Evaluating Large Language Models for Automated Cyber Security Alarm Analysis Processes

BEAU JONKHOUT, University of Twente, The Netherlands

Cybersecurity faces ever-increasing complexities and novel threats, with recent reports from CrowdStrike, IBM, and ENISA highlighting the appearance of 34 new adversaries and a 15% increase in cyberthreats. Traditional security tools and methodologies such as the NIST CSF 2.0 are under pressure to evolve rapidly to keep pace. The enormous volume of security alarms leads to "alert fatigue," a condition where the overwhelming volume of security alarms causes individuals to become desensitized and less responsive, which can cause significant oversight in threat detection. This thesis investigates the application of Natural Language Processing (NLP), particularly Natural Language Understanding (NLU) and Large Language Models (LLMs) like OpenAI's GPT series, to streamline the decision-making processes in cybersecurity alarm analysis. By delegating key decision-making aspects to LLMs, this approach seeks to mitigate alert fatigue. The research evaluates current state-of-the-art models, develops a methodology for assessing the efficacy of LLMs, and analyses their capabilities in specific security analysis. Results indicate that LLMs match human response levels to a non-random degree, suggesting that they can potentially support operators in reducing alert fatigue, but need further optimization. Data and findings are made available publicly to facilitate further research and verification.

Additional Key Words and Phrases: Artificial Intelligence, Natural Language Processing, Natural Language Understanding, Alarm Analysis Process, Large Language Models

1 Introduction

Cybersecurity is an ever-evolving field, facing increasingly complex and frequent threats. The Global Threat Report 2024 by CrowdStrike reports 232 distinct adversaries, with 34 new ones emerging [45]. IBM notes a 15% increase from 2020 to 2023 in the global cost of a data breach, now averaging $4.45 million [11], while the European Union Agency for Cybersecurity (ENISA) observes a trend toward larger, i.e. more complex DDoS attacks [9]. Traditional tools such as Zeek [33] and Snort [30] struggle to keep pace, as does the NIST Cybersecurity Framework 2.0 [25], despite its comprehensive guidelines. The rising volume of security alarms, a significant challenge for operators, leads to *alert fatigue*, where the overwhelming number of repetitive, low-quality alerts causes critical threats to be overlooked. This issue was highlighted in an International Data Corporation report, which found that 83% of professionals struggle with alert volume, leading to missed threats [60]. Furthermore, a ReliaQuest survey reveals enterprises typically deploy 19 different security tools, yet only 22% are considered essential, which increases *alert fatigue* [58].

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

To address this fatigue problem, automating the analysis process could significantly alleviate the workload. For instance, a security operator monitoring events flagged by an intrusion detection system (IDS) such as Snort [30], which operates on predefined rules and triggers alarms for specific activities such as five failed login attempts, faces a decision-making process. The initial alarm raised after five incorrect password entries does not necessarily confirm a brute force attack, which would require the operator to perform several validation steps. These might include identifying the source IP address and assessing whether it is from a known device and location. While the latter two steps can be automated through traditional programming logic, human reasoning is necessary for other tasks. For example, a person might be on vacation, and therefore has an unknown location, or a specific person might characteristically make many mistakes, and therefore causes many errors. In these cases, both scenarios should not be mistakenly classified as brute force attacks. This kind of decision-making demands human insight to accurately interpret the context of the errors.

To automate the human decision-making, this paper advocates for the use of Natural Language Processing (NLP), a branch of Artifical Intelligence (AI) that enables machines to understand and interpret human language. Within the realm of NLP, Natural Language Understanding (NLU) focuses specifically on comprehending the intent and meaning behind the text. This study suggests employing NLU to enhance automation in areas of the human reasoning process that traditional programming logic struggle to manage effectively. NLU leverages Large Language Models (LLMs), such as OpenAI's Generative Pre-Trained (GPT) series [35, 56], which are deep learning-based models designed to process and understand text which can do an accurate job of human reasoning. LLMs are important, because they can help to analyze large volumes of threat data, identify patterns, and provide insights that human analysts might miss. LLMs have already shown significant potential in automating complex tasks [13], including cybersecurity [62]. LLMs have the potential to significantly automate the security analysis process, accurately assessing the validity of alerts with a nuance that resembles human judgment, while being less susceptible to *alert fatigue*.

This study intends to define a framework employing LLMs to improve, or perhaps even surpass, human performance in the alarm analysis process. This framework will establish guidelines to assess whether some, or potentially all, tasks involved in analysing a potential threat can be automated using LLMs. This is done by delegating more responsibility and decision-making to LLMs through carefully constructed prompts, where a prompt is an instruction fed to the LLM to provide an output. This delegation can reduce *alert fatigue* on security operators, in order to eventually improve performance of a security operator handling security alarms. The framework will be designed to be applicable to several LLMs, thus staying relevant when the state-of-the-art LLMs change. The effectiveness of LLMs

for each stage of various alarms will be evaluated, determining how the LLM can take over the decision-making process from humans at *each step*.

To automate the security analysis process, the following research questions (RQs) have been defined as the basis of this research:

- **RQ1:** What are the current large language models available for automating the analysis of security alarms and what are the most important steps in the security alarm analysis?
- **RQ2:** How can a comprehensive evaluation methodology be developed to assess the effectiveness of different LLMs in the context of security alarm analysis?
- **RQ3:** What are the advantages and limitations of different LLMs when analysing one specific security alarm?

The remainder of this proposal is organized as follows: Section 2 addresses the first research question and provides related work on the subject. Section 3 answers the second research question by presenting a comprehensive framework for security analysis automation. Next, Section 4 applies this framework to state-of-the-art models to explore the advantages and disadvantages of using LLMs in the security analysis process. Finally, this paper will end with a conclusion, limitations and future work.

2 Related Work

In this section,the first research question will be addressed, which focuses on exploring the available LLMs and identifying the key steps in analysing security alarms, so that these steps can be automated.

2.1 From Artificial Intelligence to Large Language Models

To understand the what LLMs are, it is essential to understand where they come from. Therefore, an overview will be presented in the this section. AI represents the foundation of machines mimicking human-like intelligence, encompassing a spectrum of technologies that enable machines to perceive, understand, act, and learn [44]. A subset of AI, ML, involves algorithms that learn from data to improve decision-making over time [57]. Within ML, NLP empowers computers to process and generate human language, playing a crucial role in applications such as translation and speech recognition [44]. Advancing further, NLU delves into interpreting human language in a meaningful way, essential for tasks such as language inference [43] and question answering [53].

Language Models predict word sequences in text, providing the backbone for NLP tasks. LLMs, such as the GPT [56], extend this concept by leveraging deep learning to generate text that can closely mirror human writing. Characterized by their billions to trillions number of parameters and complexity, these models are important in enhancing NLU and supporting more sophisticated applications, such as intent recognition in security event analysis [14]. This narrative journey from AI to LLMs showcases their evolution and underscores their potential in advancing the understanding and implementation of complex language-based tasks.

2.2 From Transformers To LLMs

This aforementioned NLP and NLU processes are implemented through a series of steps, namely through models which keep improving, eventually arriving at the current state-of-the-art LLMs. In this section, the journey from the transformer model to ultimately GPT-4 is highlighted. Vaswani et al. [61] introduces the transformer model, which has been influential in the field of NLP. This model is the base for many popular language models, such as Bidirectional Encoder Representations from Transformers (BERT) [40], Generative Pre-training Transformer 1 (GPT-1) [56] and OpenAI's GPT-4, [35]. This influence is visible because BERT has been used as the basis for many language models, such as SecureBERT [36], SecurityBERT [41], RoBERTa [48], ALBERT [46] and BART [47]. At the same time, GPT's owner OpenAI has received an 80 Billion $ valuation of OpenAI [28] in 2024. This influence could be possible because the transformer model processes large datasets efficiently and can recognize complex patterns in data. Its importance lies in the sense that the language models, that will be discussed, use this model or some variant of it to process remarkable results.

Building up on BERT, BERT [40] is a deep learning model for natural language processing designed by Google. It stands out for its use of bidirectional training of transformers, meaning it learns information from both the left and right context of a token within a sentence. This two-way ability significantly improves its effectiveness and understanding in tasks such as sentiment analysis, question answering, and language inference, evidenced by high rankings on NLP benchmarks [3].

Furthermore, the introduction of Generative Pre-Training 1 (GPT-1) by Radford et al. [56] shows another remarkable step in NLP. Utilizing a variant of the transformer model, GPT-1 serves as a predecessor leading to the more sophisticated GPT-4 by OpenAI. It introduces a novel approach to achieve strong natural language understanding. This is done via unsupervised pre-training on a large corpus of text, which then uses supervised fine-tuning for specific tasks.

These developments highlight a significant evolution in NLP from simpler, one-directional models to complex, context-aware systems that underpin modern LLMs. The progression from the original transformer to advanced models such as GPT-4 has fundamentally changed how machines process and generate human language. This shift allows for more accurate and natural interactions between computers and humans. Therefore, showing opportunity to apply LLMs to the analysis process.

2.3 Available Large Language Models

To establish a well-defined list of LLMs, the search engine Google was utilized [21] was utilized and the first ten popular LLMs were selected. Additionally, the five most popular models from Ollama's [27] library as of May 2024 were included. Ollama is an open-source project that serves as a powerful and user-friendly platform for running LLMs on a local machine [34]. The combined list from Google and Ollama comprised the following models: Llama, GPT-4, Falcon, PaLM 2, Claude, Cohere, BLOOM, LaMDA, Orca, BERT, Gemma, Llama3, Mistral, Qwen, and Phi3.

The list will then be refined as follows: **1)** Removed duplicate entries of Llama, keeping only Llama 3; **2)** Specified versions as PaLM to PaLM 2, Qwen to Qwen 1.5, and Orca to Orca 2, to use the latest version in May 2024; **3)** Excluded Cohere and LamDA due to domain-specific focus; **4)** Excluded non-LLMs such as BERT for having significantly fewer parameters.

Name	Release Time	Public	Number of Parameters
GPT-4 [16]	2023	✗	1.76T[1] [38]
Falcon [20]	2023	✓	40-170B
PaLM 2 [12]	2022	✓	540B [8]
Claude 2 [10]	2023	✗	130B [55]
BLOOM [5]	2022	✓	176B
Orca 2 [51]	2022	✓	7-13B
Gemma [37]	2023	✓	2-7B
Llama3 [23]	2024	✓	8-70B
Mistral [50]	2023	✓	7B
Qwen 1.5 [29]	2024	✗	0.5-110B
Phi3 [39]	2024	✓	3-14B

Table 1. Overview of popular LLMs

Table 1 presents a list of eleven different LLMs, after the adjustments, each representing the state-of-the-art LLMs in May 2024. The selection criteria ensure that the most advanced and capable will be provided, and therefore relevant LLMs, allowing for a comprehensive analysis of their potential and applications in the alarm analysis.

2.4 Key Steps in Security Alarm Analysis

In the process of alarm analysis, the Security Operations Center (SOC) plays an important role in defending its organisation from cyberthreats. This section will discuss how SOCs detect potential security incidents, the tools they use, and the subsequent steps they take to mitigate these threats. The initial and crucial step in this process is the detection of potential security incidents.

A SOC utilizes Security Information and Event Management (SIEM) systems, such as Splunk [31] or LogRhythm [49], to monitor and analyze security alerts generated by various network devices such as firewalls, intrusion detection/prevention systems (ID/PS) and system errors. These SIEM systems gather and correlate different types of security data. Once a specific security rule has been violated, then the SIEM will notify the SOC. Extended Detection and Response (XDR) tools, such as Microsoft 365 Defender [24], enhance SIEM [32] systems by integrating and correlating alerts from specific vendors. The SIEM, along with this XDR integration, improve the security response by making it faster and more effective.

To further optimize the efficiency and effectiveness of these security tools, Security Orchestration, Automation, and Response (SOAR) automation is employed. SOAR frameworks, such as Cortex XSOAR [19], leverage guidelines from standards such as ISO/IEC 27001 [22] to define specific rules for SIEM systems, thereby standardizing responses and automating processes in threat detection and mitigation. Furthermore, the primary difference between SIEM and SOAR is that while SIEM alerts the SOC about potential threats, SOAR also automates response actions, such as quarantining malware. In conclusion, these systems, such as SOAR and SIEM, can be employed for the alarm analysis. In this study, it is assumed that detection is well done, and it needs to be understood where in this system an LLM could potentially replace human decision-making.

The next step in the analysis is to determine whether a triggered alarm (detection) is legitimate or not, and second, whether follow-up actions should be taken. In the next section, two alarms – one for phishing and one for brute-force – selected from the MITRE ATT&CK [52] website will be analysed. The information – regarding the alarm analysis – has been reconciled with a security analyst from Northwave Security [26].

MITRE ATT&CK [52] provides a detailed database of adversary tactics and techniques, essential for enhancing the alarm analysis process. By mapping alarms to specific tactics (e.g., Initial Access, Execution, Persistence) and associated techniques (such as "Brute force" or "Phishing), the framework helps categorize and contextualize each alarm within the attack life cycle (i.e. identifying which stage the attacker is), which is needed to mitigate the threat. This database provides a detailed list of cyberthreats, with 202 techniques and 435 sub-techniques for enterprises alone. However, the focus here was only on two types of alarms: brute force and phishing.

2.4.1 Brute Force Alarm A brute force attack involves an attacker systematically attempting every possible combination of passwords or keys until the correct one is found, aiming to gain unauthorized access to a system. When the Security Information and Event Management (SIEM) system triggers a brute force alarm, the Security Operations Center (SOC) should first diligently examine the security logs. This examination entails scanning for patterns of failed login attempts, identifying the specific accounts targeted, and noting the timing of these attempts. Recognizing any unusual patterns in login failures or focused attempts on high-value or privileged accounts is crucial, as these are common indicators of a brute force attack. This analysis is typically conducted using queries—a form of traditional programming logic where LLMs may not be as effective due to their inability to perform server calls directly.

In addition to log analysis, the SOC should conduct a thorough context analysis to assess the situation further. This involves examining the types of accounts involved, such as distinguishing between admin and regular user accounts, and reviewing the geographical locations of the login attempts. For instance, a login attempt from a region where the user has never accessed the system before may raise suspicion. However, an edge case such as an employee logging in from a new location while on holiday—potentially triggering the brute force alarm—might actually be benign. This highlights the importance of integrating threat intelligence, such as checking whether the IP addresses involved are known in threat databases or log files for previously observed malicious activity. Finally, the SOC may interact with users directly to verify the nature of the attempts, ranging from sending emails to making phone calls. While an LLM may not assist directly with server queries or user communication, it can effectively draft informative emails for users, leveraging the appropriate context to aid the SOC in managing potential threats and distinguishing between genuine security concerns and false alarms.

2.4.2 Phishing Alarm Phishing is a type of cyberthreat where attackers attempt to trick individuals into revealing sensitive information (such as passwords or credit card details) by posing as a trustworthy entity. Once the SIEM has detected activity or patterns in the data that resemble a phishing attack, then the SOC should conduct a different investigation.

Firstly, as a SOC Training showcases [42], the SOC should review the specific details of the phishing alert (log analysis), such as the

sender's email address, the subject line, and the content of the email. Phishing emails often contain urgent calls to action or requests for sensitive information, which are telltale signs of phishing attempts. Additionally, poor grammar and or spelling errors are common for phishing. These indicators can determine the validity of the alarm judging purely on content. An LLM is capable of determining sentiment in text, which theoretically means it could determine whether emails are urgent.

In the next step, the SOC conducts a detailed review of the email headers [59]. This includes tracing where the email originally came from to determine if the sender's domain is authentic or has been faked—a common tactic known as "spoofing." Spoofing occurs when attackers disguise their email to make it appear as if it comes from a trustworthy source, such as a well-known company. By examining the "Received from" fields in the email headers, the SOC can spot any inconsistencies that might suggest the email isn't what it claims to be. Such discrepancies often signal that the email may be part of a phishing attempt, designed to deceive the recipient. This is conducted through queries, where LLMs may not be as effective.

Furthermore, user interaction is also required. The SOC should verify the email's authenticity, if the sender is seemingly trustworthy, by contacting the supposed sender through a different communication method, such as a phone call. This can confirm whether the email was genuinely sent or if their account has been hacked. At the same time, it is vital to contact the recipient(s) and determine whether they have interacted with the phishing email, such as clicking on links, downloading files or responding to it. It is possible that they provide more information which can determine it validity. To repeat, the LLM can help with drafting informative emails.

2.4.3 Consistencies This analysis still reveals common steps in handling different security threats, despite the broad range of potential alarms outlined by MITRE. To define the key steps in security analysis, the main commonality between these two alarms will be covered. The commonality lies in log analysis, contextual understanding, and human interaction. Both alarms involve analyzing logs, such as IP addresses. Placing each alarm in its context enables the SOC to understand its true significance. For a brute force alarm, this means looking at the types of accounts involved or the geographical locations. For phishing, it involves verifying the sender's authenticity. In both situations, contextual knowledge is essential, requiring human reasoning to determine the nature of the threat. To replace this human reasoning, LLMs can be used. Additionally, human interaction is critical in both scenarios, as discussions may reveal new information that impacts the final decision. Ultimately, these steps highlight the existence and importance of nuanced human judgment in determining the validity of an alarm, where the judgement can be delegated to LLMs.

This section provides a foundation for understanding the capabilities of these models, discussing the evolution of AI, moving from general AI, ML, NLP to NLU, which contributes to the development of LLMs. It clarifies the origins of many popular LLMs, emphasizing the revolutionary Transformer architecture, which underpins many state-of-the-art models such as GPT-4. Finally, it presents a table listing eleven popular, state-of-the-art LLMs and discusses their potential role in enhancing human decision-making in security analysis, allowing for more effective contribution to the automation of these processes.

Furthermore, this section reviews the key steps involved in the analysis of security alarms within a SOC and explores the potential role of LLMs in automating these processes. By detailing the integration and functionality of SIEM, XDR, and SOAR systems, it establishes an understanding of how LLMs could enhance alarm analysis. Additionally, it examines specific examples of brute force and phishing alarms, showing overlap in log analysis, context understanding, and human interaction. This analysis suggests that LLMs can be employed more effectively on contextual textual tasks, while being less effective in API calls. Through this exploration, the research question regarding the identification and automation of key steps in security alarm analysis is addressed.

3 Evaluation Methodology

This section addresses the research question of how to develop a comprehensive evaluation methodology to assess the effectiveness of different LLMs in the context of security alarm analysis.

The ultimate goal is to discover the extent to which LLMs can be applied to security alarm analysis. To achieve this, a framework for comparing state-of-the-art LLMs with general alarms is proposed. This approach has been designed to stay relevant even when the state-of-the-art LLMs change or when new adversaries emerge. The framework is structured into three stages: preparation, computation, and interpretation of results.

The primary method to evaluate different LLMs involves comparing their conclusions with those drawn by their human counterparts. Both the LLM and the human (such as a security operator) are provided with a specific output request via a prompt. A prompt is a detailed instruction specifying what the LLM should generate, based on structured and precise information for each step of the alarm analysis. This same prompt is also given to a human, who should be able to execute the security step as well. If the LLM and the human provide identical responses, it is assumed, for the purposes of this study, that the LLM can perform equivalently to a human in that particular scenario. By evaluating this for a large amount of scenarios per step, the general picture can be understood whether responsibility can be delegated to the LLM.

For the rest of this section, the focus will be on how to systematically construct, i.e., gather the relevant information for, such a prompt for each particular case to eventually evaluate different steps of the analysis process.

3.1 Preparation

The prompts need to be populated such that the LLMs are equipped to answer the question. For generality, each prompt will have the same structure. **System prompt:** This is an instruction explaining **what the LLM is** to provide a better domain-specific answer, such as *You are assisting a security analyst in evaluating alarm events. You can only answer with 'yes' or 'no'.* **Context:** This is a text block containing **information** relevant to answer the question, such as security logs in the event of brute force attacks or emails in the case of phishing attacks. **Scenario prompt:** For each particular case, the LLM will receive a **tailored prompt**. This prompt will be a clear sub-task of a step: For analysis of the email, it can be *"Please analyse*

the subject line and sender's email address. Do any of these imply phishing?".

3.1.1 Determining the alarm and its relevant information Determining the context and scenario prompt is dependable on the specific alarm. A phishing alarm has different context than a brute force alarm. However, both of these prompt sections should be determined at some point as well as the baseline, because the baseline is used to compare the results with. This general preperation (i.e. investigation) is essential of the framework. The preparation steps are below.

Alarms: Select the alarms you want to compare from the MITRE ATT&CK [52] framework.

Steps: Discover the steps required in the analysis process for each alarm. This information should be extracted from MITRE. Eventually, this should be the scenario prompt. This scenario prompt is designed to help refine the final answer for the task. By generating multiple scenario prompts, you can choose the most effective one to ensure the highest accuracy from the LLM.

Filtering the steps & Gathering Context: Go through each step and assess whether a human should perform this step or a simple API call or query can gather the information. If the latter is the case, then this should be added to the context. If steps involve both an API call and human reasoning, then this step is removed for not being in the scope. The end result should be a list of steps where the human should use reasoning based on the information accessible to them.

Establish a Baseline: Decide on the expected outcomes for each step in the analysis, to use as a baseline for comparison. For instance, the urgency of an email can easily be determined by a human; it should either be urgent or not urgent. This baseline establishment should mean labeling the data for each step. This can be based on literature review, authoritative confirmation, and/or a security company playbook.

LLM Selection: Select LLMs to evaluate. In section 3.1.2, more guidelines are presented in order to select the LLM.

Structure the information: Structure the information so that it can be used in the evaluation. In section 3.1.3, more guidelines are presented in order to successfully structure the information.

3.1.2 Selection of LLMs To evaluate LLMs, it is necessary to have LLMs to evaluate, referred to as `language_models` in the code. For this study, any general-purpose LLM can be used. General LLMs are specifically chosen over domain-specific models due to their broad versatility, which allows them to handle a wide range of tasks, including text summarizing and log analysis. Furthermore, employing general LLMs simplifies the comparison between different models, as they are assessed based on a uniform set of capabilities rather than niche, specialized functions.

In order to select the most suitable LLMs, other factors must also be considered, including availability and performance. Public LLMs are more readily available than private models, making them easier to access. However, specific private models, such as GPT-4, are renowned for their superior performance in benchmarks, which suggests they may be more effective. For details on the leading LLMs, refer to the table 1. Performance is a crucial metric, primarily determined by benchmarks. The mentioned table highlights LLMs that excel in popular benchmarks such as the Massive Multitask Language Understanding (MMLU) [7] and the Discrete Reasoning Over Paragraphs (DROP) [54]. Limited information is available on benchmarks specific to cybersecurity. Other benchmarks, such as ImageNet [6] or HumanEval [17], focus on particular areas such as image or code generation and are therefore not relevant to this study, which does not require image processing or code generation.

Therefore, when selecting LLMs, it is important to consider a balance of generality, availability, performance. General models can be easily compared and are versatile for different tasks. Publicly available models offer ease of access and lower costs, which is crucial for widespread adoption and experimentation. Performance, as evidenced by benchmarks, provides a measure of a model's capability and efficiency in handling complex tasks.

3.1.3 Structuring of the information After gathering the information needed, it is essential to structure the information, such that it can be interpreted by an algorithm, where the role of the algorithm is to evaluate the LLMs.

`language_models`: A list of LLMs, which are selected to evaluate. **`alarms`**: A list of alarms used as scenarios, functioning as keys for each of the next items. **`steps_per_alarm`**: Maps each alarm to a list of steps. Each step is a sentence that outlines the actions the LLM (or operator) should take. **`baseline`**: Maps each alarm to a list of information, containing all necessary context and the correct answer for the LLM. Each alarm consists of a list of lists, with test cases and expected results, providing the program with relevant benchmark information.

`prompts_per_step_per_alarm`: Maps each alarm to a list of pairs. Each pair contains several (in this study, three) scenario prompts for each analysis step. Diverse prompts help select the best one for an optimal result.

`system_prompt_per_alarm`: Maps each alarm to a specific string that the algorithm should output (system prompt). This can instruct the LLM to respond in a predictable and comparable manner, such as solely answering 'true' or 'false'.

The structured information above is necessary to assess the performance of an LLM. Recall that this structure is different from the the structure in the prompt described in the beginning, this is because the structured information has all the data necessary to construct a such a prompt. This prompt is single scenario which can be tested, while the information above has all the information needed to construct all scenarios. In the next section, the discussion will focus on how this information can be used to evaluate LLMs.

3.2 Computation

In this section proposes guidelines for an algorithm, which integrates the six aforementioned factors: *language_models*, *alarms*, *steps_per_alarm*, *prompt_per_step_per_alarm*, *baseline*, *system_prompt_per_alarm*. The algorithm's output is systematically organized into a table that evaluates several language models (denoted by 'n') across multiple prompts (typically three). The process involves iterating over each step (denoted by 'k') of the analysis to collect and record specific key performance indicators (KPIs). These KPIs include the *accuracy* of each response, the *time* taken to generate the response, and the *character length* of the response.

The structure of the output table is detailed in Table 2. Each column within this table corresponds to a specific language model and prompt combination, capturing the aforementioned metrics for comprehensive evaluation and comparison across different scenarios.

	LLM_1			**LLM_2**			…	**LLM_n**		
Steps	p1	p2	p3	p1	p2	p3	…	p1	p2	p3
1							…			
…	…	…	…	…	…	…		…	…	…
k							…			

Table 2. Example Output of the Evaluation Benchmark

The algorithm should evaluate LLMs by systematically processing each one with a standard prompt set for all models. For each predefined step in the evaluation process, the algorithm cycles through various prompts linked to that step. It combines these prompts with scenarios outlined in a baseline document and the main system prompt to form a comprehensive test prompt for the model.

As the LLM responds to each combined prompt, the algorithm records the response time and compares the response to expected outcomes listed in the baseline. This comparison process should use multiple filtering layers to handle different scenarios, to automate some of the effort. For cases that remain unclear or unresolved by the filters, a human should make the final decision. This comparison helps assess the model's accuracy in understanding and reacting to the scenarios. The results for each model, including response times and accuracy measurements, are compiled into a summary table, following the structure above, providing a detailed view of each model's capabilities across different test cases.

3.3 Interpretation

The output of the algorithm, `llm_scores`, provides a quantified measure of each LLM's performance across different alarm scenarios. This output should be interpreted as follows:

Score Analysis: Higher scores indicate a better alignment of the LLM's responses with the expected baseline responses, suggesting greater efficacy in handling the specific alarm context.

Time Analysis: Time is crucial in determining costs. Longer times may lead to higher expenses and may reduce the relevance to the current situation if time is limited.

Character Length Analysis: When a LLM's response matches the expected length based on the prompt, it indicates effective and predictable performance. If asked for a simple "True" or "False" and the model instead provides a lengthy response before concluding, this suggests a lack of obedience to instructions and reduced predictability.

Model Comparison: By comparing scores across different models for the same alarms, stakeholders can determine which model is more effective at understanding and analyzing security alarms.

Alarm Analysis: Comparing scores across different alarms for the same model can reveal which types of alarms the model handles better, highlighting potential areas for improvement in model training or prompt design.

Prompt Analysis: Assessing how different prompts affect the model's responses reveals which are most effective, guiding improvements in prompt design for optimized performance

This concrete interpretation explains on how to interpret the results effectively. It takes all KPIs into account, while including analysis across different prompt, models and alarms. With these guidelines, it is possible to systematically assess different models, when the LLMs or alarms change.

This section outlines a three-phase evaluation method for assessing how well LLMs handle security alarm analysis. Therefore, it concretely shows what steps need to be taken in order to effectively evaluate different LLMs in the analysis process. In the preparation phase, all necessary information about different security alarms is gathered, a baseline for expected answers is set, and choices are made on how to test LLMs based on their availability and performance. This phase ensures that everything is set up correctly for a systematic comparison of the LLMs.

The next stages involve applying the specific algorithm (computation phase) to see how the LLMs perform and then looking at the results (interpretation phase) to understand which models did the best. The computation phase simulates how a security analyst would use the LLMs, providing them with scenarios and relevant data to see how they respond. The interpretation phase then uses these responses to measure each model's accuracy and effectiveness. This helps determine which LLMs could be useful in real-world security settings, providing clear metrics to guide future integration and improvements.

4 Applying the Framework

This section addresses the third research question by evaluating the advantages and limitations of different LLMs in analyzing three specific security alarms. This question will be answered by implementing the framework discussed in Section 3. The framework, and therefore the analysis, will be implemented using Python 3.9.0 [4] and a Jupyter Notebook [1], which facilitates a structured approach through modular Python components. This Jupyter notebook can be found on this Github [15].

4.1 Preperation

The initial step in this framework is the "Preparation" step, focusing on selecting an appropriate alarm. Phishing is selected due to its singificant involvement of human reasoning in detection and analysis, as detailed in Section 2.4.2. Brute-force alarms are excluded due to time and resource constraints.

During this phase, relevant steps are refined by extracting those requiring human reasoning, for this case solely email analysis. Open-source databases, such as the Apache Public Corpus [2] and GitHub's Phishing Pot [18], are used to source email samples. The Apache Public Corpus is valuable for its semi-spammy yet benign emails, underscoring the research's aim to assess if a LLM can differentiate between semi-spammy and genuinely dangerous phishing emails.

To label 200 emails, each is examined, and the four questions a security operator would address are answered, including analyzing the sender's email and the urgency. The results are saved. The next framework step is selecting the LLM.

Phi3:14B, Qwen1.5:4B, and Llama3:8B were selected based on performance, availability, and efficiency with the available hardware: an 8-core Intel I7 processor and an RTX 2060 graphics card. Preliminary analysis demonstrated that these models consistently

responded within 30 seconds. In contrast, larger models, such as Llama3:70B, required 5 minutes to process, rendering them impractical for handling the volume of prompts necessary. Namely, each of the three chosen models will process 2400 prompts (3 prompts for each of 4 steps across 200 emails), meaning it can take 200 hours to compute 2400 5-minute prompts for a single LLM. Consequently, these models provide a balance of performance and efficiency while achieving reasonable MMLU benchmark scores, as shown in Table 3.

Model	Params	MMLU	DROP	Public
Phi3	14B	78.0	-	✓
Llama3	8B	68.4	79.7	✓
Qwen1.5	4B	56.1	-	✓

Table 3. Performance of Selected LLMs

After selecting the models, the next step involves drafting prompts, as discussed in Section 3. A prompt is structured into three key components: a context, a scenario prompt, and a system prompt, with the order being crucial. Through testing, it was established that placing the system prompt last maximizes the likelihood of its inclusion in the computation of the result by the LLM. This approach not only makes the prompt more concrete but also facilitates easier comprehension for the operator. In this study, it is important to emphasize that the prompt is specifically crafted to respond either a 'true' or a 'false' response, and this intention has been repeatedly highlighted and reformulated, eventually simplifying the comparison across the 2,400 prompts and their respective baselines.

4.2 Computation

For the "Computation" phase of the framework, the information needed will be collected and will be evaluated for phishing specifically. All information, such as *steps_per_alarm*, *prompt_per_step_per_alarm*, and *baseline* and the implemented algorithms can be found on this GitHub [15]. This returns the following result, which will be interpreted in the next section.

4.3 Results

In this section, the results from the computation phase will be analyzed. The discussion will cover three different tables, showing the average time, average character length, and accuracy respectively, highlighting the most important information.

Table 4 shows the average time throughout the different LLMs. It is evident that Llama3 is consistently faster than Qwen:4b and Phi3. This suggests that Llama3 has the smallest cost, since it requires the least amount of computation time. Qwen on the other hand has half the amount of parameters, potentially making it quicker, but it is not. This time differentiation can be explained if it is correlated with Table 5.

During the "Preparation" phase, the LLM was programmed to respond with 'true' or 'false'—words comprising four and five characters respectively. Assuming an equal distribution of responses, the average character count per response should be 4.5, since the words 'true' and 'false' have an average character length of 4.5 characters. However, examining the average response lengths reveals that only Llama3 approximates this expected length, with an average of 5.5 characters per response. This short length likely contributes to Llama3's faster performance, suggesting it adheres more strictly to the instructions compared to Qwen and Phi3, thus needing less time to finish the response. Similarly, Phi3's decrease in time can be correlated with the decreasing character length in Phi3.

Phi3 is clearly the slowest from these three. This is predictable, since Phi3 has the highest parameter size. while the parameter size is not more than 2 times the parameter size of Llama3, it still takes a little less than 10 times as much time to compute. Additionally, during the implementation of the algorithm, the model was loaded only once, so that when running, it would not require to reload everything into the cache memory. This means that there is another reason, which is not cache, but perhaps Phi3's design. Phi3:14B has been designed to handle a wide variety of complex tasks [39], which can make its internal structure more complex and thus resulting in a higher time. In Table 4,

Step	Llama3 Time	Qwen:4b Time	Phi3 Time
0	1.964283	2.768391	28.043647
1	2.042337	2.531961	22.630253
2	2.143857	3.013252	14.742495
3	2.071740	2.738032	12.684208
Average	**2.055554**	**2.762909**	**19.525151**

Table 4. Average Time Measurements for Various Models

In table 5, it can be also seen that Llama3 is generally consistent between the token lengths, each being near 4.5 tokens, but in step 2, there is a clear outlier where it is almost double: 8.728. A potential explanation can be found in its relevant step. The security step asks the LLM to answer two questions, and Llama3 answers both of them, thus resulting in a longer response, answering true true instead of solely true.

Step	Llama3 Length	Qwen:4b Length	Phi3 Length
0	4.512	264.140	245.743
1	4.362	227.073	192.280
2	8.728	302.980	100.380
3	4.437	255.768	77.085
Average	**5.510**	**262.490**	**153.872**

Table 5. Average Character Lengths for Various Models

The accuracy (Acc) of various LLMs as detailed in Table 6 is examined, which presents accuracy measurements across multiple prompts for each model at different stages of interaction. It represents how similar the prompts of the LLM are compared to the human counter-part. This means that the message can still be accurate even if the character length varies from the expected format, which will increase accuracy. Please note that the prompt P1 used in step 0 differs from the prompt P1 used in step 1, as each step requires a unique scenario prompt.

It is evident that some prompts perform better for different LLMs. To obtain the best accuracy for each LLM, the best prompt is selected in each step for each LLM, and the average of these selections is computed. This is possible, because the prompts are different in each row.

Notably, Phi3 outperforms the other models, achieving the highest accuracy of 0.573. This suggests that Phi3 is more robust across diverse scenarios. Llama3, on the other hand, shows the lowest performance with an accuracy of 0.553 in one instance. Statistical

analysis would be useful here to determine the significance of these results given the high number of prompts (2400/LLM). Even marginal improvements in accuracy, when applied across such a high number of prompts, implies more beneficial value: Scores below 0.5 fail to exceed the benchmark set by random chance (e.g. a monkey flipping a coin), but 0.573 exceeds it significantly with N=2400. This suggests that LLM can be beneficial, but for this study, it might not be as effective as it can be.

When analyzing the average accuracy scores for each step, it is evident that performance varies between steps and models. Step 0, with an average score of 0.518, suggests that this step may not be effectively managed by LLMs. Conversely, Step 1 shows an improvement, with an average score of 0.543, indicating better performance by LLMs in assisting with this task. However, generalizations about the utility of LLMs for specific steps should be approached with caution. Notably, Qwen consistently outperforms others in Step 1, achieving accuracy scores higher than 0.575.

	Llama3 Acc			**Qwen:4b Acc**			**Phi3 Acc**			
Step	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**Avg**
0	0.440	0.440	<u>0.490</u>	<u>0.610</u>	0.575	0.575	<u>0.515</u>	0.500	0.515	**0.518**
1	<u>0.630</u>	0.575	0.600	0.490	<u>0.525</u>	0.490	0.535	<u>0.570</u>	0.475	**0.543**
2	0.500	0.505	<u>0.525</u>	<u>0.580</u>	0.575	0.520	0.480	0.530	<u>0.560</u>	**0.531**
3	<u>0.570</u>	0.460	0.470	0.495	<u>0.510</u>	0.495	0.600	<u>0.645</u>	0.520	**0.529**
Best	**0.553**			**0.556**			**0.573**			

Table 6. Accuracy Measurements for Various Models Across Multiple Prompts. The best column is the average of the accuracy of the best four cases for each LLM. These best cases are <u>underlined</u>.

In conclusion, this section highlights the advantages and disadvantages of applying LLM in the cyber security analysis process. Phi3, despite its large parameter size and longer computation time, showed the highest accuracy. This suggests that while Phi3 is computationally more expensive, it is more effective in complex decision-making tasks. On the other hand, Llama3 and Qwen showed similar performance in terms of computation time and accuracy. Notably, Llama3 adhered more closely to the formatting instructions, whereas Qwen and Phi3 often responded in sentence form, making it less predictable, but makes it easier to understand the choice of the LLM.

Specifically, in the context of phishing analysis, the tested LLMs show potential despite their relatively low accuracy. They demonstrate an ability to differentiate between phishing and benign emails with an accuracy slightly above 0.55, which, given the high sample size, suggests a non-random level of accuracy. This indicates that while LLMs may not fully replace human decision-making in security operations, this suggests that LLM can be beneficial, but in this study it has not reached it full potential.

The primary advantages of integrating LLMs include their ability to quickly interpret extensive textual content and identify underlying intentions, such as distinguishing between phishing and legitimate communications. However, limitations are evident in the granularity of the tasks. For instance, while an LLM might successfully identify a phishing attempt when analyzing the entire email content, its performance might degrade when tasked with making a determination based on less data, such as just the subject line.

Moreover, another limitation observed is the trade-off between computational time and confidence in the accuracy of the results. For example, Phi3 takes an average of 19.5 seconds per response, which might be impractical in real-time scenarios where rapid response is crucial.

Ultimately, while LLMs demonstrate promising capabilities in enhancing phishing detection, the application of these models must be carefully tailored to balance effectiveness, efficiency, and operational demands.

5 Conclusions

This thesis has explored the potential of LLMs to automate the decision-making processes in cybersecurity alarm analysis, aiming to mitigate 'alert fatigue' in security operations. This research highlighted the key steps in the analysis process and the state-of-the-art LLMs in May 2024. Next, the effectiveness of several LLMs (i.e. Llama3:8B, Qwen1.5:4B Phi3:14B) has been assessed, by developing a methodology to evaluate their performance in real-world cybersecurity scenarios, revealing that all LLMs demonstrated a non-random ability to match human responses. The study confirmed that while LLMs can significantly enhance the efficiency and accuracy of security alarm analyses, their effectiveness varies depending on the complexity of the tasks and the specific characteristics of each model.

6 Limitations

In preparing for future research, several critical considerations must be addressed, especially surrounding the limitations encountered in the current study. Firstly, the evaluation of LLMs on brute-force attack alarms was hindered by time and privacy constraints, preventing access to the necessary data due to legal restrictions. Furthermore, financial and computational limitations restricted the exploration of different LLMs. For instance, while GPT-4 emerged as a promising candidate, its utility was limited in analysing longer texts, such as two large emails, due to token length limitations. Similarly, larger models, such as Llama:70B, were too resource-intensive for standard personal computing devices. Using manually labeled data increases the risk of human errors in data reliability, and the current method of evaluating responses, which depends on human decision-making in unusual situations, is inconvenient.

7 Future Work

For future research, several enhancements and directions are suggested. First, more scalable or efficient computational strategies should be considered to accommodate the demands of larger LLMs, which may involve exploring alternative models or improving hardware capabilities. Additionally, while the current focus was on phishing attacks, a broader evaluation encompassing various techniques listed in the MITRE ATTA&K framework is recommended to provide a more comprehensive understanding of cybersecurity threats. Future studies should also utilise pre-labeled datasets to enhance the credibility of the results and consider leveraging LLMs to automate and refine the evaluation process. This could significantly improve the accuracy and efficiency of response assessments. Establishing GPT-4 or a similarly capable LLM as a baseline for benchmarks could standardise evaluations and provide a clearer metric for comparison across different studies.

Disclaimer on the Use of AI

AI Assistants, such as ChatGPT: Assisted with rewriting, formatting this document in LaTeX, including the creation of tables, and summarization of extensive data. Used a tool for difficult terminology to get a better understanding when the search engines are not sufficient.

After careful review, the author takes full responsibility for writing this document.

LLMs: Applied to evaluate and interpret complex cyber security data, demonstrating potential to automate decision-making processes and reduce human error.

References

[1] 2015. Jupyter Notebook. https://docs.jupyter.org/en/latest/
[2] 2018. Apache's Public Corpus. https://spamassassin.apache.org/old/publiccorpus/
[3] 2018. Papers with code - bert: pre-training of deep bidirectional transformers for language understanding. https://paperswithcode.com/paper/bert-pre-training-of-deep-bidirectional
[4] 2020. Python 3.9.0. https://www.python.org/downloads/release/python-390/
[5] 2022. Bigscience Bloom. https://huggingface.co/bigscience/bloom
[6] 2022. ImageNet. https://paperswithcode.com/sota/image-classification-on-imagenet
[7] 2022. Multi Task Language Understanding. https://paperswithcode.com/sota/multi-task-language-understanding-on-mmlu
[8] 2022. Pathways Language Model Paramters. https://research.google/blog/pathways-language-model-palm-scaling-to-540-billion-parameters-for-breakthrough-performance/
[9] 2023. . https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023
[10] 2023. Claude AI. https://claude.ai/
[11] 2023. *Data Breaches Report 2023.* IBM Corporation. https://www.ibm.com/downloads/cas/E3G5JMBP
[12] 2023. PaLm 2. https://ai.google/discover/palm2/
[13] 2023. unraveling the landscape of large language models: a systematic review and future perspectives. *Journal of Electronic Business & Digital Economics* 3, 1 (2023), 3–19. https://doi.org/10.1108//JEBDE
[14] 2023. What are Large Language Models (LLMs)? https://www.ibm.com/topics/large-language-models
[15] 2024. Beau GitHub. https://github.com/BCJonkhout
[16] 2024. ChatGPT. https://chatgpt.com/?oai-dm=1
[17] 2024. code for the paper evaluating large language models trained on code. https://github.com/openai/human-eval
[18] 2024. A collection of phishing samples for researchers and detection developers. https://github.com/rf-peixoto/phishing_pot/
[19] 2024. Cortex XSOAR: Security Orchestration and Automation. https://www.paloaltonetworks.com/cortex/cortex-xsoar
[20] 2024. Falcon LLM. https://falconllm.tii.ae/
[21] 2024. Google. https://www.google.com/
[22] 2024. ISO/IEC 27001:2022. https://www.iso.org/standard/27001
[23] 2024. Meta's Llama3. https://llama.meta.com/llama3/
[24] 2024. Microsoft Defender voor Office 365. https://www.microsoft.com/nl-nl/security/business/siem-and-xdr/microsoft-defender-office-365
[25] 2024. NIST CSF. https://www.nist.gov/cyberframework
[26] 2024. Northwave Cybersecurity. https://northwave-cybersecurity.com/
[27] 2024. Ollama. https://ollama.com/library
[28] 2024. OpenAI's Artificial Intelligence Deal Valuation. *The New York Times* (16 feb 2024). https://www.nytimes.com/2024/02/16/technology/openai-artificial-intelligence-deal-valuation.html
[29] 2024. Qwen. https://huggingface.co/Qwen
[30] 2024. Snort - Network Intrusion Detection & Prevention System. https://www.snort.org/
[31] 2024. Splunk. https://www.splunk.com/
[32] 2024. What is the Difference Between XDR vs. SIEM? https://www.paloaltonetworks.com/cyberpedia/what-is-xdr-vs-siem
[33] 2024. Zeek IDS. https://zeek.org/
[34] 1kg. 2024. Ollama : What is Ollama? - 1kg - Medium. https://medium.com/@1kg/ollama-what-is-ollama-9f73f3eafa8b
[35] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
[36] Ehsan Aghaei, Xi Niu, Waseem Shadid, and Ehab Al-Shaer. 2022. Securebert: A domain-specific language model for cybersecurity. In *International Conference on Security and Privacy in Communication Systems*. Springer, 39–56.
[37] Jeanine Banks and Tris Warkentin. 2024. Gemma: Introducing new state-of-the-art open models. *Google. Available online at: https://blog.google/technology/developers/gemma-open-models/(accessed 6 April, 2024)* (2024).
[38] Matthias Bastian. 2023. Gpt-4 has more than a trillion parameters-report. *The Decoder* 3 (2023).
[39] Misha Bilenko. 2024. Introducing Phi-3: Redefining what's possible with SLMs | Microsoft Azure Blog. https://azure.microsoft.com/en-us/blog/introducing-phi-3-redefining-whats-possible-with-slms/
[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[41] Mohamed Amine Ferrag, Mthandazo Ndhlovu, Norbert Tihanyi, Lucas C Cordeiro, Merouane Debbah, Thierry Lestable, and Narinderjit Singh Thandi. 2024. Revolutionizing Cyber Threat Detection with Large Language Models: A privacy-preserving BERT-based Lightweight Model for IoT/IIoT Devices. *IEEE Access* (2024).
[42] Intezer. 2022. SOC Analyst Training: How to Detect Phishing Emails. https://www.youtube.com/watch?v=Xrzsu-FFvu8
[43] Aikaterini-Lida Kalouli, Annebeth Buis, Livy Real, Martha Palmer, and Valeria De Paiva. 2019. Explaining simple natural language inference. In *Proceedings of the 13th Linguistic Annotation Workshop*. 132–143.
[44] Michail E. Klontzas, Salvatore Claudio Fanni, and Emanuele Neri. 2023. *Introduction to Artificial Intelligence*. https://doi.org/10.1007/978-3-031-25928-9
[45] George Kurtz and CrowdStrike. 2024. *CROWDSTRIKE 2024 GLOBAL THREAT REPORT*. https://go.crowdstrike.com/rs/281-OBQ-266/images/GlobalThreatReport2024.pdf
[46] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).
[47] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).
[48] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[49] LogRhythm. 2024. LogRhythm SIEM Security & SOC Services. https://logrhythm.com/
[50] mistralai. 2024. Official inference library for Mistral models. https://github.com/mistralai/mistral-inference
[51] Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, et al. 2023. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045* (2023).
[52] MITRE. 2024. MITRE ATT&CK. https://attack.mitre.org
[53] Mahdi Namazifar, Alexandros Papangelis, Gokhan Tur, and Dilek Hakkani-Tür. 2021. Language model is all you need: Natural language understanding as question answering. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 7803–7807.
[54] UCI NLP. 2024. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. https://huggingface.co/datasets/ucinlp/drop
[55] Ertugrul Portakal. 2023. Claude 2 Parameters (Parameter Size, Context Window...). https://textcortex.com/post/claude-2-parameters
[56] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).
[57] Gopinath Rebala, Ajay Ravi, Sanjay Churiwala, Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. 2019. Machine learning definition and basics. *An introduction to machine learning* (2019), 1–17.
[58] Help Net Security. 2021. Rapid increase in security tools causing alert fatigue and burn out - Help Net Security. https://www.helpnetsecurity.com/2021/03/22/security-tools-increase/
[59] siemxpert. 2023. Phishing Email Analysis. https://www.siemxpert.com/blog/phishing-email-analysis/
[60] Freya Thomson. 2022. Cybersecurity strategies: fighting alert fatigue and building resilience. https://www.openaccessgovernment.org/fighting-alert-fatigue-and-building-resilient-cybersecurity-strategies/139904/
[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[62] HanXiang Xu, ShenAo Wang, Ningke Li, Yanjie Zhao, Kai Chen, Kailong Wang, Yang Liu, Ting Yu, and HaoYu Wang. 2024. Large Language Models for Cyber Security: A Systematic Literature Review. *arXiv preprint arXiv:2405.04760* (2024).