure 5. Although C4.5 was very close in accuracy and even better with time, it was still not as good in FN and FP, in addition to other quality metrics that performed better with RF, see Figure 6.

After displaying the result, we can see that in the extracted feature case, C4.5 was the best model with 91.69% accuracy in only 0.1s , which is better than RF, which provided 89% accuracy. The C4.5 confusion matrix, Figure 4 shows the FN and FP rate at the test. However, in the selected feature case, RF performed best, with an accuracy of 99.79%, a FN of 0.0005, and a FP of 0.001, see Figure 5. Although C4.5 was very close in accuracy and even better with time, it was still not as good in FN and FP, in addition to other quality metrics that performed better with RF, see Figure 6.

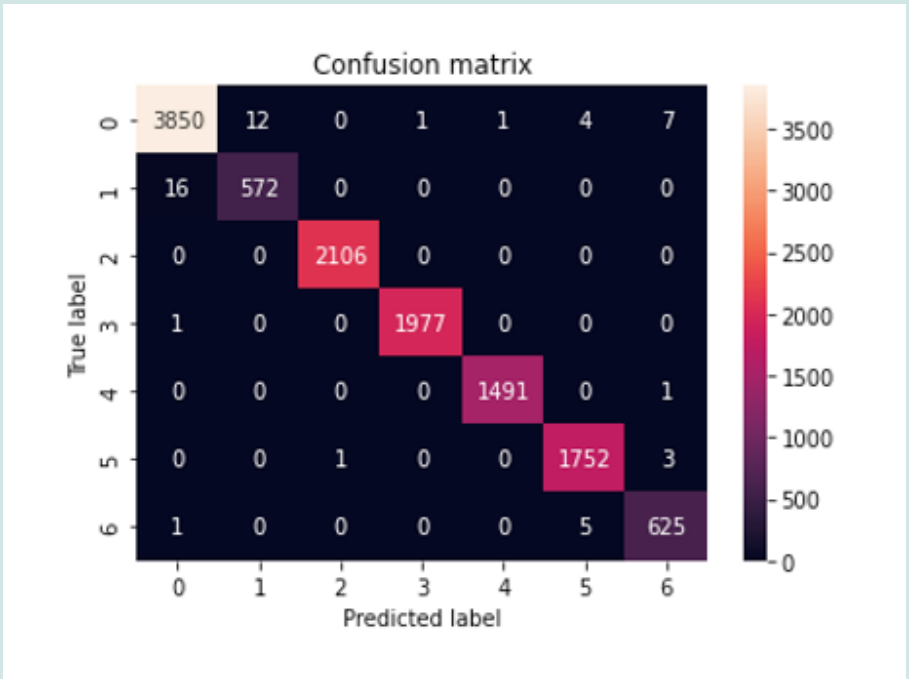

Figure 4: C4.5 Confusion Matrix on Extracted Features.

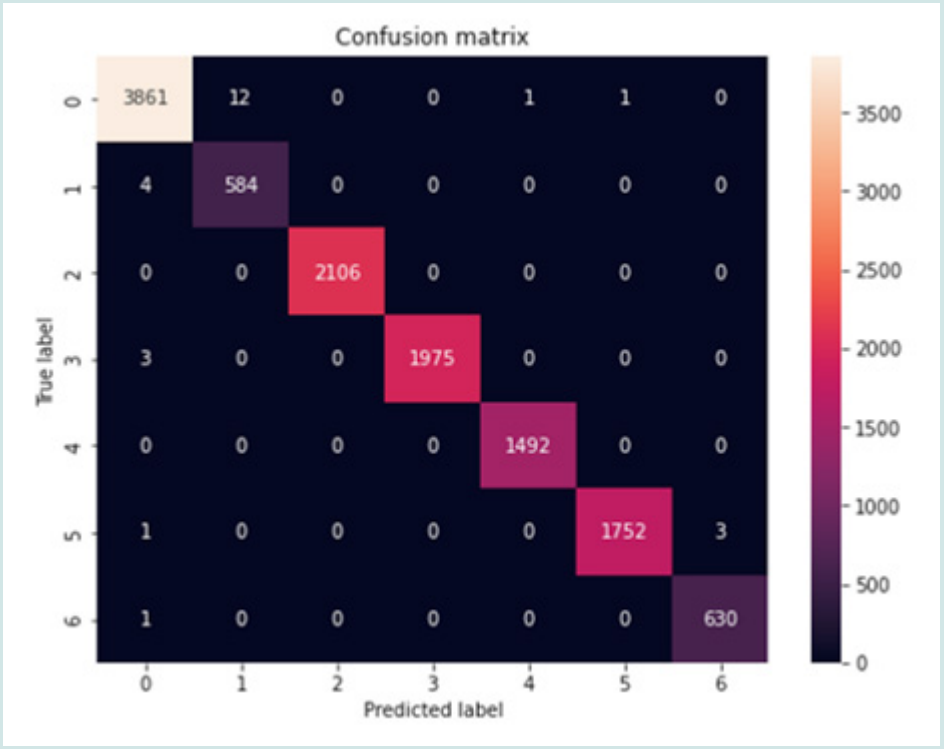

Figure 5: Random Forest Confusion Matrix on Selected Features.

C4.5

	precision	recall	f1-score	support
0	0.99	0.99	0.99	3875
1	0.98	0.97	0.98	588
2	1.00	1.00	1.00	2106
3	1.00	1.00	1.00	1978
4	1.00	1.00	1.00	1492
5	1.00	1.00	1.00	1756
6	0.99	0.99	0.99	631

RF

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3875
1	0.98	0.99	0.99	588
2	1.00	1.00	1.00	2106
3	1.00	1.00	1.00	1978
4	1.00	1.00	1.00	1492
5	1.00	1.00	1.00	1756
6	1.00	1.00	1.00	631

Figure 6: C4.5 and Rf Precision, Recall and F1 Score of Every Attack.

Finally, by comparing the results between selected and extracted features, we have found that the selected features performed way better in all aspects except time. After getting these results and determining the best model and set of features, we began the process of integrating them with an Intrusion Detection System (IDS) to help create a healthy climate for businesses and steer clear of shady network activity. The purpose of the IDS is to help computer systems learn how to deal with attacks, and each IDS gathers data from a variety of sources within computer systems and networks before comparing it to previously established patterns of discrimination to determine whether there are assaults or flaws. Monitoring network resources with the intention of spotting unusual behavior and network abuse is the purpose of intrusion detection. In essence, the intrusion detection system detects and alerts on signs of an attack.

Information is gathered from host and network resources in modern intrusion detection systems. When more threats are identified and fewer false positive alarms are generated, an intrusion detection system performs more accurately. Modern networks are constantly growing and developing, which has led to an increase in the scope and destructive capacity of cyberattacks. Systems for detecting intrusions are crucial for maintaining and strengthening network security. However, due to a lack of resources, implementing and integrating in real IDS hardware was difficult in our case, and the alternative solution was to create an IDS interface as shown in Figure 7. IDS-ML is code written in Python in the Visual Studio API using the Stremlist library to develop IDS identifiers designed to create IDS from datasets of public network traffic using both conventional and cutting-edge Machine Learning (ML) techniques. Random forest was chosen as the ML approach. By utilizing 28 features, advanced identity detection systems can recognize and predict various cyberattack types to safeguard contemporary networks. To address issues with cybersecurity, this code repository may be applied and quickly duplicated on any intrusion detection datasets.

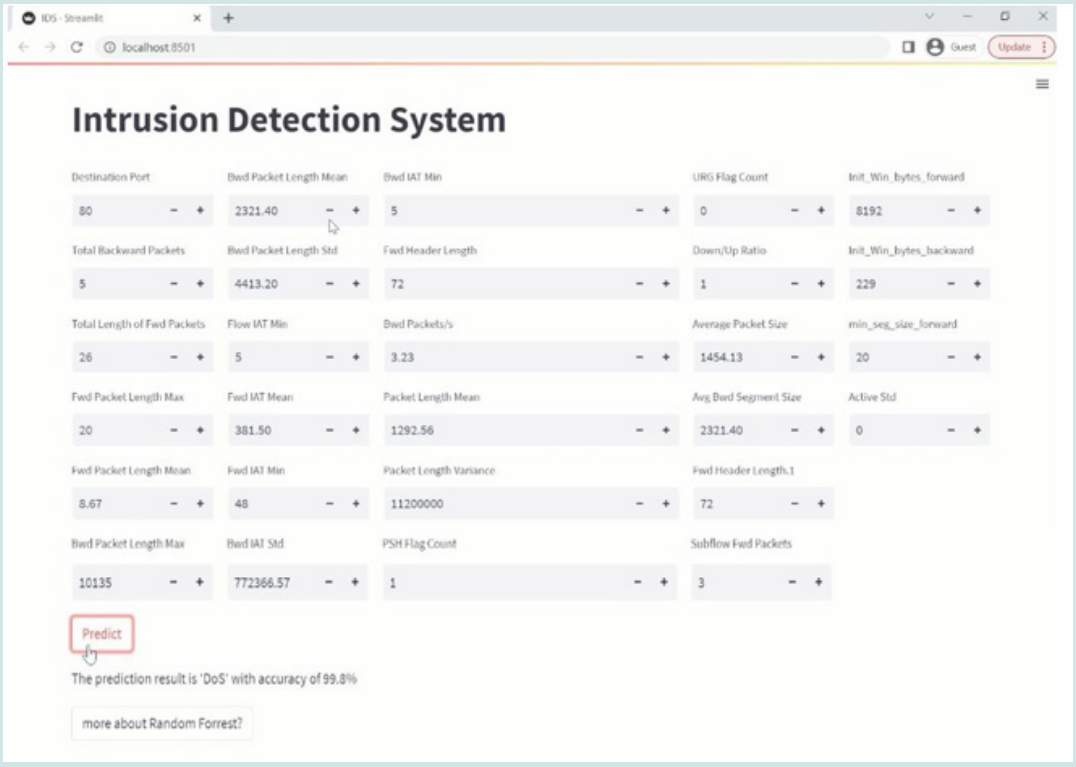

Figure 7: Developed Ids Interface.

Conclusion and Future Work

This study designed, trained, and tested a set of machine and deep learning models for detecting encrypted attacks. We selected several models, such as deep neural networks, random forests, logistic regression, linear regression, and C4.5, in order to compare their performances. We evaluate machine learning performance with the CICIDS2017 dataset, which contains encrypted flows of normal and attack traffic. As a main contribution, we tested two sets of features on these different models: unstatistically selected features and statistically extracted features using the ICA extraction method. In addition, we created an IDS interface with the best model and set of features (random forest and unstatistical selected features) and tested their viability. For future work, we might bypass the limitation of using a dataset for our project and capture and classify the attacks through real-time traffic that is monitored using the T-Stat tool.

As well as improving, we want to deploy an IDS interface on hardware devices to cover more types of attacks. As for the recommendations drawn from our papers, training, and testing models, whether machine learning or deep learning, they should be chosen according to the research requirements and the data set chosen for more accurate results. The stage of classification of features is one of the most essential stages influencing the results, so it is extracted and selected according to previous experiences and expertise.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

Research Data Policy and Data Availability Statements

The datasets analysed during the current study are available in the University of New Brunswick. (2017). CIC IDS. [Online].

Available: http:// www.unb.ca/cic/datasets/ids-2017.html

References

1. Pastor A, Mozo A, Vakaruk S, Canavese D, López DR, et al. (2020) Detection of encrypted cryptomining attack connections with machine and deep learning. 8: 158036–158055.
2. Tang S, Huang X, Chen M, Sun C, Yang J (2019) Adversarial attack type I: Cheat classifiers by significant changes. IEEE Transactions on Pattern Analysis and Machine Intelligence 43(3): 1100-1109.
3. Barut O, Grohotolski M, DiLeo C, Luo Y, Li P, Zhong T, et al. (2020) Machine Learning Based Attack Detection on Encrypted Traffic: A Comprehensive Performance Study: 7th International Conference on Networking, Systems and Security. ACM Other Conferences.
4. Harman N, Snowden A (2022) Oligomorphic groups and tensor categories.
5. Stallings W, Brown L, (2012) Computer Security: Principlesand Practice. Upper Saddle River, Pearson Education, NJ, USA.
6. Alam Shahid, Horspool N (2015) A Framework for Metamorphic Attack Analysis and Real-Time Detection. Computers & Security.
7. Yan W, Zheng M, Mcafee Z, Ansari N (2008) Revealing Packed attack. IEEE Security and Privacy Magazine 6(5):65-69.
8. Preda MD Code Obfuscation and Attack Detection by Abstract Interpretation.
9. Aldriwish K (2021) A Deep Learning Approach for Attack and Software Piracy Threat Detection. Engineering, Technology & Applied Science Research 11(6): 7757-7762.
10. Alamer, Ahmed, Ben Soh (2020) Design and Implementation of a Statistical Testing Framework for a Lightweight Stream Cipher. Engineering, Technology & Applied Science Research 10(1): 5132-5141.
11. Aslan, R Samet (2020) A Comprehensive Review on Attack Detection Approaches. Institute of Electrical and Electronics Engineers Inc 8: 6249–6271.
12. Liu J, Zeng Y, Shi J, Yang Y (2019) MALDETECT: A Structure of Encrypted Attack Traffic Detection. Research Gate.
13. Ferriyan A, Thamrin AH, Takeda K, Murai J (2022) Encrypted Malicious Traffic Detection Based on word2vec. Electronics 11(5): 679.
14. Novo C, Morla R (2020) Flow-Based Detection and Proxy-Based Evasion of Encrypted Attack C2 Traffic.
15. S. Uldun Mostfa Kamal, R. Jabbar Abd Ali, H. Kamal Alani, and E. Saad Abdulmajed, "Survey And Brief History On Attack In Network Security Case Study: Viruses, Worms And Bots," 11(1):2016
16. G Sah and S Banerjee, "0," 2022
17. Anderson, Blake, David McGrew (2017) Machine Learning for Encrypted Attack Traffic Classification: Accounting for Noisy Lables and Non-Stationarity. ACM Conferences.
18. Liu J, Tian Z, Zheng R, Liu L (2019) A Distance-Based Method for Building an Encrypted Attack Traffic Identification Framework. IEEE Access 7: 100014-100028.
19. University of New Brunswick (2017) Intrusion Detection Evaluation Dataset (CIC-IDS2017).
20. Lashkari AH (2021) CICFLOWMETE R/README.TXT at master· Ahlashkari /cicflowmeter, GitHub.
21. Anwer M, Khan SM, Farooq MU, Waseemullah (2021) Attack Detection in IoT using Machine Learning. Eng. Technol Appl Sci Res 11(3): 7273-7278.
22. Khan U, Khan K, Hassan F, Siddiqui A, Afaq M (2019) Towards achieving machine comprehension using deep learning on non-GPU machines. Engineering, Technology & Applied Science Research 9(4): 4423-4427.

DOI: 10.32474/CTCSA.2023.02.000147