ADAPTIVE CUMULATIVE ENTROPY THRESHOLD: A NOVEL APPROACH TO DDOS ATTACK DETECTION IN IOT DEVICES AND SMART HOMES SYSTEMS

Amit Kumar Jaiswal[1]

DOI: 10.21681/2311-3456-2025-5-162-171

Abstract. *The rising prevalence of smart home systems in everyday life, attacks such as cyber flooding on these interconnected devices have become critical. The present research talks about the innovative model using adaptive threshold, which applies cumulative entropy analysis of time series data to detect and mitigate flood attacks more effectively in the smart home environment. The model sets dynamic thresholds adaptable to changes in data fluctuations in real-time by utilizing cumulative entropy, a measure that identifies the unpredictability and variance of network traffic patterns. Advanced machine learning techniques will be further explored to refine the threshold process that will eventually lead to higher accuracy in detecting anomalies. In fact, essential factors including temporal patterns, types of protocols, and actions of users will be analyzed concerning their impact on objective metrics. Research aims at validating proposed adaptive threshold framework effectiveness in response toward significantly reducing false positives while improving responsiveness against emerging threats; hence contributing overall resilience of smart-home systems under flood attacks towards detected attacks. Anterior work shall focus on adapting algorithms and exploring scalability over diverse smart home architectures as an extension of this work. Research also intends to tackle questions linked with data privacy as well as system efficiency.*

Keywords: *Adaptive Threshold, Cumulative Entropy, Time Series Analysis, Flood Attack Mitigation, Smart Home Security, Anomaly Detection, Network Traffic An al ysis, Temporal Data Patterns.*

1. Introduction

The increasing penetration of smart home systems in everyday life has brought about the enormous advantages of convenience and automation. However, this evolutional process also expose the shortcomings by vulnerability issues primarily focusing on possible cybersecurity threats, one of which is the Distributed Denial of Service (DDoS) attack that can immensely menace the operation of smart devices with potential risks to users' safety and privacy. Therefore, it is extremely imperative to develop efficient countermeasures to timely detect and react upon this type of cyberattack [1,2]. A promising direction towards ameliorating smart home security is to adopt adaptive thresholding techniques relying on cumulative entropy based time series analysis. Entropy, a fundamental concept taken from inFormation theories, measures uncertain or random characteristics in given dataset. In network traffic analysis domain for instance, monitoring inherent entropy regimes facilitates distinguishing normal patterns from unusual behaviors underlying DDoS attacks. By employing cumulative entropy measures, researchers can develop adaptive methods that adjust thresholds dynamically based on realtimedata, thereby improving detection accuracy and reducing false positives [3].

2. Background on Topic

In recent years, the increasing popularity of smart home systems has raised serious concerns about cybersecurity, particularly related to distributed denial of service (DDoS) attacks that can severely overflow network resources, incapacitate smart devices, and pose threats to user security and privacy [4,5]. Therefore, defense mechanisms must be put into places to ensure the resiliency of smart home systems towards this type of attack. A promising solution can be perceived by adopting adaptive thresholding techniques based on entropy measures to analyze time-series data that originate from network traffic [13,14]. Entropy is a measure that represents the uncertainty or randomness when characterizing a certain data-set. Specifically, utilizing entropy within the context of network traffic an alysis al lows re searchers to determine how different normal network requests (which are considered as non-malicious) are from exploitative counterparts illustrating DDoS traits (which are deemed malicious)[15][17]. Researchers have consistently shown how cumulative entropy measures enable detection algorithms to become more adaptive and accurate due to their capability in adjusting dynamically with respect to current network conditions [18], [19].

1 Amit Kumar Jaiswal, Ph.D., Student/Researcher, Department of Radio Engineering and Cybernetics, Moscow Institute of Physics and Technology (MIPT). E-mail: dzhaisval.a@phystech.edu

3. Related Works

Several works have studied thresholding techniques for anomaly detection innetwork traffic. For example, Sahoo and Arora (2004) proposed a thresholding technique based on two-dimensional Renyi entropy that achieved a much better segmentation performance in image processing applications, indicating the potential of entropy-based techniques to discriminate normal patterns from anomalies [8].

Dragos et al. (2020) investigated some entropy-based metrics for uncertainty evaluation in Bayesian networks designed for cyber threat detection and concluded that the entropy measurement is important both in performance estimation of a model and as an added value to decision-making under uncertainty [5].

This work paves the way for applying two-pronged on-line entropy based defense mechanism at DDoS attack by defending attack traffic in path [7].

Recent improvements in adaptive thresholding techniques show the potential of such methods in many domains. A machine learning-aided entropy-based anomaly detection framework for dynamic network adaptations was proposed by Timcenko and Gajin (2021) [6].

They elevate the need for adaptivity that relies on threshold adjustments by real-time data analysis, which is essential in combating DDoS attacks in smarthomes [9].

The use of cumulative entropy in time series analysis has been presented in some previous works. In particular, some researches have focused on using cumulative residual entropy as a risk measure and they have proven that it is a useful tool in many different situations. This is consistent with our research goal to apply cumulative entropy for adaptive thresholding in the analysis of time series data of DDoS attacks [10,11]. Zhang et al. [12] conducted a comprehensive survey on network anomaly detection frameworks based on kinds of entropy measures such as Shannon and Renyi entropies and concluded that using many kinds of features can improve the accuracy of model to against various anomalies types.

4. Detailed Raw Dataset Description Used in this Research

The UCM_FibIoT2024 dataset gathers substantial data to understand better Distributed Denial of Service (DDoS) attacks against smart home central control units, namely the Fibaro Home Center 3. This dataset records many types of DDoS assaults, such as TCP SYN floods, I CMP fl oods, an d HT TP floods, to provide light on how they influence the operation and availability of IoT devices [16]. Data was collected on a local network using the hping3 tool for SYN and ICMP flood attacks, and the LOIC tool for HTTP flood assaults. Wireshark software was used to gather network traffic, and the information is available in PCAP and CSV formats for future analysis. The collected data includes critical details such as timestamps, source and destination IP addresses, protocols, packet lengths, and port numbers [16]. The major purpose of this dataset is to make it easier to simulate and analyze DDoS attacks on smart home central control units, hence serving as a resource for cybersecurity and IoT device protection researchers. Researchers can discover attack patterns, understand the dynamics of various forms of DDoS attacks, and design effective mitigation systems by inspecting network traffic records and packet captures [16]. The collection is structured to provide comprehensive logs for each attack, such as start and finish timings, frame numbers, and the total number of assault packets. For simplicity of usage, the data is sorted into folders, and the SYN flood attack data is further split by the ports targeted (80, 443, and 500)[16]. The UCM_FibIoT2024 dataset serves as a profitable instrument for analyzing and creating resistance against DDoS attacks on IoT gadgets. It gives a viable asset to analysts and cybersecurity experts to successfully reenact, analyze, and moderate DDoS attacks [16]. For more information about the dataset, refer to the UCM_FibIoT2024 dataset is available at https:// doi.org/10.17632/p42xjtv8pv.1. However, for this study, we will be only using HTTP flood and ICMP flood data.

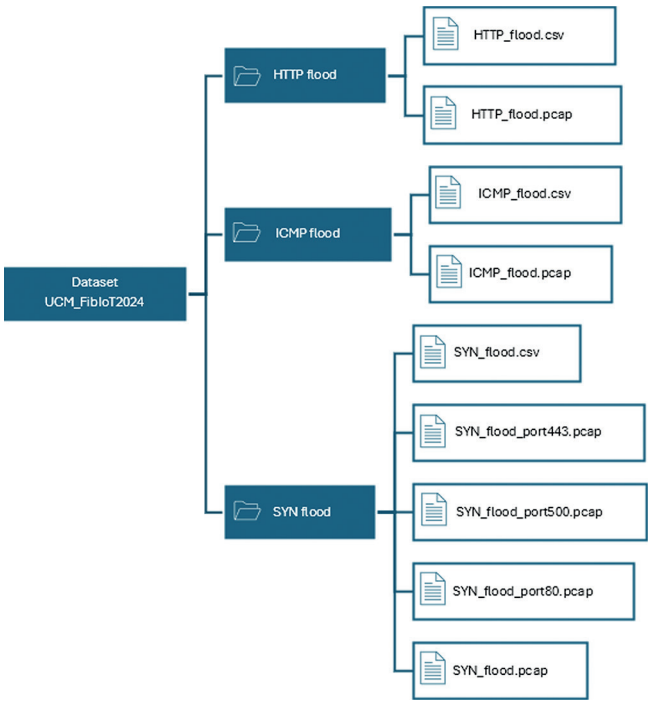

Figure 1: Flow Chart of the Raw Data Structure Files capture [17]

Table 1.

Overview of Raw CSV Dataset Columns[16]

Column Name	Description
No.	Frame number.
Time	Date and time of capture (dd.mm.yyyy hh:mm:ss).
Source	Source IP address of the packet.
Destination	Destination IP address of the packet.
Protocol	Protocol type identifying the network protocol used For each packet.
Length	Packet length in bytes.
Source port	Source port of the packet.
Destination port	Destination port of the packet.

5. Research Objective

We note here that the previous authors have utilized several thresholding techniques in different fields and datasets to research different purposes. However, no authors have used our novel approach, that is, network traffic detection with time series analysis using the cumulative entropy method with thresholding, to detect such attacks most likely in DDoS on smart home systems and on IoT devices specifically, which will ultimately help future research scope growth.

6. Methodologies Used in this research

6.1 Raw Data Preprocessing

Table 1 represents column names and descriptions of these featured columns in raw data preprocessing plays a very crucial and vital role and methods for this research. There were many steps performed on raw dataset for data preprocessing. First step was data cleaning, in this step we identified and rectified errors, inconsistencies, and inaccuracies in the raw dataset. We found source port and destination port were having huge number of missing, inconsistent and inaccurate values. We used techniques like handling missing data and removing duplicates to clean the dataset. Later we analyzed and removed source port and destination port due to their high inaccuracy and irrelevancy to this research outcome.

6.2 Exploratory Data Analysis

In this section, we will demonstrate the exploratory data analysis performed by us to analyze dataset more deeply. Firstly, we applied many data analysis codes and functions, we checked the dataset size, the dataset description in terms of counts, min and max values, and different percentiles of the dataset in each column. There are two types of attack files used, one is HTTP flood, a type of attack that targets web servers by overwhelming them with high HTTP requests. Another file is ICMP Flood, an attack that sends a large number of ICMPv6 packets (often ping requests, but in this research, data hping was used for a more aggressive attack) to a target, consuming bandwidth and resources. In Table 2, we found that HTTP flood has a higher number of packet counts but the lowest frequency, whereas ICMPv6 flood has a higher frequency in comparison to HTTP flood. These findings results in ICMPv6 floods having a higher effect in the consumption of storage and bandwidth uses, which can result in a DDoS attack in IoT devices and smart home systems.

In Table 3, we analyzed and found that IP address 10.0.1.22 has the highest number of traffic as an IP source. HTTP floods have 132 unique values whereas, ICMPv6 have the highest number of uniqueness in the traffic. As a finding, higher uniqueness in traffic are to have unknown distributed sources which is to result into a DDoS attack (Distributed Denial of service attack) in the network.

Table 2.

Summary of HTTP and ICMPv6 Flood Protocol Features

Feature	HTTP Flood	ICMPv6 Flood
Count	22,780,665	11,203,031
Unique	121	26
Top Protocol	TCP	ICMPv6
Frequency	8,060,484	11,172,532

Table 3.

Overview of HTTP and ICMPv6 Flood Source Features

Feature	HTTP Flood	ICMPv6 Flood
Count	10,799,707	11,203,031
Unique	132	11,214,481
Top Source	10.0.1.22	10.0.1.22
Frequency	4,859,392	6,602

In table 4, we analyzed and found the destination IP address request traffic. Also we found that the count of packet traffic had the same count. We discovered ICMPv6 requests, indicating a slightly higher volume compared to HTTP floods, with much higher frequency than source IP requests. Finally higher percentage of requests indicates more aggressive and sustained attack on target.

Table 4.

Overview of HTTP and ICMPv6 Flood Destination Features

Feature	HTTP Flood	ICMPv6 Flood
Count	10,799,707	11,203,031
Unique	127	1,747
Top Destination	10.0.1.22	10.0.1.22
Frequency	5,911,972	11,174,499

7. Feature Engineering

We did feature engineering task to enrich the features and applied stratified sampling technique on HTTP flood dataset and ICMPv6 flood dataset, stratified sampling, which involves dividing the population into subpopulations (strata) based on one or more common attributes; strata membership is determined by some factor(s) that are hypothesized to be related to the process being measured, such as class labels- to reduce business and increase the performance of model learning and testing.

The UCM_FibIoT2024 dataset records are enormous (millions of records), processing whole CSV file at once requires more time-consuming and computation resources. Therefore, we applied stratified random sampling to HTTP flood and ICMPv6 flood CSV fles for our experiment. We have considered the sample of frac = 0.02 for each file. In algorithm 1, we have demonstrated pseudo code representation to our code and method used on stratified sampling of high volume dataset. We have also used the time-based feature engineering method to extract each feature from the time column in separate columns (year, month day, hour, minutes, seconds, microseconds).

Algorithm 1 Sampling from Dataset by Protocol

```
BEGIN
    DATASET ← Load dataset
    GROUPED_DATASET ← GROUP DATASET BY 'Protocol'
    SAMPLED_DATA ← []
    For each GROUP in GROUPED_DATASET DO
        SAMPLE ← SAMPLE 20% FROM GROUP
        SAMPLED_DATA.APPEND(SAMPLE)
    END For
    FINAL_SAMPLED_DATA ← Convert SAMPLED_DATA TO DataFrame
END
```

7.1 Cyclical Time Encoding

Further we used Cyclical Encoding technique to create more features on time. To handle the cyclical nature of time (e.g., hours in a day, days in a week), we have converted time into a circular representation using sine and cosine functions. Let hour be the hour of the day (in 24-hour format). The angle can be calculated as:

$$\text{angle} = \left(\frac{\text{hour}}{24}\right) \times 2\pi \quad (1)$$

The sine and cosine transFormations are defined as follows:

$$X = \sin(\text{angle}) = \sin\left(\left(\frac{\text{hour}}{24}\right) \times 2\pi\right) \quad (2)$$

$$Y = \cos(\text{angle}) = \cos\left(\left(\frac{\text{hour}}{24}\right) \times 2\pi\right) \quad (3)$$

These transformations allow the model to capture the cyclical nature of time, effectively treating 23 : 00 and 00 : 00 as close to each other.

7.2 Seconds Since Epoch

The term seconds since the epoch represents the representation of time by counting the aggregate number of seconds elapsed, from a particular starting point in time, is called the epoch. Seconds since epoch are prominently used to detect and analyze Distributed Denial of Service (DDoS) attacks. Here, we highlight its usage concerning time-stamping and network traffic monitoring. In DDoS detection systems, every packet of network traffic can be timestamped in seconds since epoch format to keep the record of exactly when it was received. Accurate timestamps can be used to track trends, such as a traffic spike over an extended period of time indicating a possible DDoS attack. Using the timestamps from incoming packets detection systems can measure the number of packets or amount of traffic within a given (40 seconds) window, if too many packets arrive within that time span. Systems can compare the number of packets received in an epoch to a threshold value and then generate an alert if the packet total is above a pre-determined threshold baseline, indicative of DDoS. So, we created new feature called seconds since epoch. To do this we have created a mathematical formulae calculation to calculate seconds since epoch on each packet traffic. Let T represent the timestamp from the data sample, and let T_0 denote the epoch time defined as:

$$T_0 = \text{Timestamp}(2024, 1, 1, 0, 0, 0). \quad (4)$$

The Seconds Since Epoch can be calculated as follows:

$$\text{SecondsSinceEpoch} = (T - T_0) \div 1 \text{ second}. \quad (5)$$

Where:

- T = ['Time'] (the dataset timestamp features);
- T_0 = Timestamp representing the epoch;
- The division by 1 second effectively converts the time difference from a Timedelta object into an integer representing seconds.

The formula presented provides a clear method to calculating Seconds Since Epoch, which is fundamental in various applications involving time series analysis and event logging.

8. Mathematical Formulations For calculating entropy and detecting anomalies in packet data

8.1 Entropy Calculation

The Shannon entropy $H(X)$ for a discrete random variable X is defined as:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i) \quad (6)$$

Where:

- p_i is the probability of occurrence of the i-th outcome.
- n is the total number of distinct outcomes.

In this context, the entropy is calculated for packet lengths over a rolling window of size 10:

$$H(\text{Length}_{\text{window}}) = -\sum_{j=1}^{m} p_j \log_2(p_j) \quad (7)$$

Where:

- m is the number of distinct packet lengths in the current window.

8.2 Cumulative Entropy Calculation

The cumulative entropy at time t can be expressed as:

$$C(t) = -\sum_{i=1}^{t} H(\text{Length}_i). \quad (8)$$

Where:

- $C(t)$ is the cumulative entropy up to time index t.
- $H(\text{Length}_i)$ is the entropy calculated For packet lengths at time index i.

8.3 Anomaly Detection

Anomaly detection is performed using a simple thresholding method. The threshold T is defined as:

$$T = \mu + 3\sigma. \quad (9)$$

Where:

- μ is the mean of the cumulative entropy values.
- σ is the standard deviation of the cumulative entropy values.

An anomaly occurs when:

$$A(t) = \begin{cases} 1 & \text{if } C(t) > T, \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

Where:

- $A(t)$ indicates whether an anomaly is detected at time index t.

9. Experiments

We extracted time components from time feature. We have written our own code for date-time feature with (Year, Month, Day, Hour, Minutes, Seconds, Microseconds). We used python prebuilt library called DATETIME. In algorithm 2 Label of the algorithm is "Extract Time Components from Date-time which tells us that this algorithm is responsible for extracting specific-time-related features from date-time. In the beginning of the algorithm there is a comment saying that: dataset is a data-structure (like table or Data Frame) that has a column called Time which contains date-time values. A separate loop goes throw all rows in dataset one by one and extract time column value and stores it into new created separate column named (Year, Month, Day, Hour, Minutes, Seconds, Microseconds). The algorithm is finished with an «END» statement then. In general, this algorithm is designed to extract all the possible individual time components including year, month, day, hour, minute, second and microsecond of a date-time object separately in order to analyze or process them individually. It can be very helpful for data analysis purposes when we may want to analyze/visualize some patterns at year/month/day/hour/minute/second/microsecond level or want to filter/group by these individual time components etc. while performing some machine learning tasks over timeseries like feature engineering.

We have used mathematical formulae for sine and cosine calculations for hour, for each row, it retrieves the value of hour and assigns it to hour value. Then it fetches the value of hour and stores it in an hour variable. It subsequently calculates sine and cosine of this hour value using above mentioned Formulas 1, 2 and 3 as before. By doing these calculations, it maps the respective hour into a form of cyclic representation which helps to present time-concept to the models. Sine and cosine calculations for month, similarly, it fetches the value of month and stores it in a month variable. Then, it calculates sine and cosine for month value with formulae similar to hours but divided by 12. The algorithm ends with an «END» statement representing that all calculations have been made here. This algorithm essentially performs conversion of cyclical time data (hours & months) into simple sine-cosine way. This whole code is classically inspired from https://en.wikipedia.org/wiki/Besselpublication by Don E. Knuth which approximates values of sin() & cos().

In algorithm 3, the algorithm name is «Calculate Seconds Since Epoch». The algorithm defines the constant EPOCH_TIMESTAMP as a string, representing this epoch: «2024-01-01 00:00:00». It loops through each data row assuming that there is a column with date-times Time in the dataset. For each row, it assigns the current timestamp from column Time to CURRENT_TIMESTAMP. It calculates the difference CURRENT_TIMESTAMP minus EPOCH_TIMESTAMP as TIME_DIFFERENCE. This difference represents how much time passed between the epoch and that timestamp. The algorithm converts this value then into seconds by dividing it by one second (which might be implicit for many programming languages if you handle simply date objects). The resulting number of seconds since the epoch SECONDS_SINCE_EPOCH, it saves in an additional column named SecondsSinceEpoch, defined in memory for the dataset data_sample at corresponding row. Finally, there is an «END» after which we know that all these operations end. The main aim of converting date-time values into such a standardized numeric Format (seconds since epoch) is facilitating their usage for various operations and especially mathematical analyses during which we want to help computer somehow understand how timestamps are big/small

or older/newer than other timestamps. For example when comparing them during some model learning.

Algorithm 2 Extract Time Components from Datetime

```
BEGIN
// Assume data_sample is a data structure (like a table or DataFrame) with a column 'Time' of datetime type
// Extract year from the 'Time' column
for each row in data_sample do
    row['Year'] ← EXTRACT_YEAR(row['Time'])
end for
// Extract month from the 'Time' column
for each row in data_sample do
    row['Month'] ← EXTRACT_MONTH(row['Time'])
end for
// Extract day from the 'Time' column
for each row in data_sample do
    row['Day'] ← EXTRACT_DAY(row['Time'])
end for
// Extract hour from the 'Time' column
for each row in data_sample do
    row['Hour'] ← EXTRACT_HOUR(row['Time'])
end for
// Extract minute from the 'Time' column
for each row in data_sample do
    row['Minute'] ← EXTRACT_MINUTE(row['Time'])
end for
// Extract second from the 'Time' column
for each row in data_sample do
    row['Second'] ← EXTRACT_SECOND (row['Time'])
end for
// Extract microsecond from the 'Time' column
for each row in data_sample do
    row['Microsecond'] ← EXTRACT_MICRO-SECOND(row['Time'])
end for
END
```

Algorithm 3 Calculate Seconds Since Epoch

```
BEGIN
EPOCH_TIMESTAMP ← "2024-01-01 00:00:00"
    For each row in data_sample DO
    CURRENT_TIMESTAMP ← data_sample ['Time'] [row]
        TIME_DIFFERENCE ← CURRENT_ TIMESTAMP -
        EPOCH_TIMESTAMP SECONDS_SINCE_ EPOCH ← TIME_DIFFERENCE // 1 second
        data_sample['SecondsSinceEpoch'][row] ← SECONDS_SINCE_EPOCH
    END For
END
```

9.1 Anomaly Detection Using Threshold and Cumulative Entropy

In algorithm 4, we experimented on sample data using cumulative entropy and different threshold values. The algorithm takes sample data as a Dataframe having several columns as input, displays the cumulative entropy and which packets are considered an anomaly. A list of required column names (required columns) is created, it consists of the attributes, for example, Length, Year, Month etc to ensure that the dataset contains all the necessary information For analysis. Then it checks if all provided columns exist in sample data if any of required column is missing from dataset, then raise Value Error with suitable message. A function calculate_entropy(data) is created to compute Shannon's entropy of given data it calculates normalized value counts of unique values in the data. It returns the entropy using the Formula mentioned in 6, 7, 8, 9, and 10 earlier in sections of this paper. A new column, PacketLengthEntropy, is created in sample data. This column stores the rolling entropy calculated over the last 10 entries of the Length column, using the previously defined f unction. The cumulative sum of the PacketLengthEntropy column is calculated and stored in a new column CumulativeEntropy. This serves as the cumulative entropy over time. Any NaN value in the CumulativeEntropy column is replaced with 0, such that subsequent calculations do not fail. The threshold to determine anomalies is computed as mean(CumulativeEntropy) + 3 * standard_deviation(CumulativeEntropy), where an anomaly represents an entry being seen after which its cumulative entropy becomes larger than this threshold. Also, another new column Anomaly among the sample dataset constructed by replicate indicating if each packet's cumulative entropy exceeds the determined threshold (TRUE for anomaly; FALSE otherwise). Finally, we print columns year, month, day, hour, minute, second, Cumulative_ Entropy and Anomaly from our sample datasets.

Algorithm 4 Anomaly Detection in Packet Data

```
BEGIN
// Input: sample_data (DataFrame containing packet data with required columns)
// Output: Display of cumulative entropy and detected anomalies
// Step 1: Define required columns
required_columns ← [Length, Year, Month, Day, Hour, Minute, Second, Microsecond, Protocols...]
// Step 2: Check if all required columns are present NOT ALL(col ∈ sample_data.columns For col in required_columns)
RAISE ValueError("Missing required columns in the dataset.")
```

8: // Step 3: Define function calculate_entropy (data)
9: **Function** calculate_entropy(data)
10: // Calculate value counts of data normalized to probabilities
11: *value_counts* ← COUNT(*occurrences* of each unique value in *data*)
12: RETURN $-\sum(p_i * \log_2(p_i + \epsilon))$ where $\epsilon = 1e - 9$
13: // Step 4: Create new column *PacketLength Entropy*
14: *sample_data*['*PacketLengthEntropy*'] ← APPLY calculate_entropy ON ROLLING WINDOW OF SIZE 10 OVER *icmp_sample_data*['*Length*'] WITH min_periods = 1
15: // Step 5: Calculate cumulative entropy
16: *sample_data*['*CumulativeEntropy*'] ← CUMULATIVE SUM OF *sample_ data*['*PacketLengthEntropy*']
17: // Step 6: Fill NaN values in Cumulative Entropy with zero
18: 18: FILL NaN VALUES IN *sample_data* ['*CumulativeEntropy*'] WITH 0
19: // Step 7: Define threshold for anomaly detection
20: *threshold* ← MEAN(*sample_data* ['*CumulativeEntropy*']) + 3 $*STD$ (*sample_data*['*CumulativeEntropy*'])
21: // Step 8: Create new column *Anomaly*
22: *sample_data*['*Anomaly*'] ←TRUE IF *icmp_ sample_data*['*CumulativeEntropy*'] > *threshold* ELSE FALSE
23: // Step 9: Display results
24: PRINT SELECTED COLUMNS (*Year*,*Month*,*Day*, *Hour*,*Minute*,*Second*,*CumulativeEntropy*, *Anomaly*)
25: **END**

10. Results and Findings

10.1 Time series analysis research findings on dataset comparing HTTP flood a ttacks and ICMP flood attacks

In this research, we have created a graph for both HTTP flood attack and ICMP flood attack IOT datasets. We used the most important length and time features indicated in the dataset. Then we compared these two graphs and we discovered that ICMP traffic was much higher in the ICMP flood dataset. In Fig. 2, we also found that in the HTTP flood dataset, the other protocol traffic was higher and stable, which indicates a much lower risk. In Fig. 3, however, in the ICMP flood dataset, the other protocol traffic was less and unstable, which indicates a much higher risk. In this time series analysis, we have also found that having higher ICMP traffic in ICMP floods would have resulted in disrupting other protocol traffic in the system, creating a traffic congestion in IOT devices and smart home systems. We also discovered a very important understanding with this research analysis that, if both ICMP flood attack and HTTP flood attack have been initiated simultaneously, if both ICMP and HTTP traffic increase simultaneously, this may suggest a multi-vector attack strategy and would have and in future can have much more higher risk of traffic congestion and will result in more successful DDoS attack.

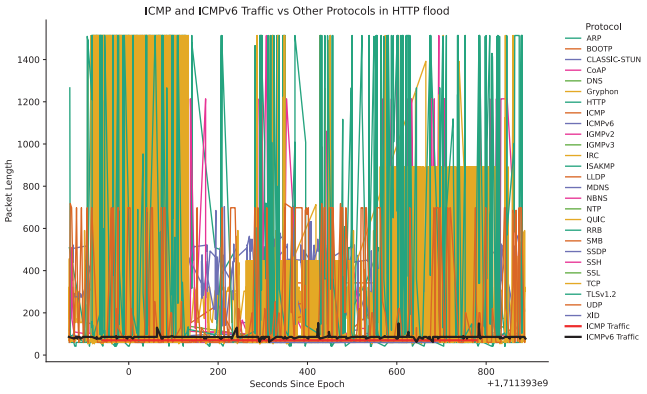

Figure 2. Time series analysis on HTTP Flooded attack traffic

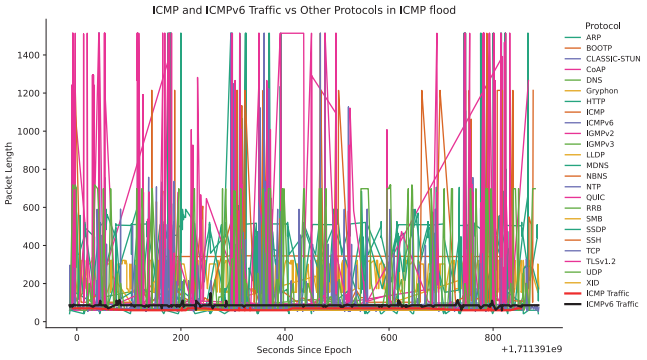

Figure 3: Time series analysis on ICMP Flooded attack traffic

10.2 Anomaly Detection using Cumulative Entropy and Thresholding Comparison

In both graphs, you will find a common chart, «Anomaly Detection Threshold». It is generally obtained by statistically calculating (taking average and standard deviation of cumulative entropy values) from the historical data. Now when your cumulative entropy exceeds that threshold, then there is an indication that an anomaly has been detected, i.e. some malicious activity (here DDoS attack) might be going on. Then, in both graphs, you can see a few explicit points marked where anomalies were detected throughout the time frame for which the analysis was done. So those peaks in Cumulative Entropy give an idea of which explicit timings during that period traffic was abnormal with respect to other timing instances.

In fig. 4, we have found that there was an unstable traffic anomaly detected after the threshold mentioned

in the ICMP flooded attack IoT dataset. We have also discovered, that there was a sudden, unstable traffic change, and an anomaly was detected after a certain point of threshold calculated. In fig. 5, however, we have investigated the HTTP flooded attack IoT dataset with the same threshold algorithm and we found no anomaly detection of any unstable traffic in comparison to the ICMP flooded attack. We have also discovered that the traffic was not able to even touch the threshold mark at any point. Hence, after all the investigation and analysis of both the graphs, we have concluded that, ICMP flooded attack poses much higher risk in IoT devices and smart systems than the HTTP flooded attack.

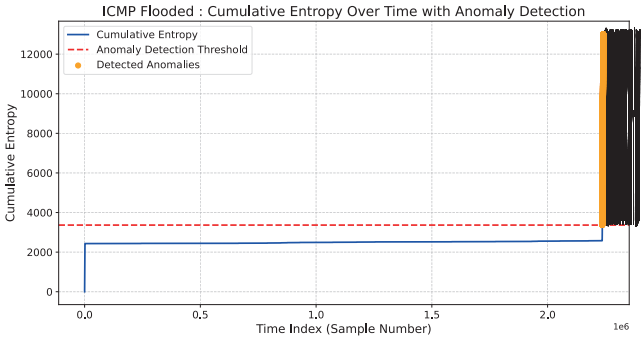

Figure 4: Anomaly Detection in ICMP Flooded attack dataset

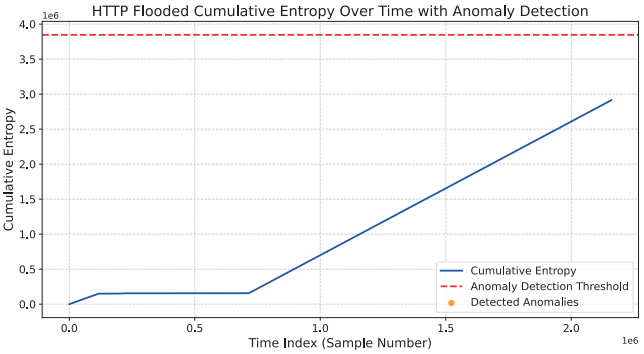

Figure 5: Anomaly Detection in HTTP Flooded attack dataset

11. Conclusion

The research findings reported in this article rovide substantial information on the impact of DDoS attacks, with an emphasis on ICMP and HTTP flood attacks in IoT contexts. We created comparison graphs of network traffic flow during different assault scenarios by analyzing time series data. Our results indicated that the level of ICMP traffic was significantly higher than HTTP in the ICMP flood dataset with the highest risk among other threats as reflected in our dataset. However, HTTP at its constant and highest rate used much traffic of other protocols within the HTTP flood dataset. Therefore, the risks associated with these types of attacks are less in general because they do not strictly use a specific protocol. By investigating cumulative entropy graphs, we noted the presence of peak sections or varying windows where anomalies occur along time signatures which function as reliable indicators to identify a potential malicious behavior and provide required knowledge on duration; how long and how network can be exposed to DDoS attacks harm. Notably, we discovered that simultaneous surges in both ICMP and HTTP traffic might indicate a multi-vector assault approach. This scenario increases the likelihood of network congestion and may lead to more effective DDoS attacks on IoT devices and smart home systems. In conclusion, our findings clearly show that ICMP flood attacks offer a far higher danger to IoT devices and smart home systems than HTTP flood assaults. The findings of this study will help to design effective countermeasures for strengthening the security of smart home systems against growing DDoS attacks. Further research is required to develop anomaly detection tools and study adaptive responses for increasing resilience to these sorts of intrusions.

12. Future Work and Ideas

In the future, we will use the same datasets, UCM_fiblo for hybrid model training on IoT devices and smart home systems datasets. This can be made possible by exploiting more powerful machine learning methods like deep learning models to improve anomaly detection accuracy in different kinds of network domains. Also, the combination with real-time big data analytics and edge computing aids in immediate threat intelligence-driven response and minimizes the latency. It is worth collaborating with IoT device manufacturers where adaptive security functionalities should be directly deployed in devices so that smart homes will have the capabilities of proactive defense against emerging DDoS attack vectors. In addition, multiple field experiments on real-world smart home deployment should be more intense, as these deployments provide a wealth of practical data that can facilitate rapid iteration on performance optimizations based on both user feedback and scientific measurements.

Acknowledgements

I am grateful to my wife, Darina Olegovna Ershova, For her professional advice and calm hand help during these challenging times. My study was helped by her expertise and vision, as well as her meticulous attention to detail. I am grateful to my department for their valuable contributions to this research, including their amazing experience of guidance. We appreciate the valuable feedback and recommendations from the faculty of Artificial I ntelligence a nd Cybersecurity at Moscow Institute of Physics and Technology (MIPT), Department of Radio Engineering and Cybernetics, (Institutsky Lane, 9, Moscow, Russian Federation, 141701). Their contributions helped shape an intellectually diverse conversation about my work.

Amit Kumar Jaiswal

АДАПТИВНЫЙ ПОРОГ КУМУЛЯТИВНОЙ ЭНТРОПИИ: НОВЫЙ ПОДХОД К ОБНАРУЖЕНИЮ DDOS-АТАК В УСТРОЙСТВАХ ИНТЕРНЕТА ВЕЩЕЙ И СИСТЕМАХ УМНЫХ ДОМОВ

Амит Кумар Джайсвал

Цель исследования: *предложить инновационную модель с использованием адаптивного порога, которая применяет кумулятивный энтропийный анализ временных рядов данных для более эффективного обнаружения и смягчения атак флудинга в среде «умного дома.*

Метод: *системный анализ, математические модели. Результат: с ростом популярности систем «умного дома» в повседневной жизни, атаки типа кибер-флудинга на эти взаимосвязанные устройства стали критически важными. В настоящем исследовании предложена инновационная модель с использованием адаптивного порога, которая применяет кумулятивный энтропийный анализ временных рядов данных для более эффективного обнаружения и смягчения атак флудинга в среде «умного дома». Модель устанавливает динамические пороги, адаптируемые к изменениям колебаний данных в режиме реального времени, используя кумулятивную энтропию – показатель, который определяет непредсказуемость и дисперсию моделей сетевого трафика. Изучены передовые методы машинного обучения для уточнения процесса установления пороговых значений, что в итоге приведет к более высокой точности обнаружения аномалий. Фактически будут проанализированы такие важные факторы, как временные паттерны, типы протоколов и действия пользователей, с точки зрения их влияния на показатели целей.*

Научная новизна: *подтверждена эффективность предлагаемых адаптивных пороговых рамок в ответ на значительное сокращение ложных срабатываний при одновременном улучшении реагирования на возникающие угрозы, что в целом повышает устойчивость систем умного дома к обнаруженным атакам типа «флуд».*

Ключевые слова: *анализ временных рядов, смягчение последствий атак типа «флуд», безопасность умного дома, обнаружение аномалий, анализ сетевого трафика, временные паттерны данных.*

References/Литература

1. *Lee S-H, Shiue Y-L, Cheng C-H, Li Y-H, Huang Y-F. Detection and Prevention of DDoS Attacks on the IoT. Applied Sciences. 2022; 12 (23): 12407. https://doi.org/10.3390/app122312407.*
2. *Shrahili, M.; Kayid, M. Cumulative Entropy of Past Lifetime for Coherent Systems at the System Level. Axioms 2023, 12, 899. https://doi.org/ 10.3390/axioms12090899*
3. *M. Tharun Kumar, G. Sesha Phaneendra babu, D. Lakshmi Narayana Reddy, «A Novel Framework for Mitigating DDoS Attacks in IoT Based Smart Network Environments using Machine Learning», Industrial Engineering Journal, ISSN: 0970-2555 Volume: 53, Issue 5, May: 2024. http://www.journal-iiie-india.com/1_may_24/125_online_may.pdf.*
4. *A. K. Jaiswal, «Deep Comparison Analysis: Statistical Methods and Deep Learning For Network Anomaly Detection», 2024. https://doi.org/10. 5281/zenodo.14051107*
5. *J. Dragos, J. P. Ziegler, A. de Villiers, A.-L. Jousselme, and E. Blasch, «Entropy-Based Metrics For URREF Criteria to Assess Uncertainty in Bayesian Networks For Cyber Threat Detection», in 2019 22nd International Conference on InFormation Fusion (FUSION), Ottawa, ON, Canada, 2019, pp. 1–8. DOI: 10.23919/FUSION43075.2019.9011276.*
6. *V. Timcenko and S. Gajin, «Machine Learning Enhanced Entropy- Based Network Anomaly Detection», Advances in Electrical and Computer Engineering, vol. 21, no. 4, pp. 51–60, 2021. DOI: 10.4316/AECE.2021.04006*
7. *P. Verma, S. Tapaswi, and W. W. Godfrey, «An Adaptive Threshold-Based Attribute Selection to Classify Requests Under DDoS Attack in Cloud- Based Systems», Arab Journal of Science and Engineering, vol. 45, pp. 2813–2834, 2020. DOI: 10.1007/s13369-019-04178-x.*
8. *P. Sahoo and Gurdial Arora, «A Thresholding Method Based on Two-Dimensional Renyi's Entropy», Pattern Recognition, vol. 37, no. 6, pp. 1149–1161, 2004. DOI: 10.1016/j.patcog.2003.10.008.*
9. *H. Lin and N.Bergmann, «IoT Privacy and Security Challenges For Smart Home Environments,"InFormation, vol. 7, no. 44, 2016. DOI: 10.3390/ info7030044.*
10. *M.C. Dani et al., «Adaptive Threshold For Anomaly Detection Using Time Series Segmentation», in Neural InFormation Processing, S. Arik et al., Eds., vol 9491 of Lecture Notes in Computer Science., Springer Cham., 2015.*
11. *Amit Jaiswal., «DOS Attack Network Traffic Monitoring in Software Defined Networking Using Mininet and RYU Controller». 2022. DOI: 10.21203/ rs.3.rs-2282189/v1.*
12. *Berezin'ski P, Jasiul B, Szpyrka M. An Entropy-Based Network Anomaly Detection Method. Entropy. 2015; 17(4): 2367–2408. DOI: https://doi.org/ 10.3390/e17042367.*
13. *Rong Lan and Lekang Zhang. 2023. Image Thresholding Segmentation Algorithm Based on Two-parameter Cumulative Residual Masi Entropy. In Proceedings of the 2022 5th International Conference on Artificial Intelligence and Pattern Recognition (AIPR '22). Association For Computing Machinery, New York, NY, USA, pp.406–411. DOI: https://doi.org/10.1145/3573942.3574041.*

14. J.Assfalg et al., «Time Series Analysis Using the Concept of Adaptable Threshold Similarity», in 18th International Conference on Scientific and Statistical Database Management (SSDBM'06), Vienna, Austria, pp. 251–260, 2006. https://www.dbs.ifi.lmu.de/Publikationen/Papers/ssdbm06. threshold.pdf.
15. D. Shang and P. Shang , «Analysis of Time Series in the Cumulative Residual Entropy Plane Based on Oscillation Roughness Exponent, Nonlinear Dynamics, vol.100, pp.,2167–2186 , 2020. DOI:10.1007/s11071-020-05646-y.
16. A. Patharkar et al., «Eigen-entropy Based Time Series Signatures to Support Multivariate Time Series Classification», Scientific Reports, vol.14, no.1, Article16076, 2024. DOI:10.1038/s41598-024-66953-7.
17. Huraj, Ladislav; Lietava, Jakub; Šimon, Marek (2024), «UCM_FibIoT2024», Mendeley Data, V1. DOI: 10.17632/p42xjtv8pv.1.
18. Yu, H., Yang, W., Cui, B. et al. Renyi entropy-driven network traffic anomaly detection with dynamic threshold. Cybersecurity 7, 64 (2024). https://doi.org/10.1186/s42400-024-00249-1.
19. M. Thakur and R.K. Sharma, «Anomaly Detection in Smart Home Networks Using Adaptive Thresholding Techniques Based on Cumulative Entropy», International Journal of Computer Applications, 2022. https: //doi.org/10.5120/ijca2016911955.
20. D. G. Narayan, W. Heena, and K. Amit, «A Collaborative Approach to Detecting DDoS Attacks in SDN Using Entropy and Deep Learning», Journal of Telecommunications and InFormation Technology, vol. 3, no. 3, 2024. https://doi.org/10.26636/jtit.2024.3.1609.