GPT Powered Log Analysis: Enhancing SOC Decision Making for Malicious and Benign Security Log Classification

ARTHUR HERMANN, University of Twente, The Netherlands

In the landscape of cybersecurity the ability to analyse and prevent new cyber attacks is essential for a business's survival. Most businesses rely on outsourcing cybersecurity to a Security Operations Center, which comprise of IT specialists responsible for analysing and classifying security logs. Traditional methods of security log analysis fall short due to the complexity and volume of security logs. This leads to analyst burnout causing disruptions to business processes. Leveraging Large Language Models' ability to interpret and generate text could automate analysis and classification of security logs reducing the likelihood that business processes are disrupted. This paper examines GPT, a specific Large Language Model's ability to analyse and evaluate windows security logs as benign or malicious according using three different prompt complexities. Results indicate that a GPT based approach with a medium or high complexity prompt consistently identify and classify benign logs but is ineffective when identifying malicious security logs.

Additional Key Words and Phrases: Log analysis, Large Language Models, Security Operations Center, Prompt Engineering.

1 INTRODUCTION

The number and sophistication of cyber attacks has increased exponentially since their inception, posing threats to business processes and organisations. It is estimated that 90% of businesses have reported a security incident resulting in 46% of those business losing sensitive data [6]. Furthermore, cyber attacks often target delicate processes such as supply chains, threaten resource availability via denial of service attacks (DDoS), and lead to disinformation [1]. A security log is a piece of text describing key information about performed user actions and contextual information such as the time when using specific software.

Security logs are essential to understanding how these attacks unfold as they provide information such as the time, date, and user behaviour of when the attack took place. Traditional cyber security methods for businesses are outsourced to Security Operations Centers (SOCs) that comprise of a team of IT specialists who analyse and classify security logs. The volume and complexity of security logs contribute to frequent analyst burnout, increasing the number of undetected malicious logs [2]. A disruption in SOC processes leads to delays in a business' execution requests, leading to a loss in revenue [7].

Advancements in Artificial Intelligence (AI) have the potential to improve existing cyber security measures. Large Language Models (LLM) are AI techniques focused on human language interpretation and text generation. LLMs suh as GPT 4 are shown to score in the 90th percentile of the Uniform Bar Exam and in the 99th percentile for the Graduate Record Examination (GRE) verbal section [10]. When using GPT to perform security log analysis, preliminary research by [4] highlights LLM's slight improvement in precision, recall, and F1 on classifying logs over existing security log analysis software such as SentiLog and DeepLog.

This paper aims to bridge the gap between existing SOC security log analysis methods and LLM's text interpretation abilities in order to improve cyber attack protection for business processes. The importance of this research lies on the profound impact cyber attacks have on business and how novel technologies such as LLMs have the potential of improving cyber attack detection. To explore the effect LLMs have on security log classification the following research question is used: What is the effectiveness of Large Language Models (LLMs) in distinguishing between malicious and benign security logs?

The majority of security log analysis software classifies security logs as low, medium or high priority while security logs between benign and malicious. Why does this difference occur? This is because the label dataset used for this paper includes malicious labels for each log. As a result, it allows an LLM to easily distinguish between benign and malicious logs. The research question can be divided into the following sub research questions:

- How accurately can LLMs classify security logs as malicious or benign?
- What are the potential implications of integrating large language models into SOC workflows for improving threat detection and incident response capabilities?
- What is the impact of prompt complexity on LLM comprehension and response generation in security log analysis?

How will the sub research questions be addressed? The first sub research question will be addressed by LLM's performance on system logs with measures such as accuracy, precision, recall, and F1 score. The second sub research question will be addressed by examining LLM's broader integration into current SOC security log analysis processes. The third research question will be addressed by incorporating three different prompt complexities.

The data used in this paper collected by Alsaheel et al. contains several types of windows security logs meant for training and testing security log classification models. Similarly, Python scripts for data processing will not be used as the machine learning models used in dataset use tokenized data in contrast to this paper's use of human language in security log analysis. To understand the uses and limitations of LLMs in log analysis it is essential to examine what LLMs are capable of along with the current landscape of SOCs.

2 STATE OF THE ART

The state of the art for SOCs involves the use of specific software and communication between analysts to to analyse and classify security logs. This section focuses on the state of the art of SOCs followed by LLMs abilities and limitations when classifying security

Author's address: Arthur Hermann, a.hermann-2@student.utwente.nl, University of Twente, P.O. Box 217, Enschede, The Netherlands, 7500AE.

TScIT 41, July 5, 2024, Enschede, The Netherlands

logs. Existing security log analysis solutions include software such as SentiLog and DeepLog which make use of specific algorithms rank security logs as low, medium, or high priority. Insufficient literature and confidential benchmarking metrics makes it difficult to assess the performance of security log analysis software[11].

A typical security log a text file which contains information about the user's operating system, time, and the requested resource. Analysts use software such as SentiLog and DeepLog to view security logs and receive insights on a security log's content. Using DeepLog and SentiLog enables analysts to make decisions on security logs, however, analysts struggle when analysing sophisticated attacks due to the volume of information contained within a log [2]. A session is a period where a user's behaviour is being monitored as they use a software, and one session can contain several security logs. When an analyst must classify an intrusion detection, they must reconstruct the session from the security log [8]. As a result, the analyst must not only reconstruct the session but examine the overall session in order to determine if a security breach has occurred. This involves examining user behaviour, IP addresses, and failed login attempts. An analyst must look at several pieces of data to label the security log and then the analyst must distinguish between common human behaviour such as having several login attempts versus a genuine attack.

The volume of information is particularly seen in signature based attacks. A signature based architecture stores malicious signatures and compares new signatures to malicious ones. For cybersecurity analysts the distinction is not obvious since some users will behave in a way in which logs are not easily distinguishable from logs where the user is acting as expected [8]. As a result, it is difficult for an analyst to distinguish the difference between a malicious user adapting their behaviour and a user who is behaving correctly. Experienced analysts are better able to distinguish the difference but are often not available due to their high turnover rate [2] . Furthermore, the lack of communication between novice and experienced analysts prevents skill transfers. When presented with a security log, analysts typically work on issues independently, resulting in more time being spent in problems and less skills being transferred from more experienced analysts[11].

Skill differences between experienced and novice analysts could be reduced using LLM to analyse security logs and provide insights to analysts. LLMs could limit the amount of skill needed to analyse a security log which could reduce the amount of time and work needed to analyse and classify security logs.

Benchmarking data for security log classification for a previous iteration of GPT (GPT-2) indicate false positive rates of between 0.756 and 0.875 for the baseline model and 0.005 and 0.121 for a fine tuned version[5]. A false positive rate is the rate which benign security logs get classified as malicious. A low false positive rate is useful indicates that benign logs will most likely be correctly classified limiting the number of logs analysts must analyse. There is no significant public research illustrating LLM's ability to classify malicious security logs so its ability to correctly detect malicious security logs is unknown. The next section addresses LLMs ability to detect and classify malicious logs.

3 METHODOLOGY

The objective of this paper is to understand the feasibility of using LLMs to classifying analysing security logs. The ATLAS dataset was selected for this study due to its range of security events from various sources, enabling GPT to recognize different types of incidents similar to real attacks. Likewise, the dataset also includes single-host attacks and multi-host attacks. A single host attack is an attack that targets a specific machine or user, and a multi host attack is comprised of coordinated actions are taken across multiple systems.

There are three types of security logs found in the dataset: firewall, intrusion detection system (IDS), and application logs. Firewall logs record traffic allowed or denied by the firewall, including details about the source and destination IP addresses, ports, and protocols used. Intrusion detection system (IDS) logs generate alerts when potential malicious activity is detected and provide details about the nature of the suspected attack. Application logs record user activities and system errors.

Data collection involved downloading raw security logs that were not labelled as benign or malicious. A single log entry from the ATLAS dataset contains the user or account that initiated the action, the target of the action, the process involved, and the specifics of the access request.

The entry typically identifies the user with a security ID, account name, and login ID. It describes the target object with details such as the object type and name. The security log also includes information about the process that performed the action, including the process ID and name. Additionally, each security log outlines the access request details and specifies the types of access requested. Given the dataset, functional and non functional requirements were made in order to define success criteria for security log classification. In order to compare the dataset against the functional requirements preprocessing the data was required. Preprocessing served to enable GPT to interpret the dataset.

3.1 Functional Requirements

Functional requirements are target requirements of a system with specific behaviours and outcomes.

FR1.1: The system shall preprocess logs to remove entries that contain noise with an accuracy equal to or greater than 0.95.

- AC1.1: The system is able to remove noise with an accuracy of at least 0.95.
- R1.1: Noise refers to the information in a security log which is not used. Most accuracy values target 0.90 to 0.99, however, due to the novelty of LLMs it is expected that it will make mistakes. A value smaller than 0.95 could significantly reduce the number of security logs being evaluated which could negatively affect the evaluation process of the methodology. In order to avoid this a threshold is higher than than the actual classification of security logs.

FR2.1: The system shall use GPT to classify log alerts with a target classification accuracy of at least 0.90.

- AC2.1: The system's classification accuracy is equal to or greater than 0.90.
- R2.1: There is no objective metric for cutoff in accuracy but common values range from 0.90 to 0.99. Since security logs are

complex and they are being analysed using novel technology, it is unlikely that GPT will consistently produce extremely accurate results which is why the value of 0.90 is chosen. Research by Ban. et al identify that LLMs are capable to escalate malicious security logs with an accuracy greater than 0.90 [3].

FR3.1: The system shall generate a textual response which indicates if a log is benign or malicious.

- AC3.1: For each log given, the system outputs a "0" for a benign log or a "1" for a malicious log.
- R3.1: A binary system was chosen in contrast to traditional classification methods in order to provide a baseline understanding of GPT's ability to classify logs. Furthermore, a binary classification system allows for scalability to employ traditional classification methods which comprise of low, medium, and high priority security logs.

Non Functional Requirements

Non functional requirements represent system objectives and criteria without attributing specific values and metrics.

NFR1.1: LLMs aid humans in a decision making process where humans are responsible for the classification of logs.

- AC1.1: The system shall provide suggestions and insights to human analysts without making autonomous decisions.
- R1.1: Onwukibo et al. claim that LLMs should not completely remove human interaction. This is because a significant challenge in log analysis is the over reliance on technologies such as machine learning to perform the analysis. The requirement for human interaction is also known as "human-in-the-loop". A "human-in-the-loop" incorporates a holistic decision which includes the "Social, human, financial, risk, reputation and otherwise" [9].

3.2 Preprocessing

The preprocessing steps for this study involves preparing raw logs for analysis using two different models: the GPT 3.5 base model and a fine tuned version of GPT 3.5. These models were chosen for several reasons. The first reason is the use of OpenAI's API which can automate large amounts of prompts and metric calculation. Similarly, GPT 3.5 is substantially cheaper than recent models such as GPT 4 which makes analysing a large amount of security logs cost effective. A base model and a fine tuned version were chosen in order to better understand the relationship fine tuning has on a model. This relationship allows for a comparison between the base and fine tuned model.

For the GPT 3.5 base model the security logs had to be separated into separate files to ensure that each input to the model did not exceed its processing limit of 60,000 tokens. This step was done to maintain data integrity while conforming to the GPT's constraints. Logs containing indicators of malicious activity were labeled as malicious, while logs without such indicators were labeled as benign. The labeling process allowed to assess GPT's performance by comparing its predicted labels to the actual labels.

Preprocessing data for the fine tuned model involved splitting the raw logs into two subsets: 70% of the data was reserved for training the model and 30% was set aside for testing. This split is a common ratio in machine learning. Fine-tuning the GPT 3.5 model required creating a .jsonl file that containing a minimum of 10 messages. A message consists two prompts: a prompt made by the user and the corresponding GPT response. The prompt given to GPT has a role of "system", while the response has a role of "assistant". The final message contains both prompts separated by a comma. An example system prompt is shown below:

```
{"role": "system", "content": "Process Information: SYSTEM Process ID:3776, Process Information: Source Address:0.0.0.0}
```

The security log contains information such as the ID of the system process, and source address. The assistant prompt with the correct classification is shown below:

```
{"role": "assistant", "content": "0"}
```

The final message will contain both prompts in a single line:

```
{"role": "system", "content": "Process Information: SYSTEM Process ID:3776, Process Information: Source Address:0.0.0.0},{"role": "assistant", "content": "0"}
```

Each message prompt contained 16 raw logs and the response prompt was the correct classification of those logs. A total of 10 messages were used to train the model. The choice of 10 messages was chosen for two reasons. The first reason is that the model's performance did not improve when using a larger number of messages. Likewise, the costs associated with fine tuning GPT encourages a lower number of messages due to the lower amount of resources required. Fine tuning substantially increased GPT's performance described later in this section.

3.3 Prompt Engineering

Prompt engineering serves to leverage GPT's abilities and provide insight into the effect prompts have on performance. Initially, the approach involved including one file containing raw logs and another separate file containing the corresponding malicious labels. However, uploading two files proved to be impractical due to GPT 3.5's token limit. Including malicious labels in the prompt itself was found to reduce the number of tokens in the prompt. Furthermore, GPT's token limit requires waiting for one minute until the limit is reset.

A chunk is a single file consisting of a subset of the raw security logs as described in preprocessing. Ten chunks were given in total for each prompt totaling circa 9000 security logs. Including more chunks for identification and classification would significantly increase the time for data collection and ten chunks appears to be a sufficient number of security logs. When reducing the number of logs, GPT would frequently analyse less security logs than given thus resulting in an insufficient number of classified logs.

When prompting the fine tuned model ten chunks from the training data were used. OpenAI offers parameters for influencing the text generated by GPT. There are three main parameters: temperature, presence penalty, and frequency which range from 0 to 1. Temperature adjusts the randomness of the generated text. A low temperature value (e.g. 0.1) produces more focused and predictable responses while a high value (e.g. 0.9) results in more diverse and

creative outputs. Presence penalty discourages the repetition of tokens. A high presence penalty value encourages the introduction of new concepts and a low value allows more topic repetition. Frequency penalty reduces the likelihood of repeating the same tokens in close proximity. For example, a high frequency penalty makes the model less likely to repeat words and a low frequency penalty does not penalize repetitions thus encouraging more frequent use of the same words. All parameters are set to 0 produce a classification which is not random, does not introduce new concepts, and stays within the same topic.

This paper examines three different prompt complexities: low, medium, high. While prompt complexities are subjective, a brief explanation is provided as to why the prompt belongs to the specified complexity shown below:

Low complexity prompt:

> Please classify each log as 0 (benign) or 1 (malicious). Write either a 0 or 1 followed by a space for each log in a single line.

This prompt was used as a low complexity prompt since it was straightforward. This did not involve passing in the malicious logs since it is reserved for the medium and high complexity prompt.

Medium complexity prompt:

> You are given logs which must be classified as 0 for benign and 1 for malicious. If a log contains a malicious label classify the log as 1. Please separate each classification by a space and in a single line. The total number of benign and malicious logs should be given at the end. One of the malicious logs is `0xalsaheel.com`

The methodology behind the medium complexity prompt was to introduce malicious labels in the prompt itself. The aim of the medium complexity prompt is to analyse if introducing malicious labels will aid GPT in identifying unseen malicious labels.

High complexity prompt:

> Suppose you are an SoC analyst responsible for identifying anomalies in the log entries using the labelled training data. You are responsible for classifying these anomalies as malicious (1) or benign (0). Identify the malicious labels and make sure to classify all logs given in the `.txt` file in a single line. After you are finished include the total number of classified benign and malicious logs where each classification is either a 0 or 1 separated by a space. Malicious labels are `0xalsaheel.com` and `c:/users/aalsahee/payload.exe`. An example response would be 0 1 0 1 0 0 where there are 6 security logs to be classified.

The high complexity prompt differs from the medium complexity prompt in which it employs a hypothetical scenario along with an example output. This prompt aims to examine the effect of security log classification by providing contextual information about the given task. After providing low, medium, and high complexity prompts performance metrics were calculated.

4 RESULTS

Each prompt would return a classification containing predicted labels which would be compared to the actual labels and performance metrics would be calculated as shown below:

- True Negative (TN): The number of benign logs correctly classified.
- False Positive (FP): The number of benign logs incorrectly labelled as malicious.
- False Negative (FN): The number of malicious logs incorrectly classified as benign.
- True Positive (FP): The number of correctly identified malicious logs.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Example performance metrics are shown below using a low complexity prompt in a single chunk:

Metric	Value
Accuracy	0.95
Precision	0.14
Recall	1.00
F1 Score	0.25

Table 1. Example Performance Metrics

Each prompt was analysed using 10 chunks and their average performance metrics were taken. The table below illustrates the prompt complexity used, total number of logs given, total number of logs analysed by GPT and average performance metrics:

Complexity	Logs Giv.	Logs An.	Acc.	Prec.	F1	Rec.
Low	8597	1252	0.722	0	0	0.88
Medium	3824	677	0.795	0	0	0.5
High	9534	2916	0.706	0.0005	0.001	0.9

Table 2. Average Performance Metrics by Prompt Complexity

The results indicate significant variations in the performance of the model across different levels of prompt complexity. For low complexity prompts, GPT achieved an accuracy of 0.722, a precision of 0, an F1 score of 0, and a recall of 0.88. A high accuracy but low precision suggest that although GPT was able to correctly identify a large number of relevant logs it also generated a substantial number of false positives. In the context SOC classification a high amount of false positives suggest that a large number of benign logs would be classified as malicious.

When increasing the prompt complexity to medium, the GPT's accuracy improved to 0.795, indicating a better overall performance. However, the precision and F1 score remained at 0, and the recall decreased to 0.5. A lower recall suggests that GPT is more likely to classify malicious as benign compared to the low complexity prompt. A high accuracy and low recall indicate that GPT is able to better identify security logs but it still struggles with high false positive rates.

For the high complexity prompt GPT demonstrated the highest recall but this was at the expense of accuracy, precision, and F1 score, which were all low. The drop in accuracy and particularly low precision suggest that given a higher prompt complexity, GPT generated a higher number of false positives. Across all prompt complexities the number of logs analyzed was significantly lower than the number of logs given highlighting GPT's limited processing capabilities. There appears to be a trade-off between prompt complexity and performance particularly when examining that higher prompt complexities present a higher accuracy but a higher number of false positives. Significant performance differences are seen when comparing the base GPT model shown below:

Complexity	Logs Giv.	Logs An.	Acc.	Prec.	F1	Rec.
Low	9421	968	0.976	0.937	0.876	0.823
Medium	6569	753	0.995	1	0.727	0.571
High	8697	1764	0.991	0.793	0.827	0.864

Table 3. Average Fine Tuned Performance Metrics

The fine tuned model showed improvements over the base model. For low complexity prompts the accuracy increased and had significant improvements in precision and F1 score which indicate a reduction in false positives. A balance between accuracy and precision suggests that GPT was able to better identify malicious security logs compared to the base model. For the medium complexity prompt the fine tuned model showed improvements in accuracy and a substantial increase in precision and F1 score compared to the base model although recall slightly decreased. An increase in accuracy and a decrease in recall indicates a balanced performance with fewer false positives and a reasonable detection rate. For the high complexity prompt the fine-tuned model achieved slightly lower accuracy than medium complexity while surpassing the base model's performance with improved precision and maintained a high recall. A high precision and high recall suggest that the high complexity prompt is able to handle complex prompts with improvements in the number of correct and incorrect malicious security log classifications.

5 DISCUSSION

The results demonstrate the differences in performance between the GPT 3.5 base model and a fine tuned version for security log analysis using three prompt complexities The base model showed high accuracy but struggled significantly with precision and F1 score, especially under low and medium complexity prompts, indicating a high rate of false positives. This high recall but low precision suggests that while GPT could identify a large number of relevant logs, it also incorrectly flagged several benign security logs as malicious leading to inefficiencies in threat detection. In contrast the fine tuned model had performance improvements in security log classification more specifically on precision suggesting a great reduction in incorrectly flagged benign security logs.

When reflecting on the methodology specific steps were particularly effective. The data from the ATLAS dataset provided a large range of security logs and allowed for results closer to those in the real world. Likewise, the decision to split the raw logs into 60,000 token chunks was necessary to conform to GPT 3.5's token limit. Shuffling the logs to balance benign and malicious entries ensured that GPT was exposed to a more balanced number of benign and malicious security logs which is more representative of real world SOC security logs.

Fine tuning GPT using a .jsonl file proved to be effective because of GPT's performance increase. The structure of giving security logs and receiving the correct classification in the fine tuning process enabled the GPT to learn from specific examples which could have contributed to its performance improvement. The choice of 10 messages enabled a balance between providing enough examples for GPT to improve without spending significant resources.

The methodology could be improved if more time and resources were available. For example, increasing the number of fine tuning messages beyond 10 could lead to even better performance. This is because larger and more diverse training set might allow GPT to better classify new and unseen security logs. Additionally, implementing more sophisticated preprocessing techniques such as advanced noise filtering could further improve the data quality resulting in higher performance. Another area for improvement is the performance evaluation of this paper. While accuracy, precision, recall, and F1 score are common metrics in machine learning, metrics such as the time taken for log analysis and the could provide a more comprehensive assessment of its applicability in SOCs due to the large amount of security logs needed to be analysed.

5.1 Limitations

The dataset only contained security logs from the ATLAS dataset which most likely does not cover the full spectrum of possible security events encountered by SOCs. As a result, this could affect GPT's ability to classify new types of security logs.

Computational constraints when processing security logs influenced the methodology. The decision to split logs into 60,000 token chunks was resulted from GPT's processing limits and may not be able to handle longer security log sequences. When comparing performance metrics, values such as true positive, true negative, false positive, and false negative were not recorded during data collection thus reducing the amount of information for security log classification. Furthermore, GPT's processing limits required a minute of idleness after classifying a chunk resulting in a significant time increase when classifying security logs.

The methodology for fine tuning GPT 3.5 was relatively basic. Fine tuning was done by uploading the .json file with example messages to OpenAI's fine tuning system. As a result, the specific algorithms used for fine tuning are not known. More sophisticated fine tuning techniques combined with larger and more complex datasets could yield better results. Furthermore, the fine tuning file was limited to 10 messages due to the financial cost associated with

fine tuning. A substantially larger number of messages could prove effective in increasing GPT's learning capability. GPT's learning capability could reduce its performance over time as new types of attacks emerge, suggesting the need for continuous model updates and retraining for new types of security logs.

The categorization of security logs as benign or malicious is different than the classification methods SOCs use. Most log analysis software classify logs based on their priority level comprising of low, medium, and high priority. While a binary classification method can distinguish a security log simply it cannot prioritize security logs which is a common practice in SOCs.

5.2 Further Research

Further research involves improving the methodology for security log detection and classification. Security logs are currently classified as benign or malicious and by developing a model that can also assign priority levels (low, medium, high), researchers can create a more nuanced threat detection system that aligns better with traditional SOC workflows. One of the main limitations of the ATLAS dataset is that the types of logs fall short compared to those received by SOCs. Expanding the dataset to include a wider variety of log sources and types, could enhance GPT's ability to perform effectively across a larger number of real world environments. Additionally, continuous fine tuning and retraining of GPT with newer models and up to date datasets could better adapt to the evolving nature of cyber attacks. Examining GPT's time performance could offer a more comprehensive evaluation of its effectiveness for daily SOC operation. Research needs to be done to benchmark current SOC software and enable direct performance comparisons between traditional and novel methods of security analysis.

REFERENCES

[1] Wasyihun Admass, Yirga Yayeh, and Abebe Diro. 2023. Cyber Security: State of the Art, Challenges and Future Directions. *Cyber Security and Applications* 2 (Oct. 2023), 100031. https://doi.org/10.1016/j.csa.2023.100031

[2] Enoch Agyepong, Yulia Cherdantseva, Philipp Reinecke, and Pete Burnap. 2020. Challenges and performance metrics for security operations center analysts: a systematic review. *Journal of Cyber Security Technology* 4, 3 (July 2020), 125–152. https://doi.org/10.1080/23742917.2019.1698178

[3] Tao Ban, Takeshi Takahashi, Samuel Ndichu, and Daisuke Inoue. 2023. Breaking Alert Fatigue: AI-Assisted SIEM Framework for Effective Incident Response. *Applied Sciences* 13, 11 (Jan. 2023), 6610. https://doi.org/10.3390/app13116610 Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[4] Chris Egersdoerfer, Di Zhang, and Dong Dai. 2023. Early Exploration of Using ChatGPT for Log-based Anomaly Detection on Parallel File Systems Logs. In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*. ACM, Orlando FL USA, 315–316. https://doi.org/10.1145/3588195.3595943

[5] Egil Karlsen, Xiao Luo, Nur Zincir-Heywood, and Malcolm Heywood. 2023. Benchmarking Large Language Models for Log Analysis, Security, and Interpretation. http://arxiv.org/abs/2311.14519

[6] Kaspersky Lab. [n. d.]. *DAMAGE CONTROL: THE COST OF SECURITY BREACHES*. Technical Report. https://media.kaspersky.com/pdf/it-risks-survey-report-cost-of-security-breaches.pdf

[7] Scott Musman, Mike Tanner, Aaron Temin, Evan Elsaesser, and Lewis Loren. 2011. Computing the impact of cyber attacks on complex missions. In *2011 IEEE International Systems Conference*. 46–51. https://doi.org/10.1109/SYSCON.2011.5929055

[8] Adam Oliner, Archana Ganapathi, and Wei Xu. 2012. Advances and challenges in log analysis. *Commun. ACM* 55, 2 (Feb. 2012), 55–61. https://doi.org/10.1145/2076450.2076466

[9] Cyril Onwubiko and Karim Ouazzane. 2019. Challenges towards Building an effective Cyber Security Operations Centre. *International Journal on Cyber Situational Awareness* 4, 1 (Dec. 2019), 11–39. https://doi.org/10.22619/IJCSA.2019.100124

[10] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. GPT-4 Technical Report. http://arxiv.org/abs/2303.08774

[11] Manfred Vielberth, Fabian Bohm, Ines Fichtinger, and Gunther Pernul. 2020. Security Operations Center: A Systematic Study and Open Challenges. *IEEE Access* 8 (2020), 227756–227779. https://doi.org/10.1109/ACCESS.2020.3045514