ChatSEC: Spicing up Vulnerability Scans with AI for Heterogeneous University IT

Mario Hoffmann
IT Department, Leipzig University
Infrastructure Group
Leipzig, Germany
e-mail: mario.hoffmann@uni-leipzig.de

Erik Buchmann
Dept. of Computer Science, Leipzig University
Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, Germany
e-mail: erik.buchmann@uni-leipzig.de

Abstract—**With their heterogeneous and self-administrative structure, universities and comparable institutions differ from others in the industry and business in terms of enforcing IT security policies. This makes it challenging for the CIO (Chief Information Officer) and IT department to enforce common IT security rules. Through fast pacing positional changes within research groups, information on installed and maintained systems, as well as responsibilities can be lost. This has a negative impact on IT security. In this paper, we describe our ongoing work on ChatSEC, our approach to improve the reports generated by a vulnerability scan appliance. By using large language models and external threat intelligence, ChatSEC generates intuitive explanations how to assess and mitigate the reported vulnerabilities. Our preliminary evaluation indicates, that ChatSEC has much potential to improve IT security at universities and similarly heterogeneous institutions.**

***Keywords-AI; Heterogeneous Infrastructure; IT Security*.**

I. Introduction

The implementation of IT security at universities and similar institutions differs greatly from that in business and industry. Students need to set up their own servers to host their lab projects. Researchers change institutions without leaving instructions for security maintenance and the planned lifespan of the services they have set up. Each research group needs its very own, highly specific IT infrastructure. Some research communities are expected to transfer lab data via insecure plain-text protocols, such as the File Transfer Protocol (FTP). Due to the academic self-administration, the CIO and the central IT department have limited authority to enforce security rules. It is also not desirable to restrict the research groups and the educational programs by committing them to use only central IT services, that are secured by the IT department. On the other hand, the Local System Administrators (LSA) in the research groups do not necessarily have much knowledge in IT security. The rapid pace, at which research develops and research personnel changes position, means that responsibilities and information on installed systems quickly become outdated.

An internal vulnerability scan at the Leipzig University (May, 2023) detected 8311 vulnerabilities, 535 of them unique, on 3825 hosts, with scores from 2.1 (low) to 10.0 (critical) on the Common Vulnerability Scoring System version 3 (CVSSv3) metric [1] [2]. The scan detected 19 different operating systems (OS). Because the OS were installed in different versions, we observed a total of 39 different OS instances. Approximately 4,2% of the discovered vulnerabilities were due to missing OS patches or insecure OS configurations. Approximately 72% of the vulnerabilities were on hosts outside of the IT Department. Figure 1 shows a typical vulnerability scan report for one host.

We shared the scan reports with the responsible LSAs, and asked them to fix the vulnerabilities. We observed LSAs inheriting this responsibility from a predecessor, and had little expertise with the system. We also observed, that complex requests from the IT department were postponed in favor of undelayable teaching- and research assignments. Maintaining IT security requires to invest much more time, than just setting up a system as a demonstrator or for teaching purposes.

The concern of this work-in-progress paper is to close this gap between the IT department and the LSAs in the research groups. We propose ChatSEC, our approach to tailor the results of a vulnerability scan with a large language model (LLM) for specific target groups in a highly heterogeneous IT environment. Thus, ChatSEC has a different focus than approaches such as Microsoft's Security Copilot [3] or SecBot [4]. In particular, we make three contributions:

- We describe ChatSEC, our approach to utilize AI to prepare and extend vulnerability scan reports for LSAs with limited IT security knowledge.
- We discuss implementation alternatives to generate intuitive explanations from domain-specific reports, and to add threat intelligence and specific mitigation strategies.
- We provide a preliminary evaluation of our approach, based on the aforementioned vulnerability scan.

Our preliminary evaluation indicates, that ChatSEC has the potential to greatly improve the IT security at universities and similarly structured, heterogeneous organizations. We plan to integrate our approach into our next vulnerability scan, which will provide us with a data set of scan reports to improve and user feedback for a qualitative study on the LSA's perception of ChatSEC. For security considerations, we plan to test LLMs that can be hosted as a service by the IT department.

Paper structure: Section II presents related work. Section III describes our ChatSEC concept, which is briefly evaluated in Section IV. Section V concludes the paper.

II. Related Work

This section summarizes approaches comparable to ours, LLMs, NLP approaches and threat intelligence.

Vulnerability Scan Report

Scan Time: Wed, Jan 1, 2024 1:00 AM - Wed, Jan 1, 2024 1:09 AM, Hosts scanned: 1

Results (1/1)

Vulnerability	*Severity*	*Host*	*Location*
Ubuntu: Security Advisory (USN-5767-1)	9.8 (High)	012.345.67.89	package

Summary
The remote host is missing an update for the 'python2.7, python3.6, python3.8, python3.10' package(s).

Vulnerability Detection Result
Vulnerable package: libpython3.8
Installed version: libpython3.8-3.8.10-0ubuntu1 20.04.5
Fixed version: >=libpython3.8-3.8.10-0ubuntu1 20.04.6

Solution
Solution Type: Vendorfix. Please install the updated package(s).

Affected Software/OS
'python2.7, python3.6, python3.8, python3.10' package(s) on Ubuntu 18.04, Ubuntu 20.04, Ubuntu 22.04, Ubuntu 22.10.

Vulnerability Insight
Nicky Mouha discovered that Python incorrectly handled certain SHA-3 internals. An attacker could possibly use this issue to cause a crash or execute arbitrary code. (CVE-2022-37454) Python incorrectly handled certain IDNA inputs. An attacker could possibly use this issue to expose sensitive information, denial of service, or cause a crash. (CVE-2022-45061)

References
CVE: CVE-2022-37454 CVE-2022-45061 WID-SEC-2023-1007 WID-SEC-2023-0561 WID-SEC-2023-0255 WID-SEC-2023-0138 WID-SEC-2022-2043 WID-SEC-2022-1816 DFN-CERT-2023-1109 DFN-CERT-2023-0886 DFN-CERT-2023-0580 DFN-CERT-2023-0571 DFN-CERT-2023-0552 DFN-CERT-2023-0429 DFN-CERT-2023-0422 DFN-CERT-2023-0120 ($\cdots$)
Other: https://ubuntu.com/security/notices/USN-5767-1 advisory_id:USN-5767-1

Hosts 1 of 1

IP-Address	*Hostname*	*OS*	*High*	*Medium*	*Low*	*Total*	*Severity*
012.345.67.89	ourhost.ourdomain.tld	Canonical Ubuntu Linux	1	0	0	1	9.8 (High)

Figure 1. Typical vulnerability scan report

A. Comparable Approaches

We aim for compensating the lack of IT security expertise by using LLMs, with a specific focus on heterogeneous university IT. Existing approaches have a different focus: Microsoft's Security Copilot [3] helps IT security teams of large companies to process security-related data, and contributes to a security strategy. SecBot [4] is a questionnaire-based chatbot for IT security, that helps end-users with security planning and mitigation strategies. ChatIDS [5] rewrites IDS-generated threat alarms in an intuitive way to help end-users securing a smart-home scenario. Comparable approaches in other fields exist, e.g., to interpret research papers in chemistry [6].

A recent survey [7] also shows, that current chatbots can provide IT-security knowledge to help with IT security related tasks, based on domain specific datasets. This needs a high adaptability, because the models are used in unknown contexts. Using fine-tuned, domain-specific LLMs [8] with domain-specific embeddings can increase the precision.

B. Large Language Models

LLMs [9] [10] refer to pre-trained models, based on large amounts of texts and data, which utilize statistical distribution of tokens to obtain a generative ability. They differ from their predecessors Pre-trained Language Model (PLM) in model and data sizes. LLMs are the first models, that show *emergent abilities* [11], such as *multi-step reasoning* or *instruction following* to solve complex tasks. Those abilities are significant for currently used models, such as Llama-3 [12], Claude 3 [13] Gemini [14], or GPT-4 [15].

LLMs are instructed by textual **prompts** in natural language. Well-engineered prompts improve and bias the generated output [16] [17] [18]. Thus, the prompts are used to utilize the emergent abilities. **Prompt-engineering** strategies [19], [20] include Chain-of-Thought prompting [21] (asking the LLM step by step), Reflection [22] (asking the LLM to rethink his answer), Few-Shot prompting [23] (giving examples) or Repetition [18] (repeating relevant aspects in the prompt). Over-generalization is a common issue with LLMs [18]. Slight variations of the prompt can have a big impact on the model output [24]. If reproducible outputs are needed, the seed and other model parameters can be fixed [25].

C. NLP approaches

To evaluate the LLM output, we also use approaches from natural language processing (NLP).

Readability measures assess the level of readability of texts by categorizing texts into school grades or scoring systems. The most common readability measures are Flesch-Kincaid-Grade-Level (FKG) [26] for English texts, and Wiener-Sachtextformel IV (WSF) [27] for German texts. Table I shows a mapping between the metrics: FKG results in school grades, ranging from 0 to 18, according to college years of the U.S. school system. The WSF algorithm refers to German school

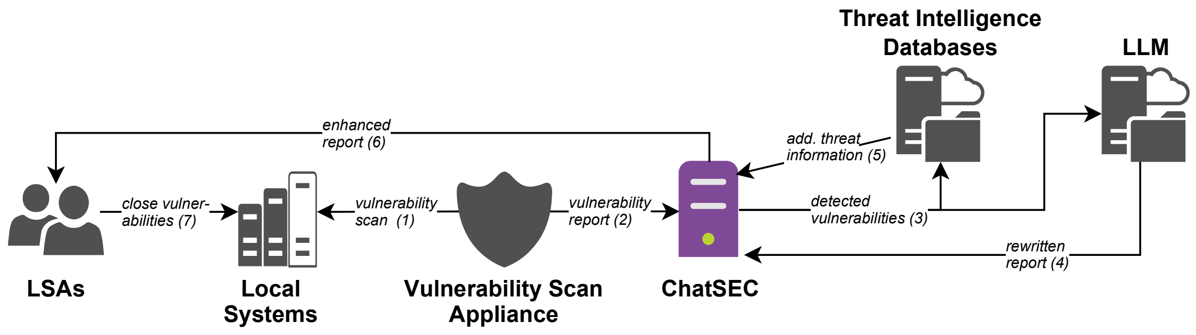
Threat Intelligence Databases
LLM
enhanced report (6)
add. threat information (5)
close vulner-abilities (7)
vulnerability scan (1)
vulnerability report (2)
detected vulnerabilities (3)
rewritten report (4)
LSAs
Local Systems
Vulnerability Scan Appliance
ChatSEC

Figure 2. ChatSEC architecture and data flow.

TABLE I
FKG & WSF READABILITY METRICS

Readability	FKG	WSF
very hard	17-18	13-15
hard	13-16	12
rather hard	10-12	11
medium	8-9	9-10
rather simple	7	7-8
simple	6	6
very simple	5	4-5

grades, and ranges from 4th to 15th grade. The lower the result, the more readable is the assessed text.

Stemming reduces a word to its root [28] to normalize texts. For example, the root form of the words ”change”, ”changing” and ”changes” would be ”chang”. This unifies derivations of a word by removing suffixes. Different languages need different stemming algorithms [29] [30]. Popular stemmers are:

- PortStemmer (Python port, [31]; English language)
- Snowball (PortStemmer v.2, [32]; multilingual)
- Cistem (based on [33]; German language)

D. Threat Intelligence

Threat Intelligence [34] is indispensable for threat mitigation and -prevention. There are three major options to gather threat intelligence data, build IT security related contexts between all obtained data, and exchange this information:

The **Common Vulnerabilities and Exposures** (CVE) program is a list of publicly known vulnerabilities [35]. A single vulnerability is identified by IDs in the format ”CVE-2024-1234”, with ”2024” as the year of occurrence and ”1234” as a consecutive number. The list offers several options to further describe the vulnerabilities, such as *references*, *affected versions* or *cross-references* to other data sources. A common source for CVEs is the National Vulnerability Database (NVD) [36]. The **Common Weakness Enumeration** (CWE) list provides a *root cause mapping* [37]. It correlates with the CVE ids and identifies the underlying root cause of a vulnerability. Thus, a CVE registered vulnerability is an *instance* of one or more CWE-described weaknesses. There are several description levels ranging from abstract to very detailed. The **Searchsploit** framework by ExploitDB [38] is often used by penetration testers and security researchers. It provides a *list of available exploits* for different software, operating systems or CVE ids.

III. Our ChatSEC Concept

We aim to implement an IT security approach in a research institution, which depends on plenty of heterogeneous IT systems. Those systems must be managed locally in the research groups, and cannot be unified under the umbrella of the IT department. There are two disctinct roles involved:

The **Local System Administrator (LSA)** has domain knowledge of the IT services needed of its research group, and is *responsible for specific IT components*, that are not part of the central IT infrastructure of the university.

The **IT Department (ITD)** is responsible for the university's general IT infrastructure, and also for its perimeter security. This includes *periodic vulnerability scans*, and *supporting the LSAs* in securing their systems. The ITD possesses expert knowledge on IT security, but has limited human resources.

We propose ChatSEC to rewrite vulnerability scan reports for LSAs. The rewritten reports must be easy to read, intuitively understandable and directly applicable for an LSA. Needlessy complex, convoluted sentences full of security-specific technical concepts should be avoided, and references to mitigation strategies must to go beyond ”Browse the vendor homepage for a security patch that solves your issue”.

A. ChatSEC Architecture and Workflow

The architecture of ChatSEC is shown in Figure 2. ChatSEC obtains scan reports, as illustrated in Figure 1, from a vulnerability scan appliance. ChatSEC then rewrites these reports with the help of a LLM. Finally, the rewritten report is sent to the LSAs, who are responsible for closing vulnerabilities. The rewriting process consists of three stages:

1) Generate an **intuitive explanation**
2) Add **threat intelligence** information
3) Add **mitigation strategies**

In the following, we explain our concept for these stages, and we explain how the output of the stages can be evaluated.

This vulnerability description is about a security advisory for Ubuntu operating system. The advisory states that the remote host (a computer connected to a network) is missing an update for the 'python2.7, python3.6, python3.8, python3.10' packages. The update is announced through the USN-5767-1 advisory. The first vulnerability, identified as CVE-2022-37454, is related to Python's handling of certain SHA-3 internals. This vulnerability could be exploited by an attacker to cause a crash or execute arbitrary code on the affected system. The second vulnerability, identified as CVE-2022-45061, is related to Python's handling of certain IDNA inputs. An attacker could potentially exploit this vulnerability to expose sensitive information, perform a denial of service attack, or cause a crash on the affected system. In summary, these vulnerabilities in Python could allow attackers to crash the system, execute arbitrary code, expose sensitive information, or cause a denial of service attack. It is important to update the affected packages to protect the system from these potential risks.

Figure 3. Transformed scan report from Figure 1

B. Intuitive Explanations

As Figure 1 illustrates, vulnerability scan reports are often written and formatted in a very technical, less intuitive way. Therefore, LSAs need help to understand them linguistically and conceptually. This stage of ChatSEC generates intuitive texts and examples from the scan results by utilizing a LLM. In particular, we generate prompts, that instruct the LLM to summarize each scan result in an intuitive way. This helps LSAs to understand the actual problem reported, without the need to deep dive into IT security.

C. Threat Intelligence

The severity score of a vulnerability indicates its threat potential. However, it is challenging for an LSA to find out how serious similarly scored vulnerabilities could affect a specific system. This stage enriches the output from the first stage with threat intelligence data (e.g., the number of active exploits) without decreasing the readability of the texts. Therefore, we fetch vulnerability-related information from multiple threat intelligence data sources.

D. Mitigation Strategies

This stage helps the LSA to mitigate the vulnerabilities from the scan report. If the vulnerability can be closed with an update, ChatSEC generates intuitive instructions how to obtain and install updates, based on software package, OS version information, etc. from the scan report. However, we observed that 95.8% of the vulnerabilities detected by our scan report refer to configuration problems. In this case, ChatSEC either obtains mitigation information from the NVD [36] with a tag from the report. Alternatively, individually created ChatSEC requests can be used.

E. Evaluation Options

We see three options to evaluate ChatSEC: To find out, if ChatSEC's output is intuitively understandable, we can use **NLP techniques**, such as readability metrics and stemming: The fewer domain-specific words are used, the better for non-domain-specific readers. The evaluation of the correctness of the generated output needs a **manual assessment** by an expert. User experiments allow to obtain **direct feedback** from LSAs through questionnaires. **Indirect feedback** can be obtained by repeating the vulnerability scan, some time after ChatSEC has explained the results of the first scan to the LSAs: If ChatSEC's reports were indeed understandable and helpful, we should see a vast decrease in the number of vulnerabilities.

IV. Evaluation

To obtain evidence of how promising our concept is, we implemented the two stages *Intuitive Explanations* and *Threat Intelligence* into a research prototype of ChatSEC, and we evaluated it with *NLP techniques* and a *manual assessment*.

A. Implementation and Evaluation Setup

Our focus is the feasibility and applicability of ChatSEC on our internal vulnerability scan, as described in Sec. I. Thus, we left aside aspects of prompt engineering, model selection and model tuning (cf. Sec. II). We managed the vulnerability scan results with Greenbone [39], and implemented our ChatSEC prototype using the GPT-3.5 Turbo API of ChatGPT [15].

We used Chain-of-Thought prompting [21], i.e., we executed Figure 4 and 5 in a sequence, which we found most promising in preliminary tests: The first prompt produces a simplified report. The second prompt adds examples to that report. $< \cdots >$ denotes the position where the names, descriptions and details from the vulnerabilities managed with Greenbone are inserted. Because our LSAs use German and English, with the command "Answer in German." we instructed the LLM to also generate German output, and we evaluated both versions. To obtain focused answers, we set the system prompt parameter "temperature" to 0.2 [40]. To provide an intuitive example, Figure 3 shows ChatSEC's output with the vulnerability scan report from Figure 1 and the prompts from Figure 4 and 5.

You will be provided with a vulnerability description. [Answer in German.] Help in the following order: Summarize the provided vulnerability description and explain it to a non-technician. <Vulnerability information>

Figure 4. System prompt to simplify vulnerability scan reports

You will be provided with a vulnerability description. [Answer in German.] Give a simple example to show what can happen if the provided vulnerability is exploited to a non-technician. <Vulnerability summary from Figure 4>

Figure 5. System prompt, that adds examples to the simplified report

To add threat intelligence, ChatSEC fetches the CWEs and CVEs associated with each vulnerability scan report from the CWE list [37] and the NVD CVE database [36]. ChatSEC also queries Searchsploit [38] for the number of available exploits per vulnerability. ChatSEC uses the prompts shown in Figure 6 and 7 to translate this domain-specific information into an intuitive text, that can be added to the translated report and sent to an LSA. The number of known exploits can be appended directly, without an extra prompt.

You will be provided with a list of properties of an IT security severity score. The list is formed as a key-value list, where the values are on the right side of the colon. [Answer in German.]. Assume that you answer to a non-technician. Help in the following order: Answer in sentences. Explain each list item and assume that "low" or "None" is highly critical.
<list of severity properties>

Figure 6. System prompt to generate a severity explanation

You will be provided with an IT weakness. Assume that you answer to a non-technician. Help in the following order: Explain the weakness enumeration in simple terms. Only return: Summarize your own explanation for non-technicians. Do not provide mitigation advices.
<CWE description>

Figure 7. System prompt, that explains a CWE

B. Intuitive Explanations

We evaluate the understandability and readability of ChatSECs output first. Therefore, we let ChatSEC generate two outputs for each of the 535 unique vulnerabilities from our internal security scan, both in English and German.

To measure how much our ChatSEC implementation relies on a **vocabulary from the security domain**, we computed the average number of words for the scan report and the generated texts first. The orginal scan reports are much shorter than the texts produced by ChatSEC (see Table II). We stemmed both the generated texts and the National Institute of Standards and Technology's (NIST) glossary [41], and counted the matches. The last three columns of Table II show the matches per stemmer. Consider Column 3: PortStemmer found, that 14% of the 68 words from the original security scan could be found in the NIST glossary, and 15.5% of ChatSEC's Englisch output of 345 words. Thus, our prompt did not let the AI restrict the use of domain-specific vocabulary. A "n.a." refers to a stemmer, that is not applicable to English or German texts.

TABLE II
AVERAGE NUMBER OF WORDS AND DOMAIN-SPECIFIC VOCABULARY

	Avg. num. of words	**Port-Stemmer**	**Cistem**	**Snowball**
Original report	68	14%	n.a.	7%
English output	345	15.5%	n.a.	15.5%
German output	275	n.a	10%	10%

We measure the **readability** with FKG for English texts and WSF for German texts. ChatSEC's English output was evaluated with an average FKG of 12, ranging from 8.7 to 16.5. Thus, on average the English texts are rated as "rather hard" or "hard" to read, with exceptions spanning from "medium" to "very hard" (cf. Table I). The WSF assesses the German output with an average score of 7 ("rather simple"), ranging from 4.3 to 10.6, i.e., from "very simple" to "rather hard". Thus, the generated German texts were slightly easier to read than the English ones.

C. Threat Intelligence

While understandability and readability can be evaluated automatically, an evaluation of the correctness, completeness and applicability of threat intelligence added to a security scan report requires an expert assessment. For the time being, we therefore limit ourselves to a proof of feasibility. We declare success, if ChatSEC is able to enrich its intuitive output with the correct number of exploits, correct additional CVSSv3 data and the right root causes, without significantly decreasing the readability. For evaluation, we have chosen CVE ID 2019-0708, which appeared in our vulnerability scans.

With this CVE ID, ChatSEC queries various external data sources: Searchsploit [38] tells ChatSEC, that this ID is associated with four known exploits. By querying the CWE list [37], ChatSEC learns that the root cause of the vulnerability is CWE-416, named as "Use After Free" and described as "Improper Control of a Resource Through its Lifetime". From the NVD CVE database [36], ChatSEC learns that this vulnerability has the CVSSv3 score 9.8 (critical), and is associated with the following vulnerability information:

- "attackComplexity":"low"
- "privilegeRequired":"None"
- "userInteraction":"None"

Figure 8 shows an example for the resulting threat intelligence, that is translated with the prompts from Figure 6 and 6. This text is then added to ChatSECs translated vulnerability report and sent to the LSAs.

The severity score indicates, that the attack complexity is low, meaning that it is relatively easy for an attacker to exploit the security vulnerability. Additionally, no privilege is required for the attack, meaning that the attacker does not need any special access rights or permissions to carry out the attack. Furthermore, there is no user interaction required, suggesting that the attack can be automated without user input.
The referenced cause CWE-416, or "Use After Free," is a computer security vulnerability where a program tries to use memory that has already been freed up. This can cause errors or allow hackers to exploit the system.
Searchsploit found 4 currently available exploits for this vulnerability.

Figure 8. Intuitive threat intelligence for CVE ID 2019-0708

We found, that the ChatSECs generated output was correct both in English and German. The English output had a FKG of 13.5 (hard), the German output had a WSF of 4 (simple). Thus, adding threat intelligence information did not decrease the readability of ChatSECs output, and the German version is still easier to read.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

V. Conclusion

The heterogeneous IT ecosystem and the self-administrative organizational structure makes it challenging to implement IT security in universities. In this paper, we outlined our ongoing work on ChatSEC, our approach to rewrite vulnerability scan reports for local system administrators with limited IT security knowledge. We focused specifically on intuitive explanations, additional threat intelligence and mitigation strategies, that are applicable by our target auditory without having to browse external sources. Our evaluation provided evidence, that ChatSEC, with the help of an LLM, indeed produces reports that are helpful to assess and close detected vulnerabilities.

As part of our future work, we plan to conduct extensive user experiments in combination with the next internal security scan, to obtain direct feedback on ChatSEC. Based on this feedback, we will improve the LLM prompts and readability scores. We will also include further sources for threat intelligence, and fully implement the integration of mitigation strategies into ChatSEC's reports. We will also test open source LLMs that can be installed locally, to avoid that information on detected vulnerabilities must leave the premises. Eventually, we plan to integrate ChatSEC into a Security-as-a-Service tool, that can be used on demand.

References

[1] Forum of Incident Response and Security Teams (First), "Common Vulnerability Scoring System (CVSS-SIG)," https://www.first.org/cvss, retrieved: Aug. 2024.

[2] National Institute of Standards and Technology (NIST), "Vulnerability Metrics (CVSS)," https://nvd.nist.gov/vuln-metrics/cvss, retrieved: Aug. 2024.

[3] Microsoft Cooperation, "Microsoft Security CoPilot," https://www.microsoft.com/de-de/security/business/ai-machine-learning/microsoft-security-copilot, retrieved: Aug. 2024.

[4] M. F. Franco *et al.*, "SecBot: A business-driven conversational agent for cybersecurity planning and management," in *2020 16th international conference on network and service management (CNSM)*. IEEE, 2020, pp. 1–7.

[5] V. Jüttner, M. Grimmer, and E. Buchmann, "ChatIDS: Explainable cybersecurity using generative AI," *arXiv preprint arXiv:2306.14504*, 2023.

[6] K. G. Yager, "Domain-specific chatbots for science using embeddings," *Digital Discovery*, vol. 2, no. 6, pp. 1850–1861, 2023.

[7] S. Shafee, A. Bessani, and P. M. Ferreira, "Evaluation of LLM Chatbots for OSINT-based Cyberthreat Awareness," *arXiv preprint arXiv:2401.15127*, 2024.

[8] S. Pal, M. Bhattacharya, S.-S. Lee, and C. Chakraborty, "A domain-specific next-generation large language model (LLM) or ChatGPT is required for biomedical engineering and research," *Annals of Biomedical Engineering*, vol. 52, no. 3, pp. 451–454, 2024.

[9] M. Shanahan, "Talking about large language models," *Communications of the ACM*, vol. 67, no. 2, pp. 68–79, 2024.

[10] W. X. Zhao *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[11] J. Wei *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[12] Meta, "Llama 3," 2024, retrieved: Aug. 2024. [Online]. Available: https://llama.meta.com/llama3/

[13] Anthropic, "The Claude 3 model family: Opus, Sonnet, Haiku," https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf, 2024, retrieved: Aug. 2024.

[14] Gemini Team Google, "Gemini: A family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[15] OpenAI, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[16] S. Arora *et al.*, "Ask me anything: A simple strategy for prompting language models," in *Proceedings of the 11th International Conference on Learning Representations*, 2022.

[17] C. Si *et al.*, "Prompting GPT-3 to be reliable," *arXiv preprint arXiv:2210.09150*, 2022.

[18] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–21.

[19] J. W. Rae *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446*, 2021.

[20] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 199–22 213, 2022.

[21] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[22] G. Kim, P. Baldi, and S. McAleer, "Language models can solve computer tasks," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36, 2023, pp. 39 648–39 677.

[23] T. Brown *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[24] T. Hagendorff, "Machine psychology: Investigating emergent capabilities and behavior in large language models using psychological methods," *arXiv preprint arXiv:2303.13988*, 2023.

[25] E. Lee, "Control OpenAI model behavior with seed: Step-by-step with code," https://drlee.io/control-openai-model-behavior-with-seed-step-by-step-with-code-9bba4e137a63, 01 2024, retrieved: Aug. 2024.

[26] J. P. Kincaid and et al., "Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel," Research Branch Report, Tech. Rep., 1975.

[27] R. Bamberger and E. Vanecek, *Lesen - Verstehen - Lernen - Schreiben*. Diesterweg, 1984.

[28] D. Khyani, B. Siddhartha, N. Niveditha, and B. Divya, "An interpretation of lemmatization and stemming in natural language processing," *Journal of University of Shanghai for Science and Technology*, vol. 22, no. 10, pp. 350–357, 2021.

[29] D. Yogish, T. Manjunath, and R. S. Hegadi, "Review on natural language processing trends and techniques using NLTK," in *Proceedings of the 2nd International Conference on Recent Trends in Image Processing and Pattern Recognition*. Springer, 2019, pp. 589–606.

[30] A. G. Jivani *et al.*, "A comparative study of stemming algorithms," *International Journal of Computer Applications in Technology*, vol. 2, no. 6, pp. 1930–1938, 2011.

[31] C. J. Van Rijsbergen, S. E. Robertson, and M. F. Porter, *New models in probabilistic information retrieval*. British Library Research and Development Department London, 1980.

[32] M. Porter, "Snowball," https://snowballstem.org/, retrieved: Aug. 2024.

[33] L. Weissweiler and A. Fraser, "Developing a stemmer for German based on a comparative analysis of publicly available stemmers," in *Language Technologies for the Challenges of the Digital Age: 27th International Conference, GSCL 2017, Berlin, Germany, September 13-14, 2017, Proceedings 27*. Springer, 2018, pp. 81–94.

[34] R. McMillan, "Definition: Threat intelligence," https://www.gartner.com/en/documents/2487216, 2013, retrieved: Aug. 2024.

[35] MITRE Corporation, "Common vulnerabilities and exposures," https://cve.mitre.org/, retrieved: Aug. 2024.

[36] National Institute of Standards and Technology (NIST), "National Vulnerability Database (NVD)," https://nvd.nist.gov/, retrieved: Aug. 2024.

[37] MITRE Corporation, "Common weakness enumeration," https://cwe.mitre.org/, retrieved: Aug. 2024.

[38] OffSec, "ExploitDB - SearchSploit," https://www.exploit-db.com/searchsploit, retrieved: Aug. 2024.

[39] Greenbone, "Vulnerability Management," 2024, retrieved: Aug. 2024. [Online]. Available: https://www.greenbone.net/

[40] OpenAI, "OpenAI Platform API Reference," https://platform.openai.com/docs/api-reference/chat/create, retrieved: Aug. 2024.

[41] National Institute of Standards and Technology (NIST), "Cybersecurity Basics - Glossary," https://www.nist.gov/itl/smallbusinesscyber/cybersecurity-basics/glossary, retrieved: Aug. 2024.