CRUSH: Cybersecurity Research using Universal LLMs and Semantic Hypernetworks

Mohit Sewak*,†, Vamsi Emani† and Annam Naresh†

Microsoft Security Research (MSecR), Microsoft R&D, India.

Abstract

Enterprise Knowledge Graphs (EKG) are powerful tools for representing and reasoning about complex and dynamic domains, such as cyber threat intelligence. However, designing and constructing such graphs can be challenging, especially when dealing with heterogeneous and noisy data sources. This paper presents our novel approach to using Large Language Models (LLM) for EKG design and development based on our experience building a Threat Intelligence Graph (TIG) using GPT3.5/ GPT4/ ChatGPT. We show how LLMs can automatically extract, infer, validate, and summarize information from various sources, such as threat reports, literature, scripts, etc., and populate the EKG with relevant entities, relationships, and properties. We also demonstrate how LLMs can identify malicious intents in any script file and map it to the TIG to detect any malicious techniques linked to the script. We demonstrate that an LLM-EKG-based approach could deliver up to 99% recall on the task of detection of malicious scripts and a consistent 90%+ recall on the task of detection of specific threat types across all script-based cybersecurity threats.

Keywords

Large Language Models, Enterprise Knowledge Graph, Threat Hunting, Endpoint Protection, GPT

1. Introduction

Cyber threat intelligence is the process of collecting, analyzing, and disseminating information about current and emerging cyber threats, such as actors, targets, techniques, tools, etc. Cyber threat intelligence can help organizations to proactively defend themselves from cyberattacks by providing them with actionable and timely insights. However, cyber threat intelligence is also a complex and dynamic domain, where the information is often scattered across various sources, such as news articles, social media posts, dark web forums, etc. It can be incomplete, inconsistent, or inaccurate. In addition, cyber threats are constantly evolving and adapting to new technologies and environments, making it difficult to keep track of them.

Malicious scripts are common vectors of cyber threats and attacks, as they can execute arbitrary code on the target system and compromise security. Detecting malicious scripts is, therefore, an important task for enterprise security. However, detecting malicious scripts is difficult, especially while preserving the privacy and intellectual property rights underlying the code in these scripts. Moreover, malicious scripts can be dynamically generated or modified by other scripts or programs, making them hard to analyze statically. Therefore, traditional signature-based or rule-based methods are ineffective in detecting malicious scripts as they can only match known files or near absolute pattern-matching.

One way to address these challenges is to use dynamically designed Enterprise Knowledge Graphs (EKG), which are graph-based representations of knowledge that capture the semantics and context of a domain. EKGs can enable efficient and effective reasoning and query over large-scale and heterogeneous data sources by leveraging the rich structure and semantics of the graph. EKGs can also facilitate knowledge discovery and sharing among stakeholders by providing a common and intuitive view of the domain. Graphs on executables, API calls, and control flows have been used in the past [1]. The graphs used in such approaches had been designed, extracted, and enriched using the expertise of subject matter experts (SME) and manually designed extractors. Hence, these are neither scalable nor widely extendable to applications, even across the cybersecurity domain. Designing and creating dynamic and scalable EKGs without extensive SME involvement can be challenging, especially when dealing with complex and dynamic domains like cyber threat intelligence. Some of the challenges include:

- How to define the schema and ontology of the graph that can capture the essential concepts and relationships of the domain?
- How to extract relevant information from various sources and map them to the graph schema?
- How to validate and improve the quality and consistency of the information in the graph?

CIKM'23: Workshop on Enterprise Knowledge Graphs Using Large Language Models, 22nd October 2023, University of Birmingham, UK

*Corresponding author.

†These authors contributed equally.

mohit.sewak@microsoft.com (M. Sewak);
venkata.emani@microsoft.com (V. Emani);
annaresh@microsoft.com (A. Naresh)
https://www.linkedin.com/in/mohitsewak/ (M. Sewak)
0000-0001-8375-5713 (M. Sewak)

CEUR Workshop Proceedings (CEUR-WS.org)

- How to summarize and present the information in the threat reports and malicious scripts in graph format and vice versa?
- How to make decisions based on information in the graph?
- How to correlate insights across multiple graphs or subgraphs?

In this paper, we present our novel approach, CRUSH (for **C**ybersecurity **R**esearch using **U**niversal LLMs and **S**emantic **H**ypernetworks), to using Large Language Models (LLM) to alleviate these challenges associated with designing, creating and using such dynamic and scalable EKG in an automated manner, without much SME involvement and in real-time. Using this approach, we design and enrich a Threat Intelligence Graph (TIG) using the GPT series of LLMs, like the GPT 3.5 (Turbo), GPT 3.5 Chat, and GPT 4 Chat, and Bing Chat (An enhanced/assisted variant of GPT 4). LLMs are large foundation neural network models trained on large-scale text corpora and can generate natural language texts based on various inputs or prompts. LLMs have shown remarkable capabilities in various natural language processing tasks, such as text summarization, question answering, text generation, etc.

However, LLMs are also not perfect. Mitigating hallucination and enhancing the ability to reason [2] are looming research challenges for any practical enterprise application of LLMs. Moreover, to the best of our information, LLMs have not previously been employed for such complex tasks of designing and enriching EKGs. Therefore, this paper shows how we leveraged these capabilities to address the challenges of designing EKGs and creating them for cyber threat intelligence. We baseline CRUSH, an LLM-EKG-based approach, to the LLM-Only-based approach for malicious script and threat type detection. We demonstrated that CRUSH is superior to an LLM-only approach but can also deliver up to 99% recall on detecting malicious scripts and a consistent 90%+ recall on detecting specific threat types across all script-based cybersecurity threats.

2. Related Work

LLMs [3] have revolutionized various applications and domains. These have revolutionized multiple domains ranging from biomedical [4, 5] to education [6]. However, their application in the cybersecurity domain has found limited applications. This is because not many applications in cybersecurity use text and natural language data and telemetry. One approach to leveraging the architecture of the large language models is to use transformers and other architectural aspects behind the LLMs and pre-train the resulting model on security data like the malware binaries. A similar approach has been used in the past for the security domain [7]; however, such approaches did not become as popular.

Open Source Intelligence (OSINT) refers to the process and methods of finding, gathering, analyzing, and using information from publicly available sources [8]. OSINT has a direct connection with cybersecurity. The cyber intelligence community has increasingly started to use public records to collect unique and valuable intelligence. This involves creating a complete profile of certain targets using openly available information [9]. However, despite the opportunities, due to the unstructured and unbound nature of the information available in OSINT, it is not easy to use OSINT [10, 11]. In the case of cybersecurity, similar challenges exist. Hence, using OSINT is still largely a human expert and curated knowledge retrieval task.

The terminology used in cybersecurity is unique, and hence, researchers have found it valuable to train (relatively) smaller versions of the transformer models on cybersecurity data [12]. Given the added complexity in the threat intelligence (TI) domain, researchers have found it useful to train such transformer models on TI data [13]. However, these relatively smaller foundation models suffer from multiple drawbacks that led to the evolution of LLMs [14], and hence are not ideal for the type of applications we intend to perform.

3. Data, LLMs and LLM-Assisting resources Used

We compare and baseline CRUSH against an LLM-Only approach as described in section 4 across two tasks. Both tasks use a common dataset, which is a collection of malicious and clean script files across different programming languages and platforms. Each malicious file is mapped to a particular threat type, which has a specific TTP. Besides the script files, we collected blog URLs describing the various threats and their respective TTPs to which these scripts are mapped.

Each threat type of interest here could use a different programming language or script type based on its TTP. Some threat types could use multiple script languages to attack machines across different platforms or to evade detection. The expertise of Microsoft's Threat Research team was used to validate each threat type and label these scripts for the identified threat type. We also collected some clean scripts across different script types to avoid bias. Figure 1 shows the distribution of collected scripts across threat types. Whereas Figure 2 shows the distribution of these scripts based on scripting language or the extension.

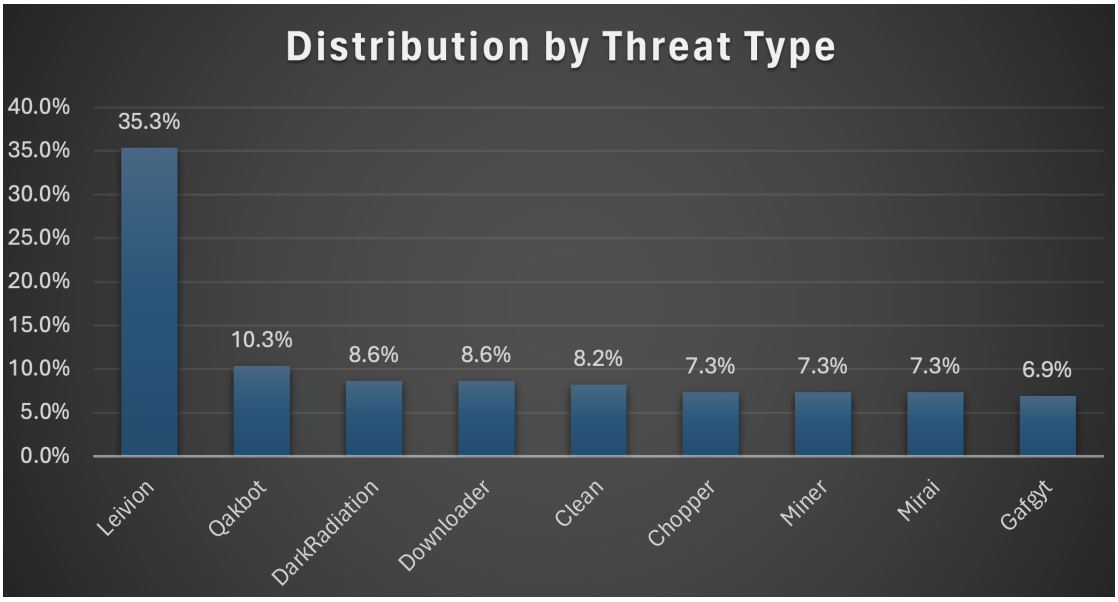

Figure 1: Distribution of Scripts based on Threat Type (Including Clean Scripts)

3.1. LLMs Used

We used multiple LLM Endpoints or APIs to demonstrate the universality of the application and the approach. Some of these endpoints are enhanced with either search or Retrieval Augmented Generation (RAG) [15] capabilities or both. The LLMs used are:

- GPT 3.5 Turbo
- GPT 3.5 Chat (ChatGPT)
- GPT 4.0 Chat (ChatGPT Plus)
- Bing Chat - A search and RAG enhanced variant of GPT-4

3.2. Addressing LLM Issues - Noise, Hallucination, Reasoning, and Planning

At appropriate stages in the methodology (refer section 5, we use LLMs to standardize the TTPs to standardized threat nomenclature as provided by third parties like MITRE ATT&CK[16] to reduce noise in the EKG and for a fair comparison. Some of the LLMs endpoints in the list (example LLM 3.1) come augmented with RAG, search, and planning. So, these variants of LLMs could automatically reason when to use search, RAG, or generation and could also create a plan of action to use these steps in a coordinated manner for the desired action [17]. For universal interoperability of the solution across all the stated LLMs, we enhanced the remaining with the MITRE ATT&CK TTP listing by either direct injection in the LLM context or by using RAG. We further augmented some LLM variants with reasoning and planning assist frameworks like the Microsoft Semantic Kernel [18].

4. Baseline: Classifying Malicious Scripts with LLM-Only approach

Large LLMs, like GPT-4, are often trained on a corpus of data that includes code and script files across different programming languages. Many applications and commercial tools employ such LLMs as co-pilots for coding [19]. This validates that these models could understand programming languages and produce syntactically correct code for a specific intent in a given programming or scripting language. Therefore, arguably, such LLMs possess the capability of understanding the intent of an existing code or script file. Therefore, to produce a baseline to validate our model against, we create an LLM-Only baseline model that combines best practices and techniques ranging from prompt engineering, reasoning, and planning [20, 21] to get the optimal performance out of an LLM directly. This approach uses two or more series of prompting with reasoning. In the first series of prompting, LLMs are used to summarize and then understand the intent of a particular script file. Due to the context length limitations, not all script files could fit into the LLM context. In such a case, we chunk the script into sizes that could fit into the LLM context, with adequate

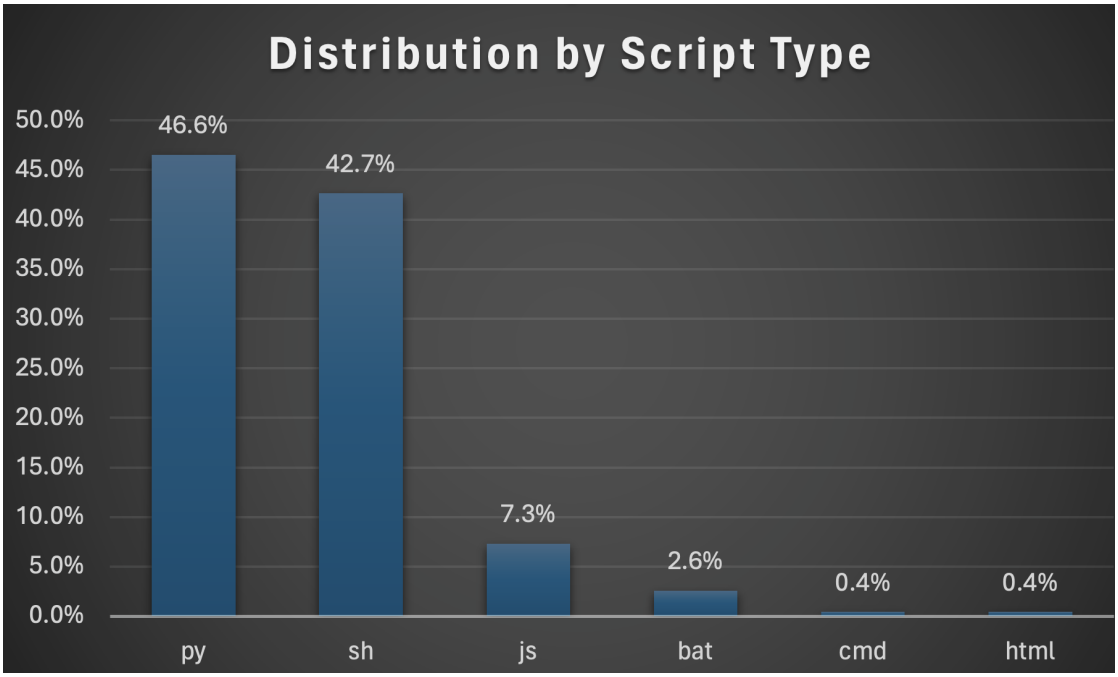

Figure 2: Distribution of Script based on Scripting Language

overlaps across chunks to maintain continuity. In a subsequent step, the obtained intent is again passed through another round of prompts to assess if the resultant code intent could be part of malicious Tactics, Techniques and Procedures (TTP). For the chunked scripts, another series of prompting was carried out to join the intents obtained from each chunk in sequence into a coherent intent description for homogeneity and further processing. In the next series of prompting, the script is classified as malicious or clean based on the response to these prompts. In yet another series of prompting, we tried to obtain the specific threat type from a limited list of threats) that the specific TTP could lead to.

4.1. Baseline Results

We baselined the performance of the Prompt Engineering based on the description in section 4. Figure 3. shows the binary model's baseline performance based on the TTP's maliciousness assessment. Figure 4 shows the class-wise recall of the multivariate model that assesses the treat type based on the TTP.

As shown in Figure 3, the precision of this model is not good. Also, Figure 4 shows that recall for certain threat types is not good.

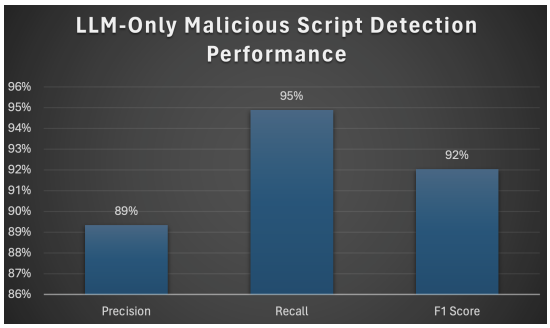

Figure 3: LLM-Only Baseline Performance - - Malicious Script Detection

5. CRUSH Methodology

Figure 5 shows a high-level schematic of our methodology. In this, we first use prompt engineering methodologies [22] to obtain a schema of the different node and relationship types relevant to the TI landscape. Examples of some of the node types that the LLM suggested to keep in the schema range from Malware Family, Malware Name, Threat Group, Techniques, etc.

Subsequently, in the next step, we provided some threat feeds and queried the LLM with another series of prompt engineering to extract the relevant nodes and

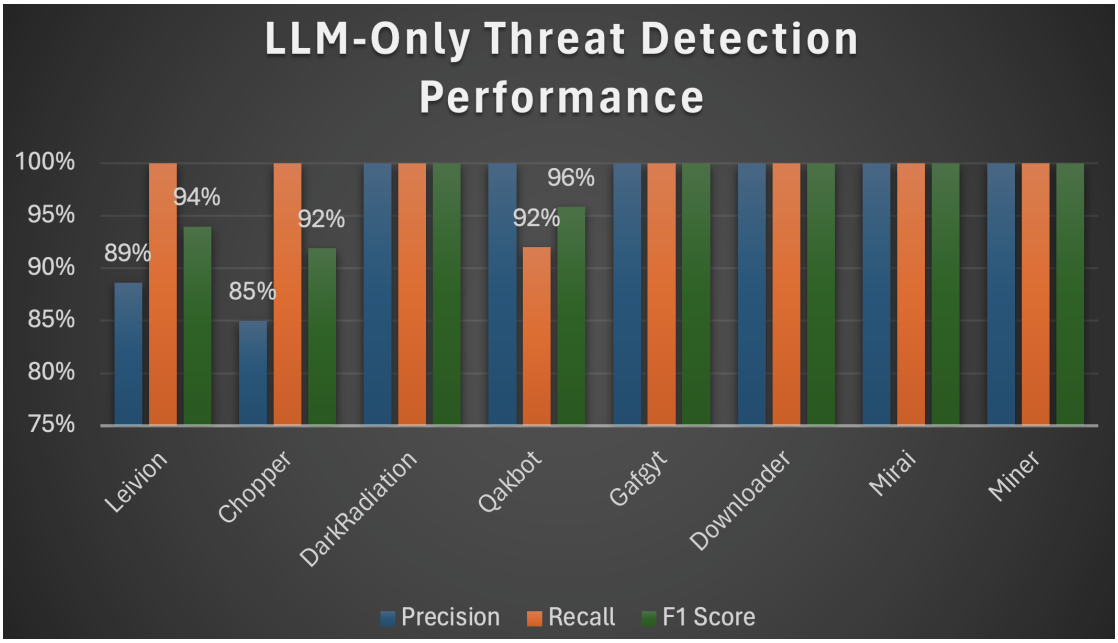

Figure 4: LLM-Only Baseline Performance: Threat Type Detection

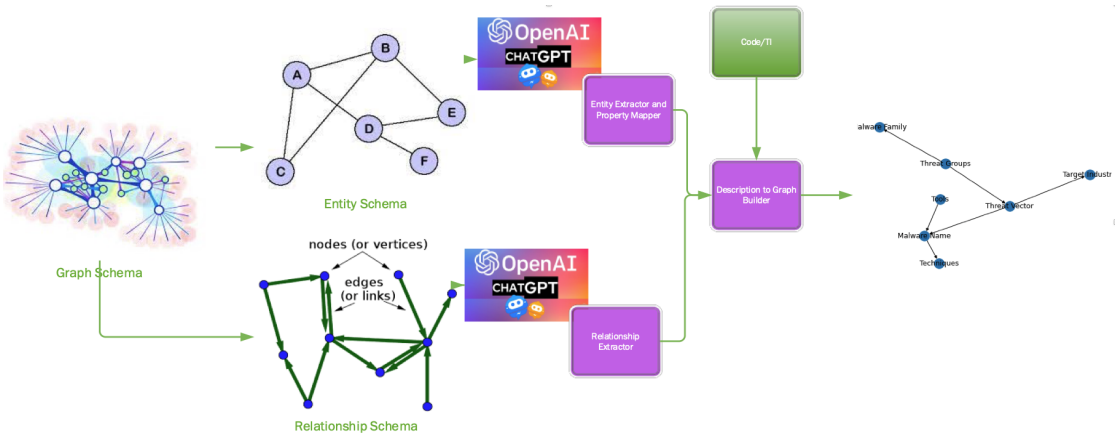

Figure 5: High-Level Process Schematic

relationships in the form of triples [23] (i.e. node-relation-node). For noise considerations, instead of querying over all entities, this process is repeated over individual relations in the schema. Further, longer feeds were broken into smaller parts to reduce noise and optimally fit into the context window.

The triples extraction process was transformed into multiple LLM-skill/plugins for the LLM to make this process generalizable and scalable [18]. Many of these extractions may require nested LLM plugins or skills. Also, while working on a large and scalable graph like this, it may become challenging to recall the appropriate llm-plugins. Therefore, to alleviate these challenges and enhance interoperability across multiple LLM endpoints, we used an LLM planner [24].

The sub-graph generated by the CRUSH for each script was intersected with the global TI graph made using the TTP description of all the combined threat types and their descriptions. The resulting feature vectors were used for classification using a logistic regression classifier, and

the results obtained were compared against the LLM-Only approach. The qualitative assessment and labelling are described in sub-section 5.1, and the quantitative performance benchmarks are provided in section 6.

5.1. Evaluation Mechanism

Since no standardized training or evaluation data exists for this purpose, we relied on the domain expertise of professional threat intelligence researchers to validate the constituents of the graph, and then later, the insights obtained from the traversal of this graph for different threat types and other cybersecurity purpose. For this purpose, we collaborated with Microsoft's Threat Intelligence Research team, which researches OSINT and other threat intelligence means and mechanisms. We acquired a collection of labelled scripts across different threat types from this team. Further, we used the SMEs from the team to validate the graph connection and entities suggested by the CRUSH approach for their relevance to the specific threat type and the specific TTP as discovered in the intent in the script code.

Following the qualitative assessment by the threat experts, we evaluated the two approaches empirically using the labels provided by the threat experts.

6. LLM-EKG Threat Detection Results

Using the methodology as described in section 5, and the labels collected as described in section 5.1, the binary and threat-class classification performance metrics for CRUSH (LLM-EKG) approach were computed. Figure 6 shows the performance in a binary detection scenario, where we are interested in knowing whether a script is malicious or clean. This result shows a significant uplift in the recall, taking it to 99% as compared to a baseline of 95%, of the LLM-Only model while maintaining or improving the precision and F1 scores over the baseline model.

Similarly, Figure 7 shows the performance metrics of the CRUSH-based model in detecting the specific threat type. For almost all threats but one, there is an improvement or maintenance in all the metrics (precision, recall, F1-score). Also, for most of the threats, the recall is 100% for the CRUSH approach.

7. Conclusion

We presented CRUSH, an approach to use LLMs and semantic hypernetworks to design and enrich a TIG for cyber threat intelligence. We showed how CRUSH can extract, integrate, and infer information from various cyber threat intelligence sources and how it can represent and query the complex and dynamic relationships in the TIG. We evaluated CRUSH on malicious script and threat type detection, showing that it can achieve high recall and outperform the LLM-only approach. We used multiple LLM endpoints or APIs to demonstrate the universality of our approach and how we addressed some of the common issues associated with LLMs, such as noise, hallucination, reasoning, and planning. We used LLMs to standardize the TTPs to standardized threat nomenclature provided by third parties like MITRE ATT&CK. Some of the LLM endpoints we used were enhanced with either search or RAG capabilities or both. These variants of LLMs could automatically reason and plan for the desired action. For universal interoperability of our solution across all the LLMs, we enhanced some LLM variants with reasoning and planning assist frameworks like Microsoft Semantic Kernel.

We believe that CRUSH is a promising and innovative approach to using LLMs for cyber threat intelligence and that it can be extended to other domains and tasks that require designing and enriching EKGs. We also hope that CRUSH can inspire more research on combining LLMs and semantic hypernetworks for natural language processing and artificial intelligence.

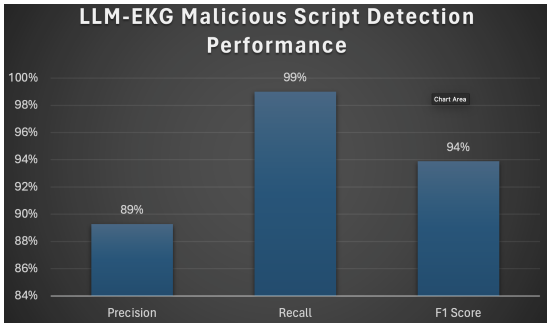

Figure 6: CRUSH (LLM-EKG) Performance - Malicious Script Detection

References

[1] T. Alsmadi, N. Alqudah, A survey on malware detection techniques, in: 2021 International Conference on Information Technology (ICIT), IEEE, 2021, pp. 371–376.

[2] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, et al., A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity, arXiv preprint arXiv:2302.04023 (2023).

[3] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al., A

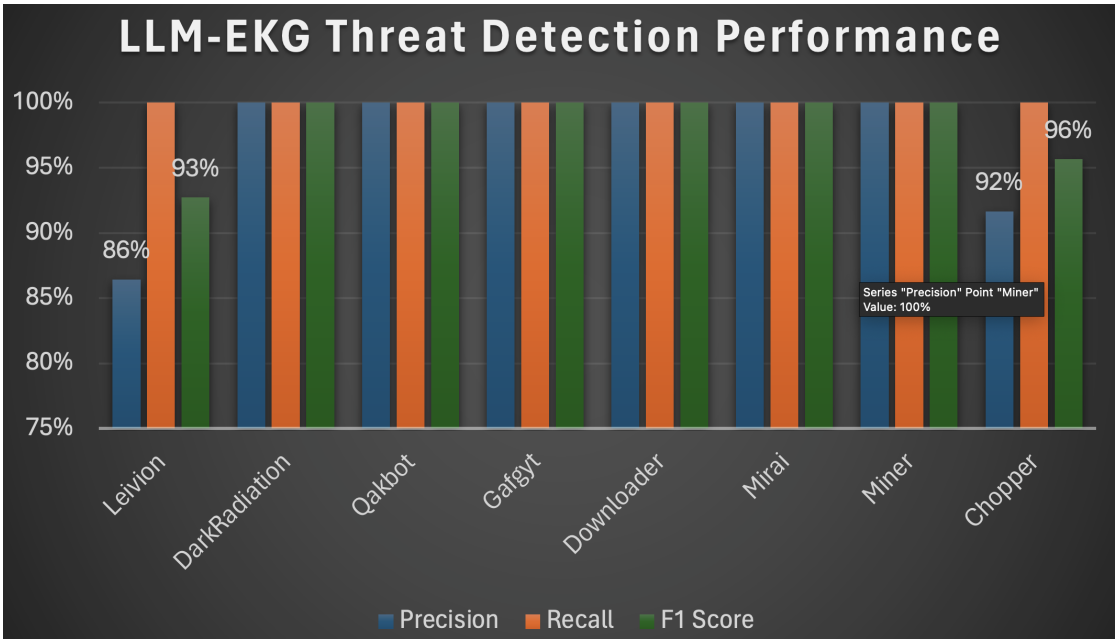

Figure 7: CRUSH (LLM-EKG) Performance: Threat Type Detection

survey of large language models, arXiv preprint arXiv:2303.18223 (2023).

[4] A. Lecler, L. Duron, P. Soyer, Revolutionizing radiology with gpt-based models: Current applications, future possibilities and limitations of chatgpt, Diagnostic and Interventional Imaging 104 (2023) 269–274.

[5] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, D. S. W. Ting, Large language models in medicine, Nature medicine (2023) 1–11.

[6] E. Kasneci, K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, et al., Chatgpt for good? on opportunities and challenges of large language models for education, Learning and individual differences 103 (2023) 102274.

[7] A. Rahali, M. A. Akhloufi, Malbertv2: Code aware bert-based model for malware identification, Big Data and Cognitive Computing 7 (2023) 60.

[8] J. R. G. Evangelista, R. J. Sassi, M. Romero, D. Napolitano, Systematic literature review to investigate the application of open source intelligence (osint) with artificial intelligence, Journal of Applied Security Research 16 (2021) 345–369.

[9] F. Tabatabaei, D. Wells, Osint in the context of cyber-security, Open Source Intelligence Investigation: From Strategy to Implementation (2016) 213–231.

[10] J. Pastor-Galindo, P. Nespoli, F. G. Mármol, G. M. Pérez, The not yet exploited goldmine of osint: Opportunities, open challenges and future trends, IEEE Access 8 (2020) 10282–10304.

[11] G. Hribar, I. Podbregar, T. Ivanuša, Osint: a "grey zone"?, International Journal of Intelligence and CounterIntelligence 27 (2014) 529–549.

[12] K. Ameri, M. Hempel, H. Sharif, J. Lopez Jr., K. Perumalla, Cybert: Cybersecurity claim classification by fine-tuning the bert language model, Journal of Cybersecurity and Privacy 1 (2021) 615–637. URL: https://www.mdpi.com/2624-800X/1/4/31. doi:10.3390/jcp1040031.

[13] M. Campobasso, L. Allodi, Threat/crawl: a trainable, highly-reusable, and extensible automated method and tool to crawl criminal underground forums, in: 2022 APWG Symposium on Electronic Crime Research (eCrime), IEEE, 2022, pp. 1–13.

[14] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, et al., A comprehensive survey on pretrained foundation models: A history from bert to chatgpt, arXiv preprint arXiv:2302.09419 (2023).

[15] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in Neural Information Processing Systems 33 (2020) 9459–9474.

[16] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nick-

els, A. G. Pennington, C. B. Thomas, Mitre att&ck: Design and philosophy, in: Technical report, The MITRE Corporation, 2018.
[17] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, React: Synergizing reasoning and acting in language models, arXiv preprint arXiv:2210.03629 (2022).
[18] Microsoft, Semantic kernel, https://github.com/microsoft/semantic-kernel, 2013.
[19] N. Nguyen, S. Nadi, An empirical evaluation of github copilot's code suggestions, in: Proceedings of the 19th International Conference on Mining Software Repositories, 2022, pp. 1–5.
[20] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al., Inner monologue: Embodied reasoning through planning with language models, arXiv preprint arXiv:2207.05608 (2022).
[21] Z. Zhang, A. Zhang, M. Li, A. Smola, Automatic chain of thought prompting in large language models, arXiv preprint arXiv:2210.03493 (2022).
[22] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, J. Ba, Large language models are human-level prompt engineers, arXiv preprint arXiv:2211.01910 (2022).
[23] R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, P. Colpaert, Triple pattern fragments: a low-cost knowledge graph interface for the web, Journal of Web Semantics 37 (2016) 184–206.
[24] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, Y. Su, Llm-planner: Few-shot grounded planning for embodied agents with large language models, arXiv preprint arXiv:2212.04088 (2022).