Available online at https://journal.rescollacomm.com/index.php/ijqrm/index

International Journal of Quantitative Research and Modeling

e-ISSN 2721-477X
p-ISSN 2722-5046

Vol. 5, No. 3, pp. 341-353, 2024

Securing Network Log Data Using Advance Encryption Standard Algorithm and Twofish with Common Event Format

Moch. Dzikri Azhari Ali[1*], Asep Id Hadiana[2], Melina[3]

[1,2,3]*Department of Informatics, University of Jenderal Achmad Yani, Cimahi, Indonesia*

mochdzikriazhari20@if.unjani.ac.id

Abstract

The rapid advancement of information technology demands enhanced security for data exchange in the digital world. Network security threats can arise from various sources, necessitating techniques to protect information transmitted between interconnected networks. Securing network logs is a critical step in strengthening overall network security. Network logs are records of activities within a computer network, including unauthorized access attempts, user activities, and other key events. This research focuses on developing a network log security system by comparing the performance of the Advanced Encryption Standard (AES) and Twofish algorithms, integrated with the Common Event Format (CEF) for encrypting network logs. Tests were conducted on network log datasets to evaluate system functionality and performance. Results indicate that the AES algorithm performs encryption and decryption faster than Twofish. Across five tests with different file sizes, AES took an average of 2.1386 seconds for encryption, while Twofish required 22.8372 seconds. For decryption, AES averaged 2.451 seconds compared to Twofish's 26.140 seconds. The file sizes after encryption were similar for both algorithms. Regarding CPU usage, AES demonstrated higher efficiency. The average CPU usage during AES encryption was 0.5558%, whereas Twofish used 23.2904%. For decryption, AES consumed 0.4682% of CPU resources, while Twofish required 13.7598%. These findings confirm that AES is not only faster in both encryption and decryption but also more efficient in terms of CPU usage. This research provides valuable insights for optimizing network log security by integrating standardized log formats, like CEF, with appropriate encryption techniques, helping to safeguard against cyber threats.

Keywords: AES, CEF, cryptography, network log, Twofish

1. Introduction

The rapid development of information technology makes information security, especially on internet networks, very important. The imbalance between technological advances and security systems that do not evolve over time can lead to system vulnerabilities (Akhriana & Irmayana, 2019). Weak internet security increases the risk of data theft, known as cybercrime (Nyoman Putri Purnama Santhi & Nengah Nuarta, 2023). In 2020, malware attacks increased by 358% and ransomware increased by 435% compared to the previous year (Budi et al., 2021). In Indonesia, since 2019, the Ministry of Communication and Information recorded 29 cases of data leakage in institutions and companies, mostly caused by weak security (Samad & Pratama Dahlian Persadha, 2022). Therefore, a security system that can detect attacks becomes indispensable (Alamsyah et al., 2020).

Cryptography is one method to maintain data security by applying information encoding techniques, and encryption is one of the methods used in cryptography (Putra et al., 2023) (Melina et al., 2022). The encryption process converts plaintext into ciphertext, making the data unreadable or secret. While decryption is the reverse process that returns the ciphertext to plaintext, so that the encrypted data can be restored to its original form (Nurnaningsih & Permana, 2018). There are various cryptographic techniques used to encode messages or information data, including substitution and permutation techniques. One cryptographic method that utilizes both is the Advanced Encryption Standard (AES) and Twofish algorithms. The AES algorithm uses 128-bit encryption blocks with keys of 128, 192, or 256-bit size and works through several rounds of transformations involving substitution, permutation, mixing, and key addition. Meanwhile, the Twofish algorithm is an algorithm that also utilizes Feistel networks and works on 128-bit blocks with keys up to

256-bit in size. Twofish is known for its flexibility and simple yet effective design, making it a strong competitor to AES in terms of performance and security (Rizvi et al., 2011).

Applying encryption to network logs can provide a high level of protection. The encryption process makes network logs difficult to read or manipulate by unauthorized parties, so a hacker will find it difficult to read or make significant changes to network logs without having the appropriate encryption key (Lantz, 2006). Network logs are records or recordings of activities that occur within a computer network that can detect and respond to attacks, and monitor network health so there needs to be a sufficient strategy to protect network log data (Ramli et al., 2023). Network log security is a very important step to strengthen the security of the network (Lantz, 2006). However, the diversity of log formats used by various applications can make it difficult for system administrators, intrusion detection system (IDS) developers, and security analysts to manage and interpret log data. To solve this problem, there are several log format standards, one of which is the Common Event Format (CEF) which is better than other formats because it has keys such as 'act' (action in the event), 'app' (application protocol), 'request', 'request Method', and others. The purpose of this standardization is to simplify the analysis process and improve interoperability between systems. However, the adoption of this common log format is still limited, especially in the context of intrusion detection systems (Sapegin et al., 2014). Moreover, in network security, CEF is used as an industry-recognized logging format, which allows data to be interpreted easily and can be implemented by security devices (More et al., 2020). CEF can make the analysis process more efficient, enable easier log integration, and speed up the identification of security threats when compared to manual methods (Buczak et al., 2017).

This research applies the AES and Twofish algorithms to encrypt network logs formatted using the Common Event Format (CEF) as a structured log standard. The AES and Twofish algorithms were chosen because they are both known for their high level of security, as well as good efficiency in handling large amounts of data. CEF was used in this research because it provides a consistent structure and facilitates integration with various other security systems. This research aims to test whether the use of these two algorithms can improve the security of network logs and how they perform in terms of encryption speed, file size after encryption, and CPU resource usage. The results of this study are expected to provide insight into the effectiveness of network log encryption in maintaining the confidentiality and integrity of sensitive information on computer networks that use the CEF log format.

2. Literature Review

Several previous studies have examined the application of encryption to enhance the security of network logs. Research conducted by (Lantz, 2006) demonstrates that applying encryption to network logs can significantly elevate the level of protection. The encryption process renders network logs difficult to read or manipulate by unauthorized parties, making it challenging for hackers to access or alter log data without the appropriate encryption key.

Further, (Dzikri Azhari et al., 2024), explored the use of the AES encryption algorithm with a 128-bit key on network log files formatted in Common Event Format (CEF). Their findings indicate that this approach effectively ensures the confidentiality and integrity of network log data. The study utilized a robust encryption key that combined uppercase letters, lowercase letters, numbers, and special characters to strengthen the security of both encryption and decryption processes. The incorporation of CEF format is recognized as beneficial for facilitating more effective security analysis of recorded network activities.

Moreover, research by (Praptodiyono et al., 2021) investigates the performance of both the AES and Twofish algorithms in securing data on networks. Their results highlight that while AES exhibits advantages in terms of computation time and memory efficiency, Twofish offers benefits such as smaller encrypted file sizes and reduced CPU usage. The study concludes that despite AES's superiority in speed and efficiency, Twofish provides a more compact data size along with enhanced security features.

Research (Buczak et al., 2017), further examined the advantages of using CEF in network log security. Their study illustrates that CEF allows for easier integration of logs and expedites the identification of security threats compared to traditional manual methods. The results indicated that a cyber analyst utilizing CEF could focus on less than 0.4% of all IP addresses to identify all malicious IPs, thereby increasing the efficiency of the log analysis process.

Overall, these studies collectively underscore the importance of encryption and the utilization of standardized formats like CEF in bolstering the security and integrity of network logs, thereby providing a robust framework for ongoing research in this field.

3. Materials and Methods

There are several stages in this research. The first stage is the collection of network log data. The second stage involves the application of the Common Event Format (CEF) to the collected log data. The third stage is the encryption

process using the AES algorithm. The fourth stage is the decryption of the data that has been encrypted with AES. After that, the fifth stage is the encryption process using the Twofish algorithm. The sixth stage is the decryption of the data encrypted with Twofish. The final stage involves the evaluation and testing of the results from both encryption and decryption processes, as can be seen in Figure 1.

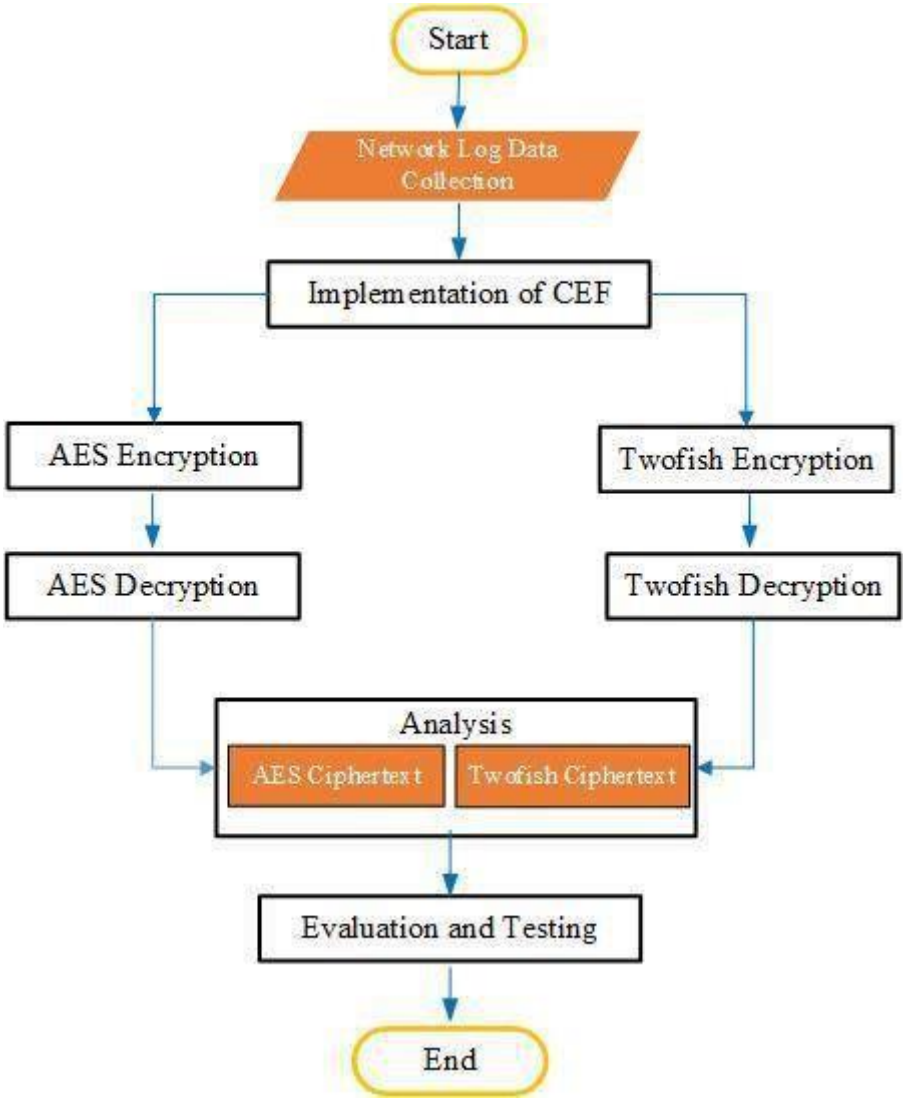


Figure 1: Research Method Flow

3.1. Materials

The data used in this study consists of network log entries collected in the Common Event Format (CEF). The logs are sourced from a third-party application, Wireshark, resulting in a dataset comprising network traffic data stored in PCAP files. The initial dataset includes various attributes such as source IP addresses, destination IP addresses, source ports, destination ports, timestamps, and descriptive messages.

3.2. Methods

3.2.1. Network Log Data Collection

The first step undertaken in this research is the collection of network log data. In this stage, gathering sufficient and relevant network log data is crucial as it serves as the input for the subsequent application of the Common Event Format (CEF) and data encryption processes. The network log data used in this research is acquired through third-party software, specifically Wireshark, which captures the data in PCAP file format. This raw log data is essential for analyzing network activities and will undergo further formatting and encryption to enhance its security and facilitate efficient analysis.

3.2.2. Application of CEF

The next step is the application of the Common Event Format (CEF), which plays a key role in standardizing and structuring the network log data. By using CEF, network log formats become more accessible and manageable for network analysts. This format not only improves the efficiency of reading and interpreting log data but also provides a structured framework to solve network issues. At this stage, the logs generated from PCAP files are converted into the CEF format.

The purpose of implementing CEF is to standardize the network log format, making it easier to analyze. In practice, each generated log will follow the CEF structure, which consists of a header and an extension. The header contains basic information such as the CEF version, source device, and priority, while the extension includes specific details about the log event. In this research, the network log data to be converted into the CEF format is extracted from PCAP file types.

The first step in applying CEF is to identify the log source to be integrated into the CEF format. Once the log source is identified, the next step is to extract key information from the logs, including the source IP address (src), destination IP address (dst), source port (spt), destination port (dpt), start time (start), end time (end), and a descriptive message (msg). The next step is to create the CEF header, beginning with selecting the appropriate CEF version. Vendor, product, product version, event signature ID, brief event description, and severity level are then added to the header. An example of a CEF header is as follows: CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|.

Once the header is created, the next step is to organize the extension part of the CEF log. The extension contains key-value pairs that provide additional details about the event. The previously extracted information is added to the extension in the appropriate format. An example of an extension is as follows: src=142.250.115.157 dst=10.8.3.101 spt=443 dpt=49725 start=2023-08-03T22:46:25Z end=2023-08-03T22:46:25Z msg=TCP Traffic from 142.250.115.157 to 10.8.3.101 on ports 443 →49725.

The final step is to combine the header and extension to form a complete log entry in CEF format. An example of a complete log entry combining the header and extension would look like this: CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.101 spt=443 dpt=49725 start=2023-08-03T22:46:25Z end=2023-08-03T22:46:25Z msg=TCP Traffic from 142.250.115.157 to 10.8.3.101 on ports 443→49725. The output of this process is stored as a text file containing network log data in CEF format.

Through this structured CEF implementation process, network log analysis becomes more effective because the data produced is more consistent and organized. This allows network analysts to quickly identify and respond to security threats, as well as resolve potential issues within the network. The implementation of CEF not only enhances operational efficiency but also improves the overall security level of the network by providing a robust framework for log management and security analysis.

3.2.3. AES Encryption

The next step is the encryption process using the AES algorithm. After the network log data has been formatted into CEF, this step involves applying the AES algorithm to encrypt the network log data. This process converts the original log data, now in CEF format (plaintext), into an encrypted form (ciphertext).

The AES encryption process begins with the generation of a 128-bit (16-byte) symmetric encryption key, which will be used to both encrypt and decrypt the network log data. The network log data to be encrypted is the previously formatted CEF data stored in a text file. The log data, formatted in the Common Event Format (CEF), is divided into 128-bit data blocks. If the data size is not a multiple of 128 bits, padding is added to fill the gap.

In the initial phase, each data block is combined with the encryption key using an XOR operation, in a step known as AddRoundKey. The data blocks then go through nine main rounds, each consisting of four stages. The first stage is SubBytes, where each byte in the data block is replaced with another byte using a substitution table (S-box) to provide strong non-linearity in the encryption. Next is ShiftRows, in which the rows of the data block are cyclically rotated to increase diffusion within the block. Then comes MixColumns, where the columns of the data block are mixed using a linear mathematical transformation. Finally, AddRoundKey is applied again, where a specific round key is combined with the data block using an XOR operation.

In the final round, the same three stages (SubBytes, ShiftRows, and AddRoundKey) are performed, but the MixColumns stage is omitted to simplify the decryption process.

Once all the rounds are completed, the encrypted data blocks (ciphertext) are recompiled into a single encrypted file. The encryption process will include metrics such as the time required, the file size after encryption, and CPU usage during the encryption process. This information will be displayed in the encryption results message, along with a download button to retrieve the encrypted AES file and the key used.

Through this encryption process, sensitive network log data is well protected, preserving its integrity and confidentiality from unauthorized access. The security of AES-128, stemming from its complex transformations and the use of a strong encryption key, ensures that the data is not easily compromised by unauthorized parties.

3.2.4. AES Decryption

After the network log data has been encrypted using the AES algorithm, the next step is the decryption process to restore the encrypted data back to its original form (plaintext). The AES decryption process follows the same method as encryption but with the steps in reverse order. Decryption is performed by uploading the previously encrypted file along with the key used during the prior encryption process.

The decryption process begins by taking the encrypted network log file (ciphertext) and splitting it back into 128-bit data blocks. Each data block then undergoes decryption rounds that consist of steps opposite to those of encryption. In the first round, the data block goes through three stages: AddRoundKey, where the data block is combined with the last round key using an XOR operation. Next is InvShiftRows, where the rows in the data block are rotated back to their original positions. Then, InvSubBytes is applied, where each byte in the data block is replaced with the original byte using the inverse substitution table (inverse S-box).

Following this, the data block goes through nine main decryption rounds, which include four stages: AddRoundKey, where the data block is combined with the corresponding round key; InvMixColumns, where the columns of the data block are mixed using the inverse linear mathematical transformation; InvShiftRows, where the rows in the data block are rotated back to their original positions; and finally, InvSubBytes, where each byte in the data block is replaced with the original byte using the inverse S-box. In the final round, only three stages are performed (AddRoundKey, InvShiftRows, and InvSubBytes), with the InvMixColumns stage omitted.

Once all the rounds are completed, the decrypted data blocks (plaintext) are recompiled into a single complete network log file. If any padding was added during the encryption process, it will be removed to restore the data to its original form. The decryption process will include metrics such as the time taken, the file size after decryption, and CPU usage during the decryption process. This information will be displayed in the decryption results message, along with a download button to retrieve the decrypted AES file.

This decryption process ensures that the encrypted network logs can be accessed again in their original form, maintaining the confidentiality and integrity of the information. By following the proper decryption steps, the system ensures that only authorized parties can access the network log data, thereby enhancing the overall security of the network log system.

3.2.5. Twofish Encryption

The next stage involves the data encryption process using the Twofish algorithm. In addition to AES encryption, the network log data that has been formatted in CEF will also undergo encryption with Twofish. The encryption process with Twofish begins with the generation of a 128-bit symmetric encryption key, which will be used for encrypting and decrypting the network log data. The network log data to be encrypted consists of files formatted in CEF with a txt file type. The CEF-formatted network log data is then divided into 128-bit data blocks. If the data size is not a multiple of 128 bits, padding will be added to fill the gaps.

In the first step, each data block is XORed with the encryption key in a stage known as initial Key Whitening. Each data block then undergoes 16 encryption rounds, consisting of several main stages. Each round begins by splitting the block into two parts: left and right. The left part is operated on by the function g, which includes byte substitution through the S-box, linear mixing based on the MDS matrix, and the Pseudo-Hadamard Transform (PHT). The output of this function g is then XORed with the two right parts and shifted for further diffusion.

After all rounds are completed, the processed data block is XORed again with the encryption key in the final Key Whitening stage. The final output of the encryption process is the encrypted network log data file. The encrypted data blocks are then recompiled into a single complete encrypted file. The encryption process will include metrics such as the time taken, the file size after encryption, and CPU usage during the encryption process. This information will be displayed in the encryption results message, along with a download button to retrieve the encryption key file and the Twofish encrypted file.

Through this encryption process, sensitive network log data will be well-protected, maintaining the integrity and confidentiality against unauthorized access. The security of the Twofish algorithm, derived from its 16-round Feistel structure and the use of a strong key, ensures that the data is not easily compromised by unauthorized parties.

3.2.6. Twofish Decryption

After the network log data has been encrypted using the Twofish algorithm, the next step is the decryption process. The encrypted log data will be restored to its original form using the Twofish algorithm. The Twofish decryption process follows the same method as encryption, but with the steps in reverse order. Decryption is performed using the same key that was used in the encryption process, ensuring that only parties with the correct key can access the original data.

The decryption process begins by taking the previously encrypted network log file (ciphertext) and dividing it back into 128-bit data blocks. In the first step of decryption, each block is XORed with the encryption key in the initial Key Whitening stage. Each data block then undergoes 16 decryption rounds, which are the inverse of the encryption rounds. Each round starts by splitting the block into two parts: left and right. The left part is operated on by the same function g as in encryption, and the result is XORed with the right parts and shifted back to its original position.

After completing the 16 rounds, the processed data block is XORed again with the encryption key in the final Key Whitening stage. The final output of this decryption process is the network log file restored to its original form. The decrypted data blocks are then recompiled into a single complete network log file. If any padding was added during the encryption process, it will be removed to restore the data to its original state.

The decryption process will include metrics such as the time taken, the file size after decryption, and CPU usage during the decryption process. This information will be displayed in the decryption results message, along with a download button to retrieve the Twofish decrypted file. This decryption process ensures that the encrypted network logs can be accessed again in their original form, maintaining the confidentiality and integrity of the information. If the decryption steps are correctly followed, the system ensures that only authorized parties can access the network log data, enhancing the overall security of the network log system.

3.2.7. Evaluation and Testing

The last step in this research is result evaluation and testing. Testing is done by measuring the performance of each algorithm used, namely AES and Twofish, namely by comparing the file size before and after the encryption and decryption process, measuring the speed of the time required for the encryption and decryption process of the two methods and also by comparing cpu usage when the encryption and decryption process is carried out in both methods. The results of this evaluation will show which algorithm is more effective based on file size efficiency and processing speed.

4. Results and Discussion

This research produced a web application that displays the application of the CEF format, as well as an easy-to-follow encryption and decryption process. The application includes features such as PCAP file upload, conversion to CEF format, file encryption using AES, file decryption using AES, file encryption using Twofish, file decryption using Twofish, and performance testing.

4.1. Network Data Log Collection

Network log data collection begins with the selection of the network interface to be monitored and the start of the data capture process. Every data packet that passes through the interface is logged, including important information such as source and destination IP addresses, source and destination ports, and the type of protocol used. After the data collection period ends, the capture process is stopped and the results are saved in PCAP (Packet Capture) format, which can be seen in Figure 2.

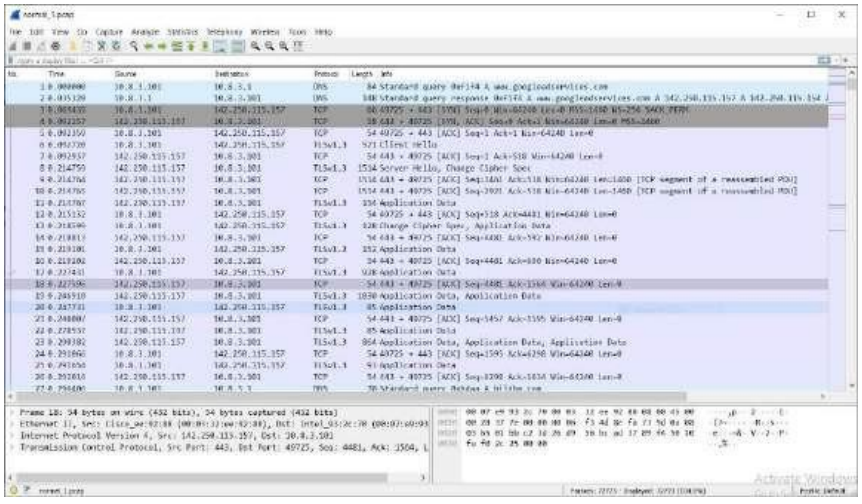


Figure 2: Network Log Collection

4.2. Application of CEF Format

The process of implementing the CEF format begins with uploading a PCAP file that contains the network log. The log file is then converted to the CEF format, which structures the network log data to be more structured and easy to understand. Once the conversion is complete, the results will be displayed and can be downloaded as a .txt file, as seen in Figure 3.

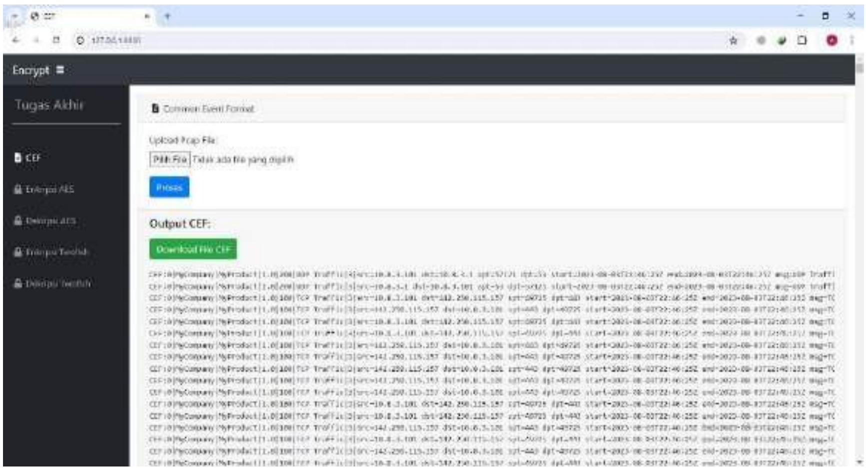


Figure 3: CEF Deployment Process

Once the network logs are converted to CEF format, the data will be organized following a uniform standard, making integration with other security analysis systems easier. An example of data that has been converted to CEF format can be seen in Figure 4.

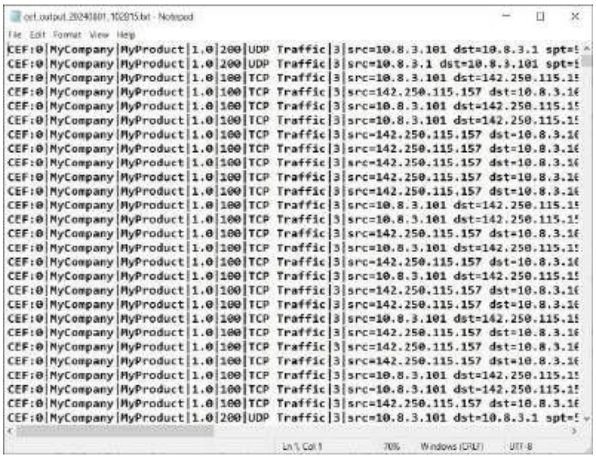


Figure 4: CEF Format Implementation Result

4.3. AES Encryption

The encryption process begins by uploading a .txt file containing network logs that have been formatted in CEF. The data in the file is then converted into ciphertext, which is a random message that cannot be read by parties without the encryption key, thus ensuring the protection of sensitive information. Once the encryption is complete, the result will display information such as the file size after encryption, the time taken for the encryption process, as well as the CPU usage during encryption using the AES algorithm. The encrypted file and the key used can also be downloaded, as shown in Figure 5.

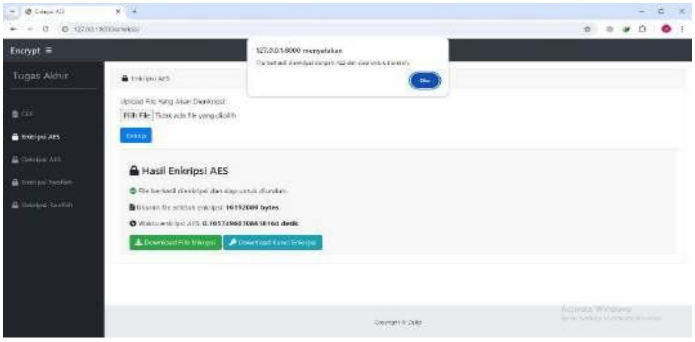


Figure 5: AES Encryption Process

Once the encryption process is complete, the data in the text file is converted into a random message that cannot be

accessed without the encryption key. An example of the result of this encryption can be seen in Figure 6.

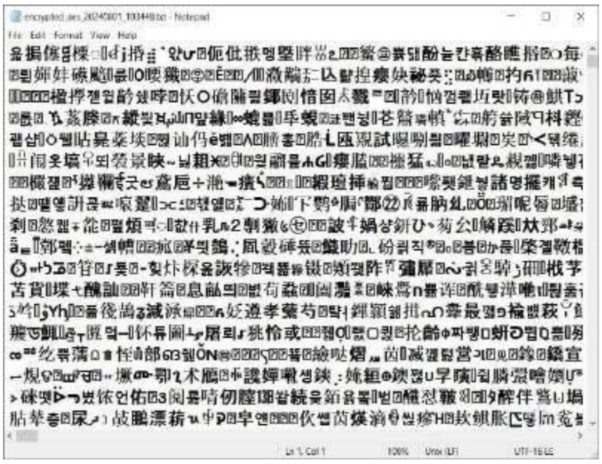


Figure 6: AES Encryption Result

4.4. AES Decryption

The decryption process will restore the file that has been encrypted with the AES algorithm to its original form so that it can be accessed and read again. This step is done by uploading the encrypted file as well as the key obtained during the previous encryption process. After that, the file will be decrypted and restored to its original form. After the decryption is complete, information such as the file size after decryption, the time taken, and CPU usage during the decryption process with AES will be displayed. The decrypted file in .txt format can also be downloaded, as shown in Figure 7.

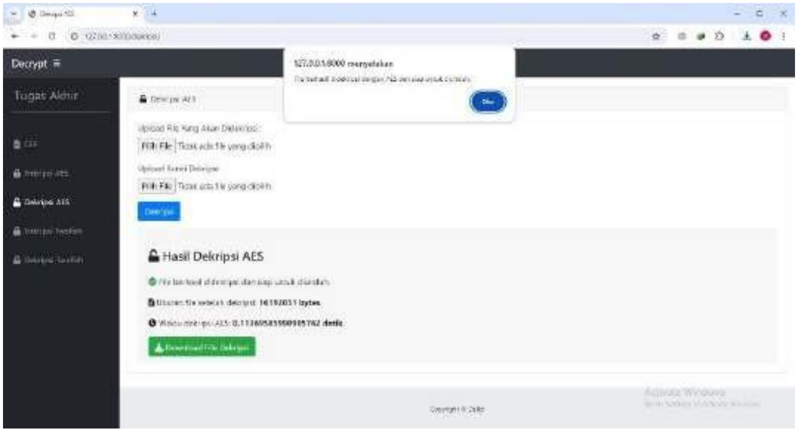
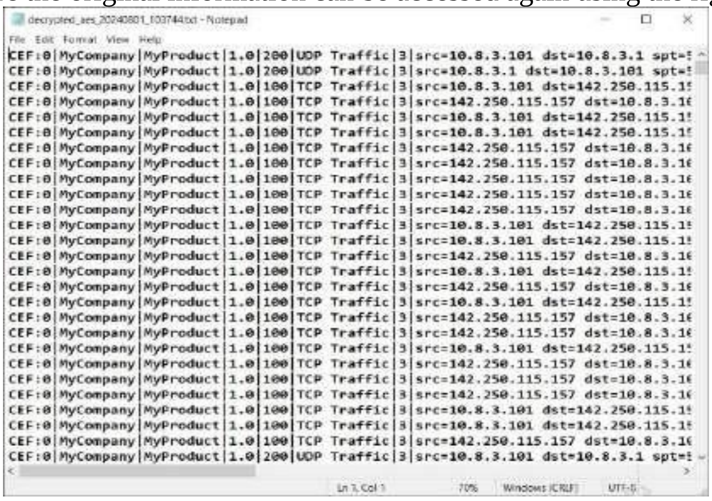


Figure 7: AES Decryption Process

After the decryption process is complete, it will ensure that the initially unreadable data is restored to its original form, as shown in Figure 8, where the original information can be accessed again using the right key.



```
CEF:0|MyCompany|MyProduct|1.0|200|UDP Traffic|3|src=10.8.3.101 dst=10.8.3.1 spt=
CEF:0|MyCompany|MyProduct|1.0|200|UDP Traffic|3|src=10.8.3.1 dst=10.8.3.101 spt=
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=10.8.3.101 dst=142.250.115.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=10.8.3.101 dst=142.250.115.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=10.8.3.101 dst=142.250.115.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=10.8.3.101 dst=142.250.115.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=10.8.3.101 dst=142.250.115.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=10.8.3.101 dst=142.250.115.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=10.8.3.101 dst=142.250.115.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=10.8.3.101 dst=142.250.115.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=10.8.3.101 dst=142.250.115.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=10.8.3.101 dst=142.250.115.1
CEF:0|MyCompany|MyProduct|1.0|100|TCP Traffic|3|src=142.250.115.157 dst=10.8.3.1
CEF:0|MyCompany|MyProduct|1.0|200|UDP Traffic|3|src=10.8.3.101 dst=10.8.3.1 spt=
```

Figure 8: AES Decryption Result

4.5. Twofish Encryption

The encryption process with the Twofish algorithm is used to secure network logs that have been formatted in CEF to improve information security. Encryption begins by uploading a .txt file of network logs that have been formatted in CEF. The data in the file is then converted into ciphertext, which is a random message that cannot be read without an encryption key, thus ensuring the protection of sensitive information. Once the encryption process is complete, the result will display information such as the file size after encryption, the duration of encryption, as well as the CPU usage during the encryption process with Twofish. In addition, the encrypted file and the key used for encryption can be downloaded, as shown in Figure 9.

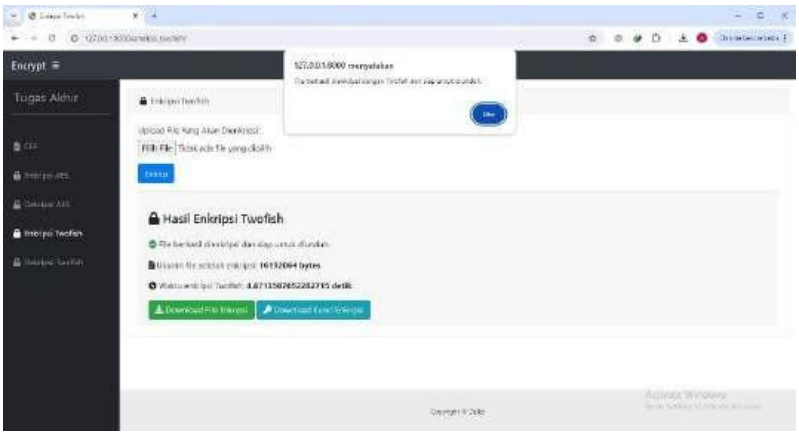


Figure 9: Twofish Encryption Process

Once the encryption with Twofish is complete, the text data in the file will turn into a random message that cannot be accessed without the encryption key. An example of the result of this encryption can be seen in Figure 10.

Figure 10: Twofish Encryption Result

4.6. Twofish Decryption

The decryption process with the Twofish algorithm serves to restore files that have previously been encrypted, so that they can be accessed or read back in plaintext form. This step is done by uploading the encrypted file and the key obtained during the previous encryption process. After that, the successfully decrypted file will return to its original format. When the decryption is complete, information such as the file size after decryption, process duration, and CPU usage during decryption with Twofish will be displayed. The decrypted file in .txt format can also be downloaded, as shown in Figure 11.

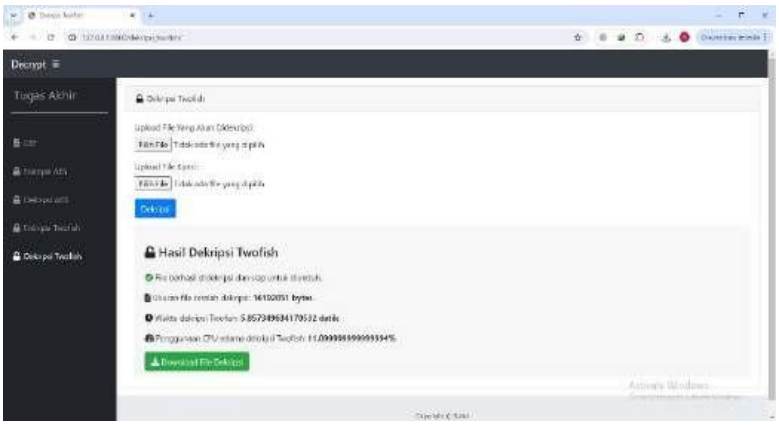


Figure 11: Twofish Decryption Process

Once the decryption process with Twofish is complete, the initially unreadable data is restored to its original form and can be accessed using the correct key. An example of this decryption result can be seen in Figure 12.

Figure 12: Twofish Decryption Result

4.7. Performance Testing

Performance testing was conducted to measure the efficiency and speed of the AES and Twofish encryption and decryption algorithms. In addition, CPU usage measurements were also taken to assess how much processor resources were used during the encryption and decryption process. This test involved 5 files of varying sizes to see how each algorithm handled the encryption and decryption process.

4.7.1. Testing the Encryption Process

Testing is done by measuring the time required to encrypt files of various sizes using the AES and Twofish algorithms. In addition, the file size before and after encryption as well as the encryption process time was recorded. To get a more complete picture of the efficiency of the algorithms, CPU usage was also measured during the encryption process, the test results can be seen in Table 1.

Table 1: Encryption Process Testing

File Size	File Size After Encryption		Processing Time Encryption		CPU Usage	
	AES (bytes)	Twofish (bytes)	AES (sec)	Twofish (sec)	AES (%)	Twofish (%)
15.4 MB	16.192080	16.192064	0.091	5.223	0.093	5.578
19.2 MB	20.220400	20.220384	0.131	6.290	0.140	6.703
58.2 MB	61.103696	61.103680	1.200	20.588	0.5	21.609
86.5 MB	90.755264	90.755248	2.755	32.051	0.75	33.484
148 MB	156.167712	156.167696	6.516	50.034	1.296	51.078
	Average		2.1386	22.8372	0.5558	23.6904

From the test results, it can be seen that the AES algorithm is consistently faster in performing the encryption process compared to the Twofish algorithm. Although the file size after encryption is relatively the same for both algorithms, the time taken by Twofish is much longer than AES, especially as the file size increases. This shows that AES is more efficient in terms of encryption processing time. In addition, the CPU usage by AES is also lower compared to Twofish, confirming that AES is not only faster but also more efficient in the use of CPU resources during the encryption process.

4.7.2. Testing the Decryption Process

Testing is done by measuring the time taken to decrypt files of various sizes using the AES and Twofish algorithms. In addition, the file size before and after decryption as well as the decryption process time are recorded. To get a more complete picture of the efficiency of the algorithms, CPU usage was also measured during the decryption process, the test results can be seen in Table 2.

Table 2: Decryption Process Testing

File Size	File Size After Decryption		Decryption Processing Time		CPU Usage	
	AES (bytes)	Twofish (bytes)	AES (sec)	Twofish (sec)	AES (%)	Twofish (%)
16.2 MB	16.192051	16.192051	0.083	4.864	0.093	28.9
20.2 MB	20.220369	20.220369	0.126	6.052	0.125	0.699
61.1 MB	61.103664	61.103664	1.629	25.677	0.421	20.7
90.7 MB	90.755239	90.755239	4.092	35.584	0.687	5.5
156.1 MB	156.167680	156.167680	6.323	58.522	1.015	13.0
	Average		2.451	26.140	0.468	13.760

From the decryption test results, it can be seen that the AES algorithm is also faster than the Twofish algorithm in performing the decryption process. Twofish's decryption time is much longer than AES, especially for large files. This shows that AES is not only faster in the encryption process but also more efficient in the decryption process. In addition, the CPU utilization by AES is also lower than that of Twofish, which confirms that AES is more efficient in utilizing CPU resources during the decryption process.

5. Conclussion

Based on the results of the research conducted, it can be concluded that the encryption and decryption system developed in this research successfully implements the Advanced Encryption Standard (AES) and Twofish algorithms to secure network log data with Common Event Format (CEF). The system is designed so that network log data stored or sent to any platform remains secure by changing the contents of the data. Even if the data is successfully retrieved by unauthorized parties, they will not be able to open it without the appropriate key. The performance test results show that the AES algorithm is faster in performing the encryption and decryption process than Twofish. The average time taken for encryption from testing 5 different files with varying sizes was 2.1386 seconds for AES, while for Twofish it was 22.8372 seconds. In the decryption process too, the average time taken by AES was 2.451 seconds, while Twofish required 26.140 seconds. The file size after encryption is relatively the same for both algorithms. Also, in terms of CPU usage, the AES algorithm shows higher efficiency. The average CPU usage for encryption with AES is 0.5558%, while for Twofish it is 23.2904%. For decryption, the average CPU usage with AES is 0.4682%, while with Twofish it is 13.7598%. This confirms that AES is not only faster in encryption and decryption but also more efficient in CPU resource usage.

Acknowledgments

The authors would like to express their deepest gratitude to Mr. Asep Id Hadiana and Ms. Melina, who have provided invaluable guidance and support throughout this research. Their insights and inputs were crucial in shaping the direction and quality of this research. The authors would also like to thank the Department of Informatics, Universitas Jenderal Achmad Yani, Cimahi, for providing the necessary facilities and resources to conduct this research. Finally, sincere appreciation is given to all parties who have contributed in any form to the completion of this project.

References

Akhriana, A., & Irmayana, A. (2019). Web App for Detecting Types of Computer Network Attacks by Utilizing Snort and Honeypot Logs. *CCIT*, *12*(1), 87–98.

Alamsyah, H., -, R., & Al Akbar, A. (2020). Network Security Analysis Using Network Intrusion Detection and Prevention System. *JOINTECS (Journal of Information Technology and Computer Science)*, *5*(1), 17. https://doi.org/10.31328/jointecs.v5i1.1240

Buczak, A. L., Berman, D. S., Yen, S. W., Watkins, L. A., Duong, L. T., & Chavis, J. S. (2017). Using sequential pattern mining for common event format (CEF) cyber data. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3064814.3064822

Budi, E., Wira, D., & Infantono, A. (2021). Strategy to Strengthen Cyber Security to Realize National Security in the Era of Society 5.0. *Prosiding Seminar Nasional Sains Teknologi Dan Inovasi Indonesia (SENASTINDO)*, *3*(November), 223–234. https://doi.org/10.54706/senastindo.v3.2021.141

Dzikri Azhari, M., Hadiana, A. I., & Melina, M. (2024). *DATA SECURITY TECHNIQUES USING ADVANCE ENCRYPTION STANDARD ALGORITHM WITH COMMON EVENT FORMAT TO IMPROVE NETWORK LOG SECURITY*. *7*.

Lantz, B. (2006). Locking Down Log Files: Enhancing Network Security By Protecting Log Files. *Issues In Information Systems*, *VII*(2), 43–47. https://doi.org/10.48009/2_iis_2006_43-47

Melina, M., Sukono, F., Napitupulu, H., & Kusumaningtyas, V. A. (2022). Electronic Signature Verification with Authentication Technique Based on Public Key Cryptography System Using Rivest-Shamir-Adleman Cryptographic Algorithm. *Jurnal Matematika Integratif*, *18*(1), 27. https://doi.org/10.24198/jmi.v18.n1.38343.27-39

More, S., Jamadar, I., & Kazi, F. (2020). Security Visualization and Active Querying for OT Network. *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020*. https://doi.org/10.1109/ICCCNT49239.2020.9225275

Nurnaningsih, D., & Permana, A. A. (2018). Design of Data Security Application with Advanced Encyption Standard Algorithm (Aes). *Jurnal Teknik Informatika*, *11*(2), 177–186. https://doi.org/10.15408/jti.v11i2.7811

Nyoman Putri Purnama Santhi, N., & Nengah Nuarta, I. (2023). Strengthening Police Law Enforcement in Order to Optimize Cybercrime Countermeasures in Indonesia. *SCIENTIA: Journal of Multi Disciplinary Sciences*, *02*(1), 15–27.

Praptodiyono, S., Muhammad, F., & Wiriyadinata, D. (2021). Analysis security system performance MIPv6 in signaling process using AES and Twofish algorithms. *Teknika: Jurnal Sains Dan Teknologi*, *17*(2), 158. https://doi.org/10.36055/tjst.v17i2.13069

Proceedings - 2011 International Conference on Communication Systems and Network Technologies, CSNT 2011, 76–79. https://doi.org/10.1109/CSNT.2011.160

Putra, W., Fahlevi, M. R., & Hidayat, A. T. (2023). Implementation of Advanced Encryption Standard Algorithm for Document Security, *Teknologi Dan Informasi*, *1*(2), 76–83. https://journal.grahamitra.id/index.php/jurikti/article/view/55%0Ahttps://journal.grahamitra.id/index.php/jurikti/article/download/55/181

Ramli, M., Soewito, B., Universitas, B., Nusantara, J., Raya, K., Jeruk, N., 27, K., Kebon Jeruk, K., & Jakarta, B. (2023). Network Security Monitoring and Evaluation with System Information and Security Management (SIEM) Approach. *Faktor Exacta*, *16*(1), 1979–276. https://doi.org/10.30998/faktorexacta.v16i1.16534

Rizvi, S. A. M., Hussain, S. Z., & Wadhwa, N. (2011). Performance analysis of AES and Twofish encryption schemes.

Samad, M. Y., & Pratama Dahlian Persadha. (2022). Understanding Cyber Warfare and the Role of the National Intelligence Agency in Countering Cyber Threats. *JURNAL IPTEKKOM Jurnal Ilmu Pengetahuan & Teknologi Informasi*, *24*(2), 135–

Sapegin, A., Jaeger, D., Azodi, A., Gawron, M., Cheng, F., & Meinel, C. (2014). Normalisation of Log Messages for Intrusion Detection. *Journal of Information Assurance and Security*, *9*(3), 167–176.