Issues in Information Systems
Volume 24, Issue 4, pp. 82-94, 2023

DOI: https://doi.org/10.48009/4_iis_2023_107

Anomaly detection in network traffic using entropy-based methods: application to various types of cyberattacks

Vadim Bashurov, *Comtrade, Serbia, Vadim.Bashurov@comtrade.com*
Paul Safonov, *Saint Cloud State University, Safonov@stcloudstate.edu*

Abstract

This paper proposes an entropy-based approach for detecting anomalies in network traffic. With the exponential growth of data and sophisticated cyberattacks traditional methods struggle to identify evolving attack patterns. To address this, we leverage Shannon and Renyi entropies to analyze network traffic datasets. We are focusing on the entire network traffic. Using a publicly available dataset with labeled traffic samples, we calculate the entropy of different traffic features to assess their effectiveness in anomaly detection and attack identification. The scalability and sensitivity of this approach make it suitable for analyzing diverse and high-volume network data, capturing changes in traffic distributions, and detecting anomalies missed by traditional metrics. The method is easily implementable and interpretable, requiring minimal training data. Our findings show promising results for nine different types of cyberattacks, offering practical insights for robust anomaly detection systems in network security .

Keywords: cybersecurity, anomaly detection, network traffic analysis, Shannon entropy, Renyi entropy, Tsallis entropy

Introduction

In a typical network environment, there are multiple devices connected, such as computers, servers, routers, switches, and IoT devices. Each of these devices may generate network traffic as they send and receive data. Network traffic can include various types of communication, such as web browsing, email, file transfers, video streaming, voice calls, and more.

The network traffic can be categorized into different types based on its source, destination, protocol, port, or other characteristics. For example, one might have internal network traffic that occurs between devices within a network, or external network traffic that originates from or is destined for outside the network.

The exponential growth of network data and the increasing sophistication of cyberattacks have highlighted the need for effective anomaly detection techniques. The first anomaly detection method for intrusion detection was proposed almost 40 years ago (Denning, 1987). Traditional methods often struggle to identify complex and evolving attack patterns. To address this challenge, we propose leveraging the concept of entropy as a measure of uncertainty and variability in network traffic. Recently, entropy-based methods which rely on network feature distributions has been of a great interest (Berezinski et al., 2015).

The research focuses on applying Shannon entropy, along with other variants such as Renyi and Tsallis entropies, to analyze network traffic datasets. The study utilizes a publicly available dataset from New Wells University, which includes labeled traffic samples with nine types of attacks. By calculating the entropy of different traffic features, we assess their effectiveness in detecting anomalies and identifying

attack patterns. The main goal of this article is to prove that entropy-based approach is suitable to detect modern botnet-like malware based on anomalous network patterns.

The scalability and sensitivity of the entropy-based approach make it suitable for analyzing diverse and high-volume network data. It provides the flexibility to capture changes in traffic distributions and identify anomalies that might be missed by traditional metrics such as packet rate. Moreover, the simplicity and interpretability of the method enable its easy implementation and applicability for real-world scenarios without the need for extensive training data.

Our findings demonstrate that entropy-based methods offer promising results for anomaly detection in network traffic. By quantifying the level of uncertainty and diversity within traffic patterns, these methods can effectively identify abnormal behavior and potential attacks. This research contributes to the growing body of knowledge in network security and provides practical insights for developing more robust anomaly detection systems.

Only a few forms of entropy have been applied to network anomaly detection. The most popular are the well-known Shannon entropy and Renyi entropy. Application of Shannon measures to detect network anomaly were proposed by Lee and Xiang (2001). Next, Lakhina et al. (2005) made use of Shannon entropy to sum up a feature distribution of network flows. By using unsupervised learning, the authors showed that anomalies can be successfully clustered. Ranjan et al. (2007) suggested another worm detection algorithm which measures Shannon entropy ratios for traffic feature pairs and issues an alarm on sudden changes. Gu et al. (2005) made use of Shannon maximum entropy estimation to estimate the network baseline distribution and to give a multi-dimensional view of network traffic. The authors claim that with their approach they were able to distinguish anomalies that change the traffic either abruptly or slowly.

Methodology

Shannon's entropy can be utilized as a method to detect anomalies in network traffic by analyzing the entropy values of specific network traffic features. Shannon's equation of entropy is a fundamental concept in information theory developed by Claude Shannon. The Shannon entropy equation is used to measure the average amount of information or uncertainty in a random variable. It calculates the entropy based on the probability distribution of the variable's possible outcomes. The equation is as follows:

$$H(X) = -\sum_{N} P(x) log_2(P(x))$$

In this equation H(X) represents the entropy of the random variable X and P(x) represents the probability of the outcome x.

The entropy value calculated using this equation quantifies the amount of information or uncertainty associated with the random variable X. Higher entropy values indicate higher uncertainty or randomness, while lower entropy values indicate lower uncertainty or more predictable outcomes.

Let's consider a simple example to calculate Shannon entropy. Suppose we have a sequence of characters: “ALLACOOL". Length of sequence is equal 8 characters.

1. Count the occurrences of each character:
   - A: 2 times
   - L: 3 times
   - C: 1 time
   - O: 2 times
2. Calculate the probability of each character by dividing the count by the total number of characters:

- $P(A) = 2/8 = 0.25$
- $P(L) = 3/8 = 0.375$
- $P(C) = 1/8 = 0.125$
- $P(O) = 2/8 = 0.25$

3. Calculate the entropy using Shannon's entropy equation:
   - $H(X) = - \Sigma P(x)*\log_2(P(x))$
   - $H(X) = -(0.25*\log_2(0.25) + 0.375*\log_2(0.375) + 0.125*\log_2(0.125) + 0.25*\log_2(0.25))$
4. Perform the calculations:
   - $H(X) = - (0.25*(-2) + 0.375*(-1.41) + 0.125*(-3) + 0.25*(-2))$
   - $H(X) = 0.5 + 0.52 + 0.375 + 0.5$
   - $H(X) = 1.77$

Therefore, the Shannon entropy of the given sequence "ALLACOOL" is 1.75 bits. This value represents the average amount of information or uncertainty associated with each character in the sequence.

To calculate the entropy of the sequence "AAAAA," we can follow the steps:

1. Calculate the probability of the unique symbol (in this case, 'A'):
   - Count the occurrences of 'A': 5
2. Calculate the probability of 'A' by dividing the count by the total number of symbols:
   - $P(A) = 5/5 = 1$
3. Calculate the entropy using Shannon's entropy equation:
   - $H(X) = - \Sigma P(x) * \log_2(P(x))$
   - $H(X) = - (1 * \log_2(1))$
4. Perform the calculations:
   - $H(X) = - (1 * 0)$
   - $H(X) = 0$

Therefore, the Shannon entropy of the sequence "AAAAA" is equal to 0. Since there is only one symbol (A) in the sequence, it has no uncertainty or variation, resulting in an entropy of 0.

Let's consider an example where we calculate the Shannon entropy of the source IP addresses in a network capture. We need to obtain attributes of the network traffic that we want to analyze. We can use any public datasets or public tools like Wireshark to capture and save network traffic. In the Table 1 there is the example of network traffic including the source IP addresses, source ports, destination IP addresses, destination ports and protocol types. All network lines are sorted by time and numbered from 1 to N.

Table 1: Dataset with specific network traffic attribuites

N	Source IP	Source Port	Dest IP	Dest Port	Protocol
1	59.166.0.1	31992	149.171.126.4	7662	tcp
2	59.166.0.2	16267	149.171.126.4	80	tcp
3	59.166.0.1	3901	149.171.126.1	76891	upd

Let's calculate the Shannon entropy for the selected feature based on the probability distribution of its values. Let's consider a simplest example with a sequence of source IP addresses:

We have the following sequence of IP addresses from Table 1:

[59.166.0.1, 59.166.0.2, 59.166.0.1]

1. Count the occurrences of each IP address:
    - *59.166.0.1: 2*
    - *59.166.0.2: 1*
2. Calculate the probability of each IP address by dividing the count by the total number of IP addresses:
    - *P(59.166.0.1) = 2/3 = 0.667*
    - *P(59.166.0.2) = 1/3 = 0.333*
3. Calculate the entropy using Shannon's entropy equation:
    - $H(X) = -\Sigma P(x)*log_2(P(x))$
    - $H(X) = -(0.667*log_2(0.667) + 0.333*log_2(0.333)$
    - $H(X) = 0.918$

Therefore, the Shannon entropy of the given sequence of IP addresses is approximately 0.918 bits. This value represents the average amount of information or uncertainty associated with each IP address in the sequence.

In this paper, we will use a scaling factor to normalize the entropy to a value of 1 for fully randomized distribution. The scaling factor for Shannon entropy is:

$$H(X) = -\sum_{N} P(x) log_2(P(x))/log_2(N)$$

This involves determining the frequency or probability of each unique value in the feature and using Shannon's entropy equation to compute the entropy value.

To define the time interval for calculating entropy from network traffic, we need to consider the specific context and requirements of the analysis. Here are some factors to consider when determining the time interval:

- Granularity: Determine the level of detail you need in your analysis. Do you want to calculate entropy for each packet, per second, per minute, or per a larger time window? The choice depends on the nature of the traffic and the insights you seek.
- Traffic Patterns: Consider the characteristics of the network traffic you're analyzing. If there are distinct patterns, such as periodic bursts or variations in traffic, you may want to align the time interval with those patterns to capture meaningful entropy changes.
- Real-Time vs. Offline Analysis: Decide whether you want to calculate entropy in real-time or analyze historical traffic. Real-time analysis typically involves smaller time intervals, while offline analysis can involve larger time windows for aggregating and processing data.
- Time Sensitivity: Determine how quickly you need to detect changes or anomalies. Shorter time intervals provide more timely detection but may result in a higher computational load due to frequent entropy calculations.
- Trade-off between Accuracy and Overhead: Consider the balance between the accuracy of entropy measurements and the computational overhead required. Smaller time intervals provide more accurate and fine-grained results but may require more computational resources.

Instead of using non-overlapping intervals, where each interval is independent of the others, overlapping intervals allow for capturing temporal relationships and detecting changes more effectively. Implementation of overlapping intervals can be performed as follows:

- Determine the window size: Define the duration or number of packets that will be considered in each interval. For example, we choose a window size of 250 records.
- Determine the overlap size: Decide on the amount of overlap between consecutive intervals. It can be a fixed percentage or a fixed number of packets. For example, we choose to have a 50% overlap.
- Slide the window: We start with the first interval and calculate the entropy within that window. Then, we slide the window by the overlap size and calculate the entropy for the next interval. Repeat this process until we have covered the entire duration of the traffic.
- Aggregate the results: Depending on your analysis requirements, we can aggregate the entropy values from each interval, compare them, or use them for further anomaly detection or pattern recognition purposes.

By overlapping the intervals, one can capture temporal dependencies and detect changes or anomalies that may span across multiple intervals. This approach allows for a more continuous and fine-grained analysis of the network traffic.

It's important to note that overlapping intervals increase the computational complexity as you perform entropy calculations for multiple overlapping windows. Therefore, you should consider the trade-off between the level of detail required and the computational resources available for your analysis. Adjusting the window size, overlap size, and monitoring the impact on the results can help strike a balance.

Results of Dataset Analysis

To validate the proposed approach, we will focus on analyzing dataset with entire internet traffic. Particularly, we have chosen a labelled dataset UNSW-NB 15, that contains the raw network packets, which were created by the IXIA *PerfectStorm* tool in the Cyber Range Lab of University of New South Wales Canberra.

Such data constitutes a hybrid of real normal activities and synthetic contemporary attack behaviors. The *tcpdump* tool was utilized to capture 100 GB of the raw traffic.

Labels in this dataset refer to nine types of attacks, namely: *Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.*

The dataset consists of lines ordered by time and numbered from 1 to infinity. Each line contains more then 50 attributes of network traffic, separated by commas, along with a text label indicating the type of attack. If a label is *None*, it means there were no attacks during that particular timestamp. We have developed a *Python* code to read the raw data and generate charts for all nine types of attacks.

Fuzzer attack

A *fuzzer*, in the context of network attacks, is a type of tool or software used to identify vulnerabilities in network protocols, applications, or systems. Fuzzing, also known as fuzz testing, is the process of sending malformed or unexpected input data to a target system to observe its response and potentially uncover software bugs, security vulnerabilities, or crashes.

A fuzzer network attack involves using a fuzzer tool to systematically generate and send specially crafted network packets or messages to a target network application or protocol implementation. The purpose is to test the robustness and security of the target system by attempting to trigger unexpected behavior or exploit weaknesses.

Table 2: Description of specific attributes of the dataset

N	Name	Description	Type
1	sbytes	Source to destination transaction bytes	Integer
2	dbytes	Destination to source transaction bytes	Integer
3	service	http, ftp, smtp, ssh, dns, ftp-data, irc	String
4	src ip	Source IP address	nominal
5	sport	Source port number	integer
6	proto	Transaction protocol	nominal

We selected six attributes from our dataset (see *Table 2*) and calculated the Shannon entropy for these specific features.

To begin, we generated a chart based on the attack label. If the label corresponds to *Fuzzers*, we plotted a point with a value of 1; otherwise, we plotted a point with a value of 0 (shown as a dashed line in *Figure 1*). For our analysis, we defined the number of packets as 250 records, which were considered in each interval. This same number was used to calculate the entropy for the *sbytes*, *dbytes*, and *service* attributes.

The charts in *Figure 1* depict the entropy values for these three different attributes of the dataset. From the figure, it is evident that the *service* attribute provides the best parameter for identifying the *Fuzzers* attack.

Figure 1: Experiment with Fuzzer. The chart for the fuzzer network attack is dashed

Reconnaissance attack

Reconnaissance, in the context of network attacks, refers to the process of gathering information about a target network, system, or organization to gain knowledge that can be used in future attacks.

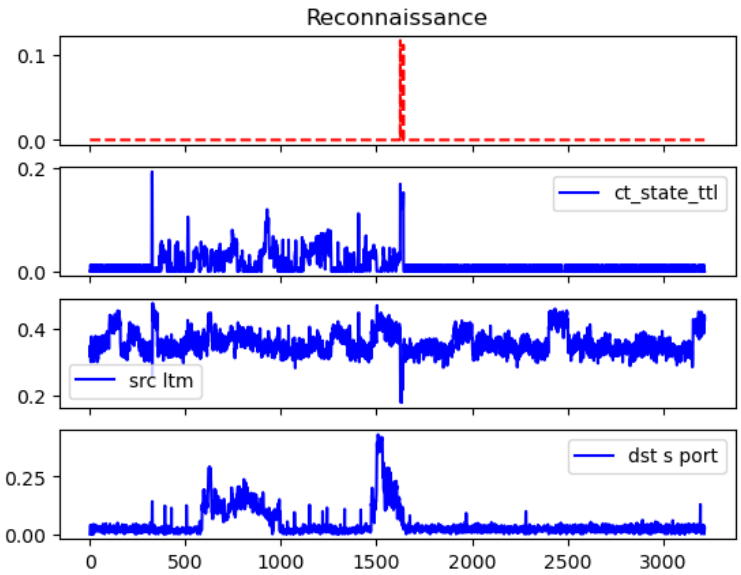

Figure 2: Experiments with Reconnaissance. The chart for the reconnaissance network attack is dashed

It is an essential step for attackers to understand the target's infrastructure, identify potential vulnerabilities, and plan their subsequent actions. A reconnaissance network attack, also known as network reconnaissance or network scanning, involves actively probing a target network to gather information about its hosts, services, and network topology. The goal is to collect data that can be used to assess the security posture, identify potential entry points, or map out the target network for further attacks.

The entropy values for the next three attributes of the dataset for *Reconnaissance* attacks are illustrated in *Figure 2*. In the graph, the first dashed line represents the *Reconnaissance* label. A point with a value of 1 is plotted if the label corresponds to *Reconnaissance*, and 0 otherwise.

Similar to our previous analysis, we considered 250 records as the number of packets within each interval. This same number was used to calculate the entropy for the three attributes.

From the figure, it is evident that the *src_Itm* attribute serves as the most effective parameter for recognizing *Reconnaissance* attacks.

DoS attack

A *DoS* (Denial of Service) network attack is a type of cyberattack that aims to disrupt or disable the normal functioning of a network, system, or service, rendering it inaccessible to legitimate users. The objective of a *DoS* attack is to overwhelm the target with a flood of malicious traffic or resource exhaustion, causing a denial of service for its intended users.

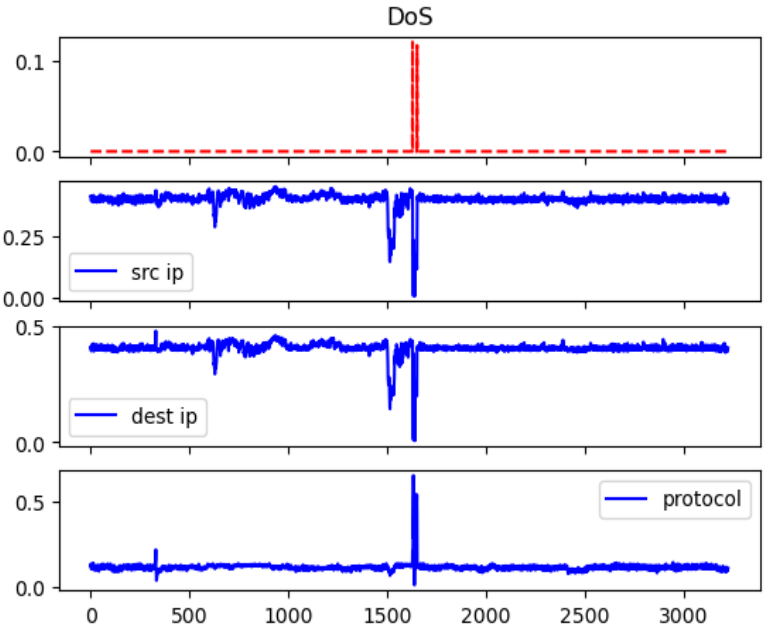

Figure 3: Experiments with DoS. The chart for the DoS network attack is dashed

In a *DoS* attack, the attacker typically exploits vulnerabilities in the target's infrastructure or services to flood the network or system with a high volume of requests or data. This flood of traffic can consume network bandwidth, exhaust system resources (such as *CPU* or memory), or overwhelm specific services, making them unresponsive or slow to respond.

The *Figure 3* showcases the entropy values for different categories of the dataset pertaining to *DoS* attacks. In the graph, the first dashed line represents the *DoS* label. A point with a value of 1 is plotted if the label corresponds to *DoS*, and 0 otherwise.

Similar to our previous analyses, we utilized a consideration of 250 records as the number of packets within each interval. This same number was used to calculate the entropy for the three attributes. From the figure, it is seen that the *protocol* attribute serves as the most effective parameter for recognizing *DoS* attacks.

Exploits attack

An *Exploits* network attack refers to the use of specific techniques or tools to take advantage of vulnerabilities in a network's infrastructure, systems, or software. The attack involves exploiting these vulnerabilities to gain unauthorized access, control, or manipulate the targeted network for malicious purposes. In an exploits network attack, the attacker leverages known or unknown vulnerabilities to launch their attack. They may use pre-existing exploits, exploit kits, or even develop their own custom exploits to target specific weaknesses in the network. The goal is to bypass security measures, compromise systems or data, or disrupt network operations.

The entropy values for different categories of the dataset related to *Exploits* attacks are visualized in *Figure 4*. In the graph, the first dashed line represents the *Exploits* label. A point with a value of 1 is plotted if the label corresponds to *Exploits*, and 0 otherwise.

Similar to our previous analyses, we considered 250 records as the number of packets within each interval. This same number was used to calculate the entropy for the three attributes.

From the figure, we see that the *protocol* attribute proves to be the most effective parameter for recognizing *Exploits* attack.

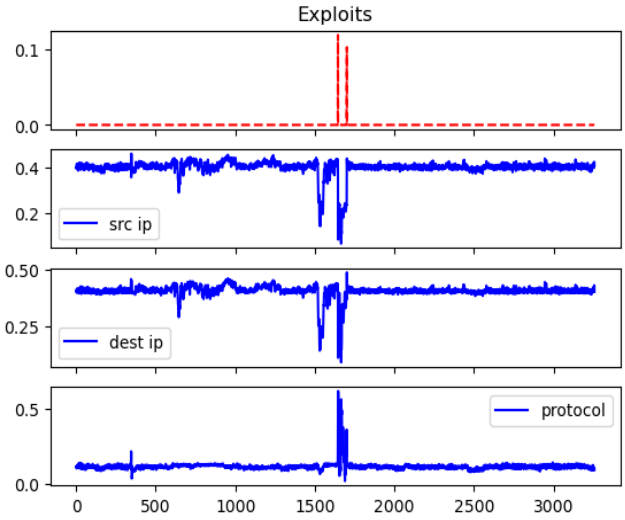

Figure 4: Experiments with Exploits. The chart for the exploits network attack is dashed

Shellcode attack

A *Shellcode* network attack involves exploiting vulnerabilities in software or systems to inject and execute malicious code, known as shellcode, on a target system via a network connection. Shellcode is typically written in low-level languages and is designed to provide the attacker with unauthorized access, control, or the ability to manipulate the compromised system.

From the figure, it appears that the *service* attribute may be the most effective parameter for recognizing *Shellcode* attack.

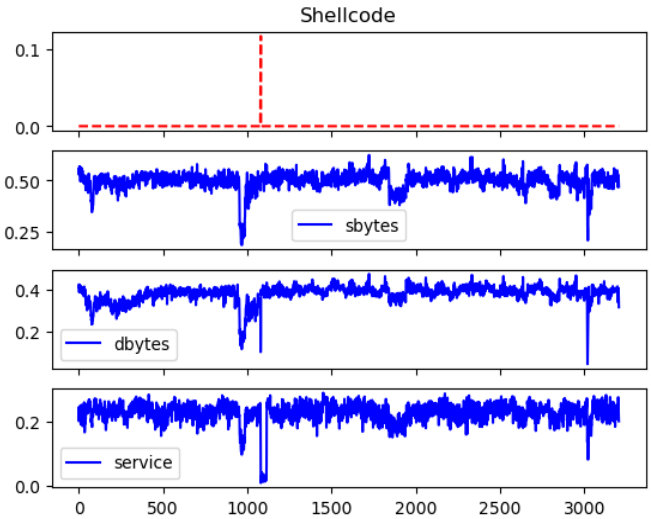

Figure 5: Experiment with Shellcode. The chart for the shellcode network attack is dashed

Generic attack

A generic network attack refers to a broad category of network attacks that do not fall into specific or predefined attack types. It encompasses a wide range of attack techniques and strategies used to exploit vulnerabilities, gain unauthorized access, disrupt network operations, or compromise the security of network systems and data. From the figure, we may conclude that the *sbytes* and *service* attributes may be used for recognizing *Generic* attack.

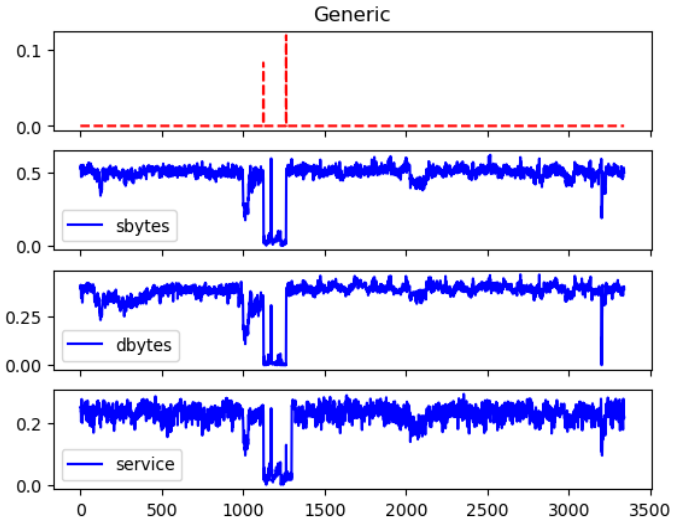

Figure 6: Experiment with Generic. The chart for the Generic network attack is dashed

Analysis attack

Attacks involving extensive analysis refer to attacks that involve extensive analysis of the target network or systems. These attacks may use reconnaissance techniques, vulnerability scanning, or network mapping to gather information about the target infrastructure. The purpose of such analysis could be to identify potential weaknesses, vulnerabilities, or entry points for subsequent exploitation. From the figure, the *protocol* attribute appears to be the most effective parameter for recognizing *Analysis* attack.

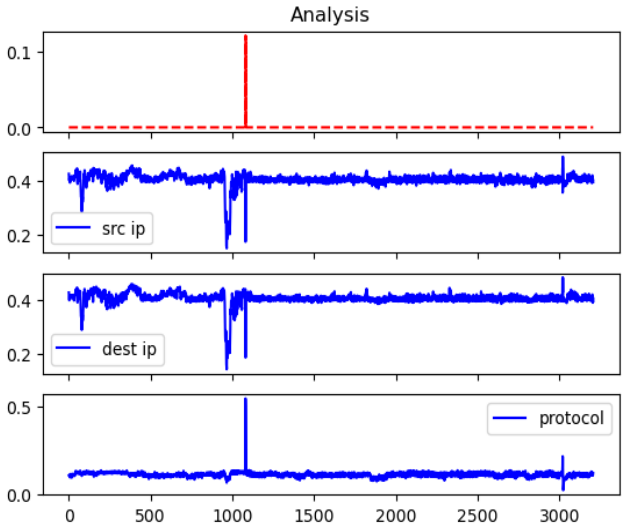

Figure 7: Experiment with Analysis. The chart for the Analysis network attack is dashed

Worms attack

Worms is a type of self-replicating malicious software that spreads across networks, exploiting vulnerabilities in systems and infecting other connected devices. It is a specific form of malware designed to propagate autonomously without requiring user intervention. From the figure, we may choose the *service* attribute to help identify *Worms* attack.

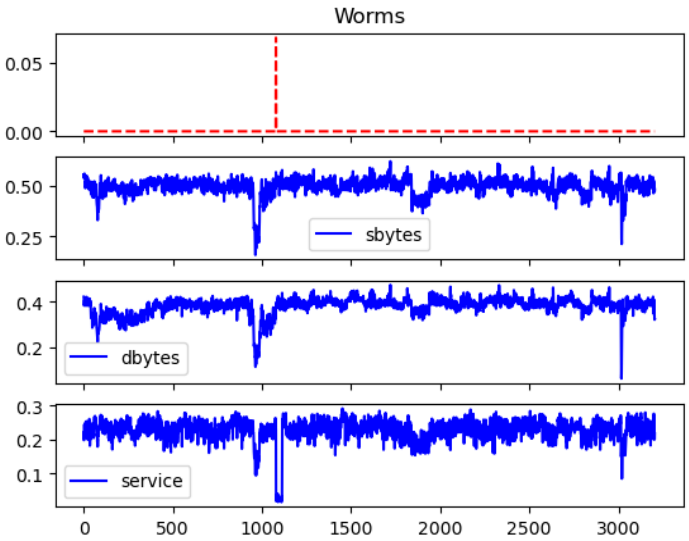

Figure 8: Experiment with Worms. The chart for the Worms network attack is dashed

Backdoors attack

A backdoor network attack refers to the unauthorized insertion or use of a hidden entry point or mechanism within a computer system, application, or network infrastructure. This backdoor provides an alternative means of access and control for attackers, bypassing normal authentication and security measures. Figure below suggests that any of three considered attributes well recognize a *Backdoors* attack.

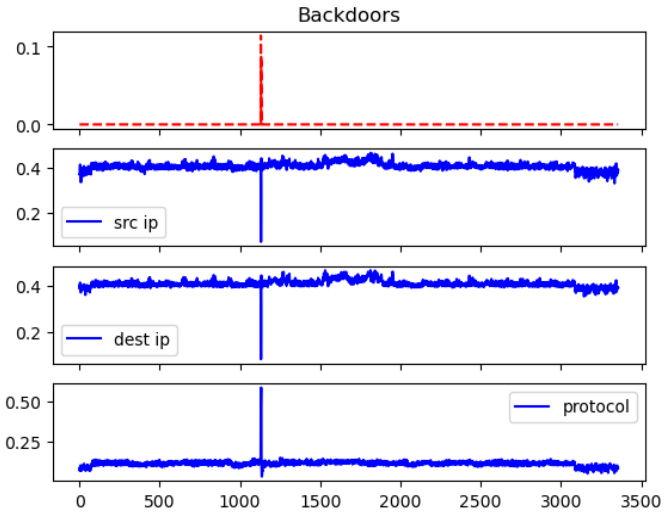

Figure 9: Experiment with Backdoors. The chart for the Backdoors network attack is dashed

Discussion

The entropy-based method can provide scalability in terms of its ability to handle large-scale networks, high data volumes, and complex traffic patterns. The scalability of the entropy-based method can be attributed to several factors:

- ***Aggregated Data Usage****:* The method can utilize aggregated data, such as Net-flow records or summarized traffic statistics, which allows for efficient processing of large volumes of network traffic data. Data aggregation reduces the overall data size and complexity, making it more manageable for analysis.
- ***Computational Efficiency****:* The calculation of entropy for a given network traffic feature is computationally efficient and can be performed relatively quickly, enabling real-time or near-real-time analysis of network traffic.
- ***Flexibility in Feature Selection****:* The method is flexible in selecting different network traffic features to analyze. This allows for focusing on specific features relevant to the detection of anomalies, which can help optimize the analysis process and adapt to the characteristics of different networks.
- ***Scalable Algorithms:*** The underlying algorithms used for entropy calculation and anomaly detection can be designed to scale well with increasing data volumes and network complexity. This ensures that the method remains effective even as the size and complexity of the network grow.

The actual scalability of the method, however, may depend on factors such as computational resources, algorithm design, and the specific implementation used.

Entropy method is just one of the techniques in a comprehensive security strategy. It should be used in conjunction with other security mechanisms, such as firewalls, intrusion detection systems, antivirus software, and regular security updates, to provide a robust defense against Internet attacks. So, using the entropy method to find attacks in network traffic involves analyzing the entropy values of various attributes or features to identify potential anomalies or patterns that may indicate malicious activity.

By applying the entropy-based method to network traffic data and analyzing the entropy values of various features or attributes, it is possible to identify deviations from normal or expected patterns. Unusual or abnormal entropy values may indicate the presence of anomalies, including potential attacks.

As a prospective direction for future research, we would like to note that the entropy-based method itself is not specifically designed to recognize or classify specific types of attacks. Instead, it is a statistical technique that measures the randomness or unpredictability of data. However, the calculated entropy values can provide insights that contribute to the identification or detection of anomalies in network traffic, which may include different types of attacks.

To classify the specific type of attack, additional analysis and techniques are typically employed in conjunction with the entropy-based method. This can involve correlating the observed anomalies with known attack patterns or utilizing machine learning algorithms to classify and categorize the anomalies based on their characteristics.

References

Baez, J.C., Fritz, T., & Leinster, T. (2011). A Characterization of Entropy in Terms of Information Loss. *Entropy*, *13*, 1945–1957.

Bashurov, V., & Zhigalov, V. (2021). Smoothed particle hydrodynamics method for numerical solution of multidimensional filtering problems. *J. Phys.: Conf. Ser. 2099 012001*

Berezinski, P., Jasiul, B., & Szpyrka M. (2015). An entropy-based network anomaly detection method. *Entropy 17 (4),* 2367–2408.

Bojovic, P. D., Basicevic, I., Ocovaj, S., & Popovic, M. (2018). A practical approach to detection of distributed denial-of-service attacks using a hybrid detection method. *Computers and Electrical Engineering 73*, 84-96.

Denning, D.E. (1987). An intrusion-detection model. *IEEE Trans. Softw. Eng.*, *13*, 222–232.

Gu, Y.; McCallum, A.; Towsley, D. Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation. Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC ’05), Berkeley, CA, USA, 19–21 October 2005; pp. 32–32. Lakhina, A., Crovella, M., & Diot, C. (2005). Mining Anomalies Using Traffic Feature Distributions. *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 217–228.

Lee, W., & Xiang, D. (2001). Information-theoretic measures for anomaly detection. *Proceedings of 2001 IEEE Symposium on Security and Privacy*, 130–143.

Ranjan, S., Shah, S., Nucci, A., Munafo, M., Cruz, R., & Muthukrishnan, S. (2007). DoWitcher: Effective Worm Detection and Containment in the Internet Core. *Proceedings of 26th IEEE International Conference on Computer Communications, Anchorage, AL*, 2541–2545.

Sarhan, M., Layeghy, S., Moustafa, N., & Portmann, M. (2020). NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. *WiCON 2020: In Big Data Technologies and Applications*, 117–135.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal, 27(3)*, 379-423.

Timčenko, V., Gajin, S. (2020). Time-series entropy data clustering for effective anomaly detection. *Proceedings of the 10th International Conference on Information Society and Technology, Information Society of Serbia, ISBN 978-86-85525-24-7*, 170-175.

Moustafa, N., Hu, J., & Slay, J. (2019). A holistic review of Network Anomaly Detection Systems: A comprehensive survey. *Journal of Network and Computer Applications 128*, 33-55.