2022 International Conference on Electrical Engineering and Informatics (ICELTICs) | 978-1-6654-8766-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICELTICs56128.2022.9932097

Mapping Linux Shell Commands to MITRE ATT&CK using NLP-Based Approach

Yevonnael Andrew
Information Technology Department
Swiss German University
Tangerang, Banten, Indonesia
yevonnael.andrew@student.sgu.ac.id

Charles Lim
Information Technology Department
Swiss German University
Tangerang, Banten, Indonesia
charles.lim@sgu.ac.id

Eka Budiarto
Information Technology Department
Swiss German University
Tangerang, Banten, Indonesia
eka.budiarto@sgu.ac.id

***Abstract*—Honeypot is a decoy computer resource used to trap an attacker and one of the most common honeypots is a medium to high interaction honeypot that is able to log shell interaction executed by an attacker. In the cybersecurity field, these collected commands can be mapped to MITRE ATT&CK, a knowledge base, and model for cyber adversary behavior. To maximize the utilization of ATT&CK, a good mapping between Linux commands and ATT&CK is desirable. In this paper, we evaluate and measure the performance of Linux commands mapping to ATT&CK Techniques and Sub-Techniques, and ATT&CK Tactics using NLP techniques, such as Bag of Words, TF-IDF, and pre-trained Word Embeddings. Cosine similarity scoring is used to extract the top-n ATT&CK Techniques and Sub-Techniques, and ATT&CK Tactics for each command. The models' performance is evaluated using *recall at n* metrics.**

***Index Terms*—cybersecurity, honeypot, MITRE ATT&CK, machine learning, NLP, word embeddings**

I. Introduction

Honeypot is defined as a decoy computer resource in which the value arises from being probed, attacked, or compromised [1]. One popular honeypot available on the market is a Cowrie [2], a medium to high interaction SSH/Telnet honeypot, which is able to log shell interaction, i.e., commands executed by the attacker. By analyzing honeypot logs, researchers are able to learn malicious attackers' activities, including possibly malicious insiders, to understand their intent.

MITRE ATT&CK [3] is a curated knowledge base and model for cyber adversary behavior and taxonomy across their lifecycle. ATT&CK for Enterprise is a version of ATT&CK that covers behavior against enterprise IT networks and clouds. ATT&CK comprises Tactics, Techniques, and Procedures (TTP): Tactics represent the reason for the attack; Techniques specify how an adversary achieves its goal; Procedures describe the specific implementation of techniques or sub-techniques used during the attack.

To achieve a particular goal, the attackers may use various shell commands inside the system they are to compromise. Attackers usually perform several activities, such as profiling the hardware and software, downloading and executing the payloads, changing permissions, and even removing their footprints before they end the execution [4]. Hence, an accurate mapping of these Linux bash commands to MITRE ATT&CK TTP [5] can precisely describe the attacker's behaviors during its presence in the compromised systems.

In this paper, we evaluate Natural Language Processing (NLP) capability to support our research in mapping Linux bash command to its corresponding MITRE ATT&CK Tactics and Techniques, particularly one-to-many mapping based on text similarity performance. A collection of Linux bash commands and their relevant descriptions and MITRE ATT&CK descriptions are gathered, several pre-processing techniques to remove stopwords and symbols are performed, and the resulting sentences are projected into vectors using various methods: Bag of Words (BoW), Term Frequency–Inverse Document Frequency (TF-IDF), and Word Embeddings. Finally, the similarity scoring between Linux bash command to ATT&CK Techniques and Tactics descriptions is calculated. With the help of domain expert knowledge, the unsupervised model can be properly evaluated.

The remainder of this paper is organized as follows: In section II, various related research on shell command processing and NLP-based approaches are presented. The next section III discusses the research framework of our work. Section IV and V present the experiment results and discussions on the results respectively. Finally, section VI provides the concluding remarks of our research.

II. Related Works

Boffa et al. [6] treat shell commands like natural language, parse the shell commands, options, and parameters, and project them into Bag of Words and Word2Vec. Trizna et al. [7] extracts features from raw shell commands and use them as input for machine learning models. These research results look promising; however, they do not consider the meaning of that particular syntax.

Hussain et al. [8] take the command description from the documentation and provide scoring on the similarity of the command using the NLP approach. However, this research only measures the similarity between commands, not an external framework like MITRE ATT&CK. Kanakogi et al. [9] map automatically CAPEC from CVE and measure similarity using TF-IDF, USE, and SBERT. Shahid et al. [10] explore the textual description of vulnerability disclosure mapped to CVSS using NLP BERT. For the explainability aspect, the top relevant words are often in agreement with the rationales of a human expert.

Ayoade et al. [11] evaluated several models to classify threat reports generated from various computer security organizations into standardized tactics and techniques, including the relevant mitigation advisories. The authors used multiple datasets to train and evaluate the dataset, that is ATT&CK dataset, Symantec dataset, and APTReport dataset.

Karuna et al. [12] use MALMO and genetic programming for optimization to map threat descriptions, which are expressed in DSL, to the top-n TTP techniques. However, the authors do not evaluate the performance of these techniques in comparison to a certain benchmark, e.g., human expert. Ampel et al. [13] use pre-trained NLP RoBERTa and supervised machine learning to label CVE with one of ten ATT&CK tactics. Unfortunately, the authors do not evaluate its performance in predicting ATT&CK techniques and subtechniques. Kuppa et al. [14] use multiple encoders and joint embeddings with MLP to map CVE to ATT&CK techniques. The research has limitations in representing the entire MITRE ATT&K techniques corpus since the authors use merely 37 ATT&CK techniques in their research.

III. Research Framework

Figure 1 shows our research framework in our experiments. The process begins with some pre-processing steps of the shell command descriptions and MITRE ATT&CK descriptions to produce a clean description, which is then converted to a vector representation. Cosine similarity will be used to measure their similarity, which will be used to produce the top-n most similar results. The process was repeated for different types of vectorizers and tokenizers to compare their performance across models.

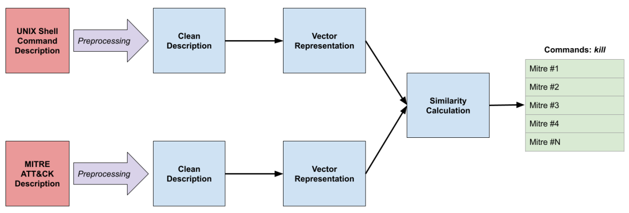

Fig. 1. Research Framework

A. Dataset

The dataset for this research consists of two different data: Linux commands description obtained from Linux online manual [15], and MITRE ATT&CK Techniques and Sub-Techniques description [16].

B. Linux Commands Description

To the best of our knowledge, no official and standardized Linux commands description has been published. Thus, the authors found the Linux man-pages project [15], which provides a comprehensive UNIX commands description. The project provides a manual, divided into multiple sections as follows:

- Section 1: User commands
- Section 2: System calls
- Section 3: Library functions
- Section 4: Devices
- Section 5: Files
- Section 7: Overviews, conventions, and miscellaneous
- Section 8: Superuser and system administration commands

Section 1 (user commands and tools) and 8 (administration and privileged commands) are the most relevant and suitable for our research. Each of the sections contains a command page that contain multiple sections, including:

- Name
- Synopsis
- Description
- Invocation
- Environment
- Options
- Return Values
- Usage Examples
- Author
- Copyright
- Colophon

Table I shows an example of command *wget* from the project.

TABLE I
Excerpt of Command 'wget' from the Online Manual (Truncated)

Section	**Content**
Name	wget - The non-interactive network downloader
Synopsis	wget [*option*]... [*URL*]...
Description	GNU wget is a free utility for non-interactive download of files from the Web. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies.
Options	Since wget uses GNU getopt to process command-line arguments, every option has a long form along with the short one. Long options are more convenient to remember, but take time to type. You may freely mix different option styles, or specify options after the command-line arguments. Thus you may write: *wget -r –tries=10 http://fly.srk.fer.hr/ -o log*. The space between the option accepting an argument and the argument may be omitted. Instead of -o log you can write -olog.
Environment	Wget supports proxies for both HTTP and FTP retrievals. The standard way to specify proxy location, which Wget recognizes, is using the following environment variables: http_proxy https_proxy. If set, the http_proxy and https_proxy variables should contain the URLs of the proxies for HTTP and HTTPS connections respectively.

This research uses only the command name and its description. The name section of a command can be short and very concise, i.e., *locale - get locale-specific information*, or it can be longer as follow: *abicompat - check ABI compatibility, abicompat checks that an application that links against a given shared library is still ABI compatible with a subsequent version of that library. If the new version of the library* *introduces an ABI incompatibility, then abicompat hints the user at what exactly that incompatibility is.*

In this example, *abicompat*, does not contain a description. So, by this rationale, a command that does not have a section description may probably already be included in the name section. Thus, if a command does not have a description section, its name will be used instead.

Python with BeautifulSoup is used to provide automatic scraping of text inside the HTML pages, and a total of 2,629 different commands have been collected. Some commands may have two pages containing the same command's manual: one page is from POSIX Programmer's Manual, and the other one is from GNU Development Tools. In this case, the latter is chosen and used for further processing. Otherwise, the POSIX version will be used instead. After removing duplicates, the final dataset consists of 2,536 different commands.

C. MITRE ATT&CK Techniques and Sub-Techniques Description

Official MITRE GitHub repository [16] provides complete MITRE ATT&CK Techniques description in JSON format, including MITRE ATT&CK Techniques and Sub-Techniques. Conversion is performed from JSON to CSV or Pandas DataFrame, and only ATT&CK descriptions that contain Linux as their platform will be extracted. Table II shows the sample data of the dataset collected.

TABLE II
SAMPLE DATA FOR MITRE ATT&CK DESCRIPTION

Column Name	Content
name	Junk Data
id	T1001.001
platforms	Linux, macOS, Windows
kill chain phases	Command and Control
description	Adversaries may add junk data to protocols used for command and control to make detection more difficult. By adding random or meaningless data to the protocols used for command and control, adversaries can prevent trivial methods for decoding, deciphering, or otherwise analyzing the traffic. Examples may include appending/prepending data with junk characters or writing junk characters between significant characters.

D. Data Pre-processing and Modeling

To prepare data for further processing by machine learning algorithms, textual data needs to be converted into a vector representation. The most common representations used in NLP are Bag of Words (BoW), TF-IDF, and Word2Vec. The next process will compare these different representations to their performances.

The bag of Words (BoW) model is one of the classical text representations in which only the occurrence frequency of words, disregarding word order dan context, are counted. The Bag of N-Grams model, an extension to the BoW model, breaks text into chunks called n-gram(s). If we have a sentence as follow: "Andrew goes to the university", BoW model will store each word individually, whereas the 2-gram (bigram) model store as follows: {"Andrew goe~~,~~ "goes to", "to the", "the university"}. By using n-gram, it captures some spatial information and some context. However, as n increases, the dimensionality and sparsity of the matrix increase rapidly [17].

Term Frequency – Inverse Document Frequency (TF-IDF) is a statistical metric to determine the importance of a given word relative to other words in the document and in the corpus. It combined two mathematical measurements: TF measures how often a word occurs in a document, and IDF measures the importance of a word across a corpus [17]. A high TF-IDF score indicates a high-frequency word in a particular document but has a low frequency in the whole corpus. Thus, TF-IDF is useful for filtering out common words.

Word embedding is a representation of words that can derive the meaning of the word from its context [17]. Mikolov et al. [18], [19] developed Word2Vec, in which the metric depends on distributional similarity based on the word analogy relationships. Word2Vec uses two different models to learn the embedding: Continuous Bag-of-Words (CBOW) model, which learns by predicting current words by contexts, and the Continuous Skip-Gram model, which learns by predicting surrounding words by current words. In this paper, pre-trained embedding models will be utilized.

Pre-processing efforts depend on the model to be used, and the following are the process for each model:

- For BoW and TF-IDF, non-ASCII characters, punctuation, digits, and stopwords will be removed, and all texts will be converted into lowercase texts.
- For Word Embedding models, all texts will be converted into lowercase texts.

To measure the performance of the texts being compared, i.e., the text similarity, cosine similarity defined in [17] is used. The formula for the cosine similarity is shown in Equation 1.

$$\cos(\theta) = \frac{\mathbf{A}\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \quad (1)$$

A high cosine value indicates that the two vectors are closely related.

E. Metrics and Evaluation

Several metrics could be used in measuring the performance of classification or clustering in machine learning, including accuracy, precision, and recall. In this research, recall is used to measure how accurately the model in identifying the relevant data. Once the Linux commands are mapped to relevant MITRE ATT&CK Techniques, which are collected from the official MITRE website, an unsupervised model is then used to cluster these mappings, and the recall values are calculated to measure its performance. It is important to note that one Linux command may be mapped to several ATT&CK Techniques. For instance, *curl* can be used in various techniques, such as Ingress Tool Transfer (T1105), Lateral Tool Transfer (T1570), Exfiltration Over Alternative Protocol (T1048), and other techniques as well.

The mapping on the official MITRE website is done manually by experts and contributors, and the process is ongoing; as new threats come in, a new mapping will be generated. In this research, the desire is to have a model that can predict as many ATT&CK Techniques and Sub-Techniques intersecting with the benchmark dataset. It is noteworthy that a low recall means the predicted ATT&CK Techniques and Sub-Techniques are not on the benchmark list, which does not imply directly that the prediction is wrong. It is possible that the techniques are not listed on the official website, or they may be related. For example, *vnc* and *ssh* are not the same but are related.

To evaluate the models, we use *recall at n*, which can be done by extracting top-n the most similar to ATT&CK Techniques and Sub-Techniques for each command and count the number of predictions that intersect with the benchmark dataset, divided by the length of benchmark dataset, for each specific commands. Equation 2 illustrates the recall formula for our research.

$$Recall - n = \frac{Predicted \cap Benchmark}{Benchmark} \tag{2}$$

The final score will be in the range of 0 to 1, with 0 being the lowest possible score and indicating a bad model. This top-n recall metric can compare the performance of our similarity scoring model.

IV. Experimental Results

Bag of Words and Bag of N-Grams are implemented by using the CountVectorizer from scikit-learn [20]. For CountVectorizer and TF-IDF tokenization, we used Unigram and Unigram + Bigrams. To achieve the root forms of inflected words, we used Snowball Stemmer for stemming, and WordNet Lemmatizer for lemmatization.

Table III shows the recall-n score of mapping to MITRE ATT&CK techniques and sub-Techniques of each model. Table IV shows the recall-n score of mapping to MITRE ATT&CK Tactics. Table III and IV also show that the model performance increases as the number (n) increases. A more detailed performance can be seen in Figure 2, which shows the recall scores of one of the models for multiple top-n. Due to the nature of how a recall metric works, the higher n in the top-n similarity, the recall will continue to increase.

Table V depicts the performance of the multiple pre-trained Word Embeddings models. The scores are worse than using TF-IDF and Bag of Words. The score in recall-10 using pre-trained Word Embeddings models can be achieved by recall-1 using the trained model.

Table V shows the performance of the multiple pre-trained Word Embeddings models. The scores are worse than using TF-IDF and Bag of Words.

The experimental results showed that for mapping to ATT&CK Techniques and Sub-Techniques, the highest recall-10 score, 0.45575, was achieved by using Unigram and Bigram with TF-IDF and Snowball Stemmer. For mapping to ATT&CK Tactics, the highest recall-10 score, 0.75397, was achieved by using Unigram and Bigram, with TF-IDF and Snowball Stemmer.

TABLE III
RECALL-N FOR MAPPING LINUX COMMANDS TO MITRE ATT&CK TECHNIQUES AND SUB-TECHNIQUES

Model	Recall-1	Recall-3	Recall-5	Recall-10
Unigram, TF-IDF + SS	0.16865	0.33135	0.38492	0.40972
Unigram, TF-IDF + WNL	0.16964	0.33433	0.40278	0.44980
Unigram, CountVec + SS	0.17560	0.31448	0.35714	0.41508
Unigram, CountVec + WNL	0.15972	0.29067	0.35119	0.39028
Uni + Bi, TF-IDF + SS	0.18948	0.33333	0.38889	0.45575
Uni + Bi, TF-IDF + WNL	0.16567	0.33135	0.38988	0.44286
Uni + Bi, CountVec + SS	0.18750	0.31746	0.35615	0.41111
Uni + Bi, CountVec + WNL	0.14782	0.26984	0.35218	0.40317

TABLE IV
RECALL-N FOR MAPPING LINUX COMMANDS TO MITRE ATT&CK TACTICS

Model	Recall-1	Recall-3	Recall-5	Recall-10
Unigram, TF-IDF + SS	0.29960	0.50595	0.61111	0.72024
Unigram, TF-IDF + WNL	0.27381	0.53968	0.65675	0.72619
Unigram, CountVec + SS	0.32738	0.48810	0.56349	0.70437
Unigram, CountVec + WNL	0.27976	0.43254	0.54762	0.66468
Uni + Bi, TF-IDF + SS	0.30357	0.53968	0.59524	0.75397
Uni + Bi, TF-IDF + WNL	0.25595	0.50992	0.57143	0.73413
Uni + Bi, CountVec + SS	0.35119	0.51587	0.56548	0.71627
Uni + Bi, CountVec + WNL	0.26190	0.46429	0.54167	0.67857

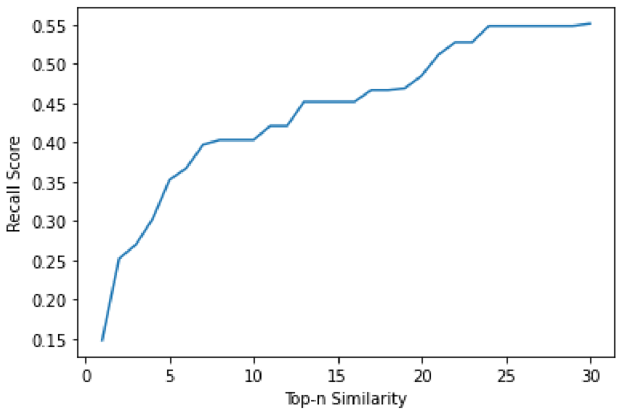

Fig. 2. Recall Score of Top-n Similarity (CountVec + Uni and Bigram with WordNet Lemmatizer)

V. Discussions

A. Top-N and Recall Score

As seen in Figure 2, as the number of n in top-n increasing, the recall score also increases. However, increasing the top-n number defeats the purpose of using machine learning, because one of the objectives of using machine learning is to automate a process and decrease human involvement and workload.

TABLE V
RECALL-N FOR MAPPING LINUX COMMANDS TO MITRE ATT&CK TECHNIQUES AND SUB-TECHNIQUES USING PRE-TRAINED WORD2VEC

Pre-Trained Model	Recall-1	Recall-3	Recall-5	Recall-10
fasttext-wiki-news-subwords-300	0.05060	0.07738	0.11210	0.14583
word2vec-google-news-300	0.08433	0.13591	0.14583	0.17659
glove-wiki-gigaword-300	0.0754	0.09028	0.15079	0.19444
glove-twitter-200	0.05754	0.11706	0.12103	0.16171

Also, having a perfect recall score is trivial, which can be done by returning all the documents. Thus, choosing a reasonable top-n that balances the number of documents returned and the recall score would be important. To help us choose the number of top-n, we use the elbow method approach [21], which is usually used in K-Means to choose an optimal cluster number. This ”elbow” is a point where the additional returns are no longer worth the additional cost. Figure 2 shows the elbow point around n = 7. Future improvement should focus on increasing recall scores in the low top-n region.

B. *Predictions using BoW and TF-IDF*

Table VI shows the most similar MITRE ATT&CK Techniques or Sub-Techniques for command *curl* using a Unigram+Bigram, TF-IDF, and WordNet Lemmatizer model.

TABLE VI
TOP-10 MOST SIMILAR ATT&CK TECHNIQUES OR SUB-TECHNIQUES FOR COMMAND ’CURL’ USING UNI+BI, TF-IDF + WNL

Rank	Name
1	Exfiltration Over Alternative Protocol
2	Ingress Tool Transfer
3	Lateral Tool Transfer
4	File Transfer Protocols
5	Web Protocols
6	Exfiltration Over Unencrypted Non-C2 Protocol
7	Steal Web Session Cookie
8	Mail Protocols
9	Password Guessing
10	Exfiltration Over Symmetric Encrypted Non-C2 Protocol

According to the data collected from the MITRE website, command *curl* can be mapped to Exfiltration Over Alternative Protocol, Exfiltration Over Unencrypted Non-C2 Protocol, Unix Shell, and Ingress Tool Transfer. In this example, three Techniques and Sub-Techniques are matched between prediction and benchmark datasets. The recall score for this specific command and model is 0.75. From Table VI, we can see that *curl* can be mapped to File Transfer Protocols. We know that *curl* can be used for FTP, even though it is not listed on the benchmark dataset. This is also the justification for using recall as a metric to benchmark our model.

Table VII shows the most similar MITRE ATT&CK Techniques or Sub-Techniques for command *curl* using a Unigram, CountVectorizer, and Snowball Stemmer model. The recall score for this specific command and model is 0.5.

Table VI and Table VII reveal results for command *curl* from different models. For future work, it may be interesting to test whether we can combine both models to produce more accurate results.

C. *Word2Vec*

Compared to BoW and TF-IDF, the performance of Word2Vec is the most underperformed. This is probably due to the contextual differences between the text used in training (pre-trained models) and prediction. For example, in cybersecurity, the words actor should have a similar meaning to the words attacker, hacker, and adversary, as they are actually

TABLE VII
TOP-10 MOST SIMILAR ATT&CK TECHNIQUES OR SUB-TECHNIQUES FOR COMMAND ’CURL’ USING UNIGRAM, COUNTVECTORIZER + SNOWBALLSTEMMER

Rank	Name
1	Lateral Tool Transfer
2	File Transfer Protocols
3	Ingress Tool Transfer
4	Steganography
5	Hidden Files and Directories
6	Application Layer Protocol
7	Exfiltration Over Alternative Protocol
8	System Owner/User Discovery
9	Deobfuscate/Decode Files or Information
10	SSH

TABLE VIII
TOP 5 MOST SIMILAR WORDS TO WORD ’ACTOR’

Model	1	2	3	4	5
fasttext-wiki-news-subwords-300	actress	actors	actor-	non-actor	actor-singer
word2vec-google-news-300	actress	Actor	thesp	thespian	actors
glove-wiki-gigaword-300	actress	starring	actors	comedian	starred
glove-twitter-200	actress	director	actors	singer	comedian

referring to the same entity. However, in daily life, the word actor may be similar to the word actress. Table VIII shows the top 5 most similar words to word actors using different pre-trained models in various pre-trained models.

VI. CONCLUSION

Machine learning and NLP have been used in cybersecurity research for some time. Our research used multiple NLP models and compared their performance in mapping Linux commands descriptions to MITRE ATT&CK Techniques and Sub-Techniques, and Tactics descriptions. The experiment results show that in most cases, TF-IDF models perform better than BoW models, and the model using both Unigram and Unigram + Bigram tokenizer performs better than Unigram only. This is expected because TF-IDF gives weighting according to how often the word appears in the document and in the corpus, and Bigram provides extra information like spatial dependency.

Pre-Trained Word Embeddings do not perform well for similarity matching in the cybersecurity context. This is due to the different word contexts trained in the pre-trained model and text used in this research. For future works, we plan to create a word embedding model specific to the cybersecurity context, experiment using different vectorization and tokenization techniques, and use an ensemble model.

ACKNOWLEDGMENT

This work was supported by The Information Society Innovation Fund (ISIF Asia), under the Knowledge Research Grant for 2021.

REFERENCES

[1] L. Spitzner, *Honeypots: Tracking hackers.* Addison Wesley Professional, 2002.
[2] M. Oosterhof, “cowrie/cowrie: Cowrie ssh/telnet honeypot.” [Online]. Available: https://github.com/cowrie/cowrie

[3] MITRE, “Faq — mitre att&ck®,” 2021. [Online]. Available: https://attack.mitre.org/resources/faq/

[4] R. Djap, C. Lim, K. E. Silaen, and A. Yusuf, “Xb-pot: Revealing honeypot-based attacker’s behaviors,” in *2021 9th International Conference on Information and Communication Technology (ICoICT)*. IEEE, 2021, pp. 550–555. [Online]. Available: https://doi.org/110.1109/ICoICT52021.2021.9527422

[5] Ryandy, C. Lim, and K. E. Silaen, “Xt-pot: Exposing threat category of honeypot-based attacks,” in *Proceedings of the International Conference on Engineering and Information Technology for Sustainable Industry*, ser. ICONETSI. Association for Computing Machinery, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1145/3429789.3429868

[6] M. Boffa, G. Milan, L. Vassio, I. Drago, M. Mellia, and Z. B. Houidi, “Towards nlp-based processing of honeypot logs,” in *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2022, pp. 314–321. [Online]. Available: https://doi.org/10.1109/EuroSPW55150.2022.00038

[7] D. Trizna, “Shell language processing: Unix command parsing for machine learning,” *arXiv preprint arXiv:2107.02438*, 2021.

[8] Z. Hussain, J. K. Nurminen, T. Mikkonen, and M. Kowiel, “Command similarity measurement using nlp,” in *10th Symposium on Languages, Applications and Technologies (SLATE 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[9] K. Kanakogi, H. Washizaki, Y. Fukazawa, S. Ogata, T. Okubo, T. Kato, H. Kanuka, A. Hazeyama, and N. Yoshioka, “Tracing cve vulnerability information to capec attack patterns using natural language processing techniques,” *Information*, vol. 12, no. 8, p. 298, 2021. [Online]. Available: https://doi.org/10.3390/info12080298

[10] M. R. Shahid and H. Debar, “CVSS-BERT: Explainable Natural Language Processing to Determine the Severity of a Computer Security Vulnerability from its Description,” in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2021, pp. 1600–1607. [Online]. Available: https://doi.org/10.1109/ICMLA52953.2021.00256

[11] G. Ayoade, S. Chandra, L. Khan, K. Hamlen, and B. Thuraisingham, “Automated threat report classification over multi-source data,” in *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2018, pp. 236–245. [Online]. Available: https://doi.org/10.1109/CIC.2018.00040

[12] P. Karuna, E. Hemberg, U.-M. O’Reilly, and N. Rutar, “Automating Cyber Threat Hunting Using NLP, Automated Query Generation, and Genetic Perturbation,” *arXiv preprint arXiv:2104.11576*, 2021.

[13] B. Ampel, S. Samtani, S. Ullman, and H. Chen, “Linking Common Vulnerabilities and Exposures to the MITRE ATT&CK Framework: A Self-Distillation Approach,” *arXiv preprint arXiv:2108.01696*, 2021.

[14] A. Kuppa, L. Aouad, and N.-A. Le-Khac, “Linking CVE’s to MITRE ATT&CK Techniques,” in *The 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–12. [Online]. Available: https://doi.org/10.1145/3465481.3465758

[15] M. Kerrisk, “The linux man-pages project,” https://www.kernel.org/doc/man-pages/, (Accessed on 09/10/2022).

[16] J. Ondricek, “MITRE CTI enterprise-attack,” (Accessed on 09/10/2022). [Online]. Available: https://github.com/mitre/cti/tree/master/enterprise-attack

[17] S. Vajjala, B. Majumder, A. Gupta, and H. Surana, *Practical natural language processing: a comprehensive guide to building real-world NLP systems*. O’Reilly Media, 2020.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Advances in neural information processing systems*, vol. 26, 2013.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[20] Scikit-Learn, “scikit-learn Machine Learning in Python,” 2022. [Online]. Available: https://scikit-learn.org/stable/

[21] F. Liu and Y. Deng, “Determine the number of unknown targets in open world based on elbow method,” *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 5, pp. 986–995, 2021. [Online]. Available: https://doi.org/10.1109/TFUZZ.2020.2966182