Named Entity Recognition in Cybersecurity

Mini-Project

Under the guidance of Dr. Chinmayananda

Khushi G K	20BCS071
Pratham Harshvardhan Dave	20BCS102
Rakshita Y	20BCS107
Shubh Bindal	20BCS121

In this survey, we look at the current methodologies used to detect and prevent Cyberattacks using Machine Learning and Pattern recognition techniques.

Machine learning algorithms play a pivotal role in identifying anomalous patterns and potential threats in vast datasets, enhancing the detection capabilities of security systems. Meanwhile, pattern recognition techniques assist in recognizing recurring attack patterns and behaviors, contributing to a more robust defense against cyber threats.

Our exploration will encompass a range of topics, including data preprocessing, feature selection, and the integration of machine learning models into intrusion detection systems. By shedding light on these critical aspects, we aim to contribute to the ongoing efforts to fortify our digital infrastructure against the ever-present and evolving cyber threats.

At the end, we will propose a theoretical method of our own and help explain why it is difficult to employ the concept of Machine Learning to such domain. A few papers have been summarized before to give the reader a general comprehension and an idea of the current work that has been done and the aspects of cybersecurity it targets.

INTRODUCTION

First, we look at what the concept of NER means. Named Entity Recognition (NER) is a natural language processing (NLP) task that involves identifying and categorizing named entities (e.g., names of people, organizations, locations, dates, and more) in text. NER is essential for extracting structured information from unstructured text data.

NER Models:

- **Rule-Based Models**: These models use handcrafted rules and patterns to identify named entities. They are straightforward but may lack the ability to adapt to new data.
- **Statistical Models**: These models use statistical techniques and machine learning algorithms to identify named entities based on features like word context and frequency. Popular algorithms include Hidden Markov Models (HMM) and Conditional Random Fields (CRF).
- **Deep Learning Models**: Deep learning models, particularly Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformer-based models, have shown significant advancements in NER. Models like Bidirectional LSTMs and BERT (Bidirectional Encoder Representations from Transformers) have achieved state-of-the-art performance.

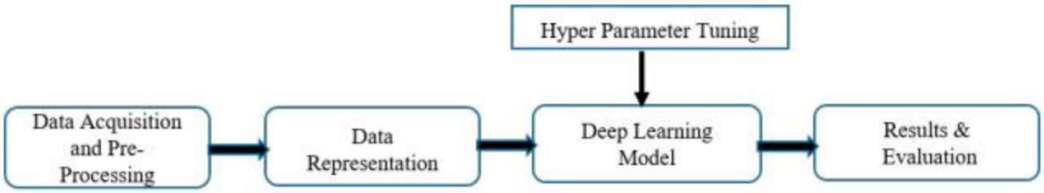
In the field of cybersecurity, abstraction of data can be presented at many layers. It can range from the bits transmitted at the physical wire to the large chunks of sentences sent in an email. Each layer of abstraction has an extremely different functionality than the other and as such, the models that work on one layer cannot be scaled onto the other. Let us now look at a few papers that use NER in the field of cybersecurity.

Study of Word Embeddings for Enhanced Cyber Security Named Entity Recognition

Aim: The goal of this research paper is to assess the effectiveness of various word embeddings in enhancing Named Entity Recognition (NER) for cybersecurity-related text. Given that a substantial portion of cybersecurity information is unstructured text, employing machine-assisted analysis is imperative. While traditional NER focuses on common entity types like people and places, the cybersecurity domain involves a broader range of specialized entities. The study evaluates both general-purpose pre- trained word

embeddings and task-specific embeddings generated through fine-tuning on a supervised dataset. This analysis aims to identify the most suitable word embedding techniques for improving NER performance in the realm of cybersecurity.

Methodology: This section covers a brief on a dataset acquired, word embeddings considered, deep learning architectures experimented with, and evaluation measures employed. Below Figure shows a high-level diagram of the proposed methodology.



Data Acquisition & Pre-Processing: For this study focused on Cyber-NER, the publicly available Cybersecurity NER corpus was utilized. In English, there are only two such annotated datasets accessible: the Cybersecurity NER corpus 2019 by Saganowski et al. [4,5], and the auto-labelled corpus provided by Bridges et al.. The auto-labelled-corpus dataset was chosen due to its larger number of examples and NER tags. It includes data from sources like Microsoft Security Bulletin, Metasploit, and NVD (National Vulnerability Database), comprising 13865 sentences with 24 BIO labelled named entities). Out of these, 10500 sentences were used for training, and the remaining 3365 for testing. However, it's worth noting that due to the auto-labelling strategy, some NER tag assignment errors were observed in the dataset. It is provided in JSON format but has been pre-processed to be in CoNLL2000 format for use in the downstream task of cybersecurity NER.

Word Embeddings: The literature introduces a range of word embeddings, each distinguished by their intended use and the methods employed for learning them. Since embeddings play a crucial role in initially conveying meaning to subsequent tasks, their impact on overall task performance necessitates empirical assessment. In the investigation of word embeddings, general-purpose embeddings are typically the initial preference due to the accessibility of readily available, comprehensive pre-trained embeddings that encompass a wide-ranging general-purpose vocabulary. The embeddings examined in this context include non-contextual embeddings like GloVe [1], fastText[2] , and contextual embeddings such as BERT[3]

Deep Learning Training Pipelines The study conducted experiments employing two pipelines to assess word embeddings in the context of the Cyber NER task. The widely accepted BiLSTM+CRF pipeline was employed for evaluating both pre-trained and task-adapted fine-tuned embeddings. Additionally, a position-wise linear feed-forward neural network was employed to evaluate the fine-tuned contextual BERT embedding.

Selection of Hyper Parameters and optimization: Optimal model performance requires fine-tuning of model parameters. Through a series of rigorous experiments, the batch size, maximum sequence length, and units in BiLSTM hidden layers were systematically adjusted to determine the parameter configuration for the highest-performing model.

Evaluation Metric The standard metrics of F1-Score, Precision and Recall are used for performance. Further, to capture the effect of class imbalance, Micro, Macro and Weighted Average of F1- Score are also calculated. For better error analysis entity wise break-up of results is also calculated.

Conclusion "This study lays the foundation for advancing domain-specific cyber security Named Entity Recognition (NER) techniques, particularly in resource-limited environments. It demonstrates that customizing

embeddings through fine-tuning for the specific task consistently yields superior results compared to using general-purpose pre-trained embeddings for Cyber NER.

A Review of Machine Learning based Zero-Day attack-detection: Challenges and Future directions

Aim: The aim of the paper is to conduct a comprehensive review of machine learning-based zero-day attack detection methods, compare and contrast their pros and cons, and identify key design and evaluation gaps. The paper also aims to provide insights into the types of machine learning models used, training data, zero-day attack testing data, and evaluation results. Additionally, the paper discusses the challenges and future directions in the field of machine learning-based zero-day attack detection.

Methodology: In order to identify similar assaults and adapt to their changing behavior, machine learning (ML) algorithms create a model using sample data, or training data. These algorithms are capable of finding intricate statistical patterns in the training data. We evaluate the zero-day attack detections based on outlier analysis. A review is conducted on zero-day attack detection using unsupervised learning methods.

Model: Outlier-based zero-day attack detection using normal data Zero-day attack detection based on outliers trains the detection model using typical data that has been seen in the past. By assuming that a zero-day assault deviates much from typical behavior, the model is able to identify zero-day attacks. For outlier-based zero-day attack detection, two typical machine learning models are auto-encoder and one-class Supportive Vector Machine (SVM).

One-class SVM-based detection:

The One-class SVM (Support Vector Machine) method is used to detect anomalies in a dataset. In this approach, a dataset containing normal data points is analyzed to spot any data points that are unusual or anomalous.

To implement this technique, the original data is transformed into a new feature space using a function. This new feature space allows for a better examination of patterns within the data. The objective is to find a line or hyper-plane in this new feature space that is positioned farthest from the center of the data.

Ideally, all the original normal data points should be positioned on one side of this line, away from the center. When a new data point is introduced, the analysis focuses on which side of this line in the new feature space it falls. If the new data point is on the same side as the normal data, it is classified as normal. However, if it falls on the opposite side, it is flagged as abnormal or unusual.

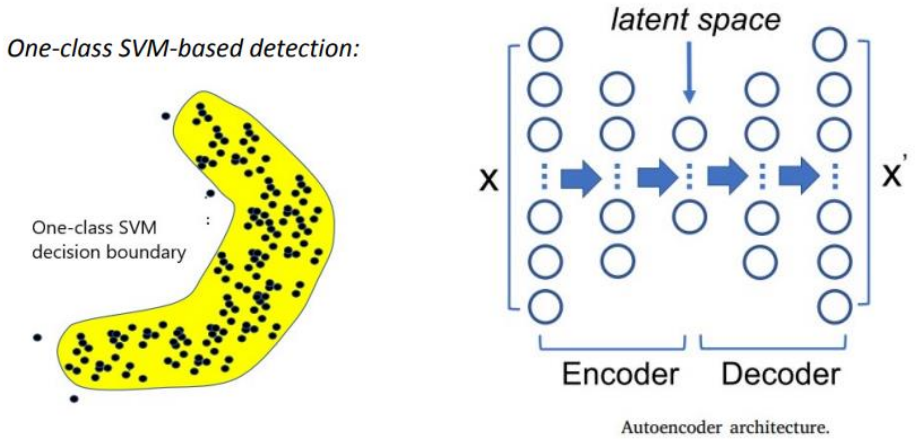
Autoencoder-based detection:

Autoencoder-based zero-day attack detection is part of the category of reconstruction methods. It operates under the assumption that "normal" data can be accurately reconstructed by an autoencoder with minimal reconstruction errors. In contrast, outliers or abnormal data points cannot be effectively reconstructed. An autoencoder is a type of neural network that is trained to replicate its input as closely as possible in its output. An autoencoder is like a machine that learns to shrink and then expand data. Imagine you have some original data, and this machine takes that data and squishes it into a smaller, simpler representation called the "latent space."

The same machine can then stretch that simplified data back to its original form.

Evaluation:

One-class SVM vs. Autoencoder:



Autoencoder architecture.

1. **Datasets**: this study utilizes two well-known data sets: the CIC-IDS2017 data set and the NSL-KDD data set. The CIC-IDS2017 data set comprises five-day pcap files that document various types of network traffic, including benign traffic and a variety of attacks. The NSL-KDD data set, on the other hand, contains network-related features extracted from a series of TCP connections captured from a local area network.

2. **Model Training**: The models used in this study are trained exclusively with normal (benign) network traffic. The benign instances are split into 75% for training and 25% for validation. After training the model, it is assessed using both benign and attack traffic. An instance is flagged as a zero-day attack if the Mean Squared Error (MSE), which measures the difference between the original instance (x) and the reconstructed instance (x'), exceeds a predefined threshold. In the case of training the One-Class SVM, a parameter 'v' is specified, representing the range between 0 and 1.

3. **Experiment Results**: The accuracy of zero-day attack detection varies significantly depending on the specific type of attack. When it comes to identifying recognizable zero-day attacks, One-Class SVM performs well. However, for more complex zero-day attacks, autoencoders tend to outperform One-Class SVM. Lastly, both models exhibit a low miss rate, indicating a low rate of false positives in their detections.

Conclusion: This paper provides a thorough examination of machine learning-based methods for zero-day attack detection. The review is organized by the types of machine learning models used in unsupervised learning. A central challenge in machine learning-based zero-day attack detection is the absence of zero-day attack representations in the available datasets. Moreover, the limited and fragmented nature of the datasets and the lack of a comprehensive feature space contribute to the proposed models not achieving the desired levels of accuracy, robustness, and reliability.

Data and knowledge-driven Named Entity Recognition for Cyber Security

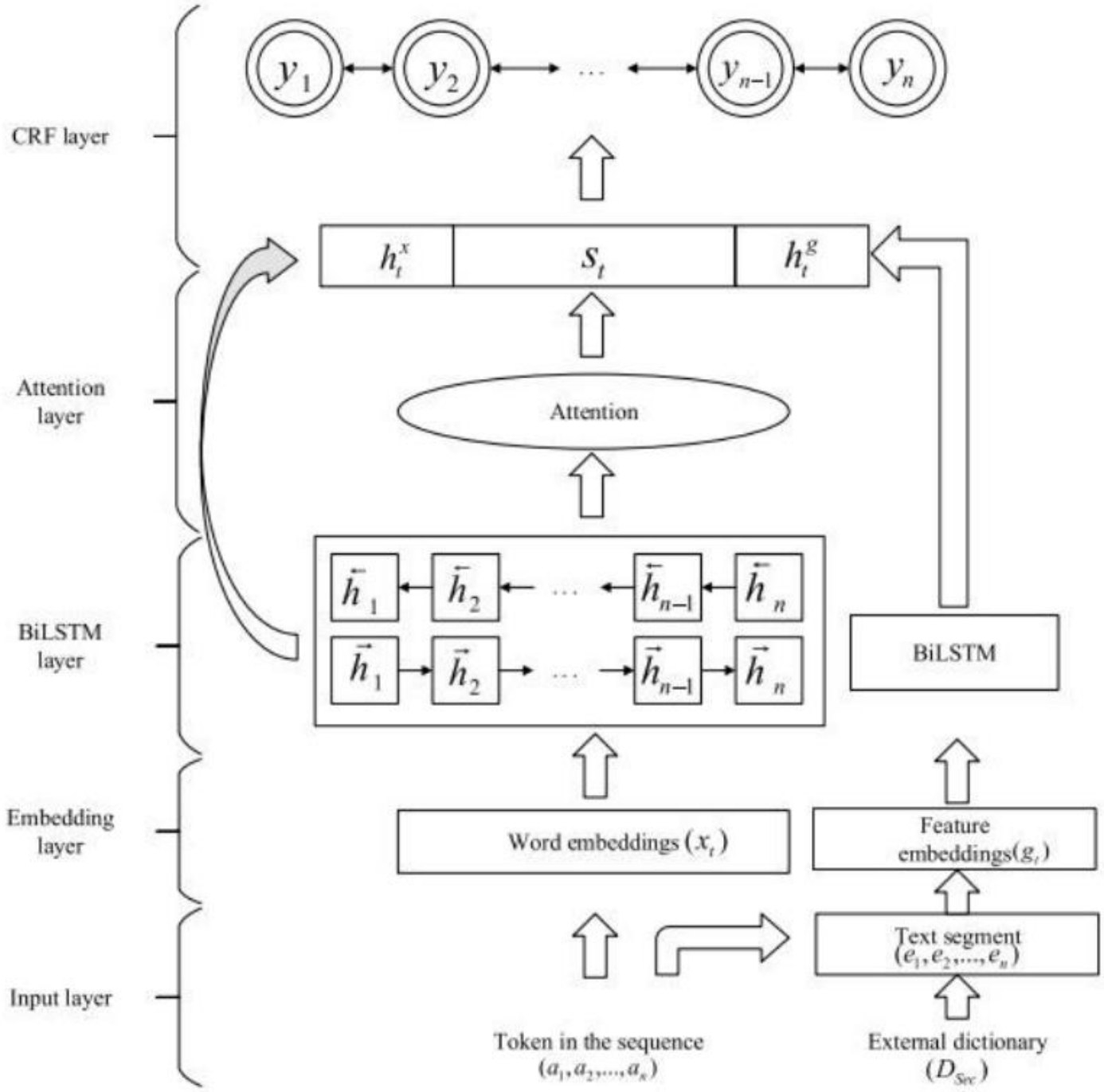
Paper's Aim: Cyber security Named Entity Recognition (NER) involves the identification and categorization of cyber security-related terms within a diverse set of texts from various sources. While deep neural networks in machine learning can automatically glean text features from extensive datasets, they often struggle with recognizing uncommon entities. Gasmi et al. introduced a deep learning approach for NER in cyber security, achieving commendable results with an F1 score of 82.8%. Nevertheless, accurately pinpointing rare entities and complex terminology remains a challenge.

To address this issue, this study proposes a novel model that combines data-driven deep learning techniques with knowledge-driven dictionary methods. This fusion enables the creation of dictionary-based features to aid in the recognition of rare entities. Furthermore, an attention mechanism is integrated into the data-driven deep learning model to enhance local text features, providing a more robust contextual representation and thereby improving the recognition of complex entities.

Methodology In this paper, the approach to cyber security Named Entity Recognition (NER) follows a sequence labeling paradigm, akin to the universal NER method. The initial term of the entity is designated as "B-type",

the subsequent terms within the entity are labeled as "I-type", and any term not constituting an entry is labelled as “O”.

Model:



The Above figure shows the over all structure of our model. The entire model consists of 5 parts: 1) Input layer; 2) Embedding layer; 3) BiLSTM layer; 4) Attention layer; 5) CRF layer The input layer comprises both text and dictionary feature vectors. The embedding layer acquires the semantic information of the text vocabulary by incorporating pre- trained word embeddings specifically tailored for cyber security. Subsequently, the BiLSTM layer conducts comprehensive feature extraction and produces a vector that is passed on to the attention layer. This attention layer allocates varying importance to distinct feature vectors within the global features, enabling the extraction of localized features. Following the processing of text embedding, the

external dictionary feature embedding is introduced. The BiLSTM network is also employed to encode features, yielding information about phrase boundaries. Ultimately, the concatenated feature vector sequence, encompassing global features, local features, and domain-specific dictionary features, is fed into the CRF decoding layer. Within the CRF layer, the Conditional Random Field (CRF) serves as the model for named entity recognition, effectively predicting the optimal global labeling sequence.

Conclusion: This paper introduces a Named Entity Recognition (NER) method tailored for cyber security. The experimental outcomes demonstrate that the suggested model outperforms established conventional methods, owing to the incorporation of domain-specific dictionary features and a multi-attention mechanism. Notably, the model exhibits enhanced effectiveness in identifying rare entities. Nevertheless, there is scope for refinement in extracting complex entities. Moreover, text mining within the realm of cyber security holds paramount importance. Information extraction in this domain still grapples with various challenges, including the identification of nested entities and extraction of overlapping relationships.

LSTM Recurrent Neural Networks for Cybersecurity Named Entity Recognition

Aim: The aim of the paper is to propose a domain-independent approach for cybersecurity Named Entity Recognition (NER) using LSTM recurrent neural networks. The goal is to automate the extraction of cybersecurity information from unstructured online sources and improve the accuracy of NER in the cybersecurity domain. The paper aims to demonstrate that the proposed method, which combines LSTM with Conditional Random Fields (CRFs), outperforms traditional methods that rely on feature engineering.

Methodology: It is demonstrated that a domain-agnostic approach, leveraging recent deep learning advancements and word embeddings, outperforms traditional techniques like Conditional Random Fields (CRFs). This novel approach utilizes word2vec word embedding, representing each word in the corpus with a low-dimensional vector.

Model: The model implemented is LSTM-CRF architecture, following the proposal by Lampal et al. , within the domain of cybersecurity Named Entity Recognition (NER). This architecture combines LSTM, word2vec models, and CRFs, with its standout characteristic being its adaptability to different domains and entity types. It operates effectively with an annotated corpus structured similarly to the CoNLL-2000 dataset. A performance comparison was conducted between LSTM-CRF and CRFSuite, recognized for its reputation as one of the swiftest and most precise CRF implementations.

LSTM-CRF: LSTMs consist of multiple interconnected layers, including an input gate, output gate, memory cell, and forget gate. These gates determine which input to store in the memory cell and which previous state to forget, effectively managing information over time. For a sequence of n words, each represented as a d-dimensional vector, LSTMs compute both left context and right context. The right context is derived using a reverse LSTM, processing the text sequence in reverse order. This architectural combination, comprising a forward LSTM and a backward LSTM, is known as a Bidirectional LSTM and consists of three layers.

Starting from the bottom, the first layer is the embedding layer, which takes the input sequence of words and produces dense vector representations (embeddings) for each word in the sequence. The sequence of embeddings is then passed to the bidirectional LSTM layer, where the input is refined and forwarded to the final CRF layer.

In the last layer, the Viterbi algorithm is employed to determine the neural network's output, representing the most probable tag for each word.

Evaluation:

Competitor System: Utilizing word embeddings in both systems allows us to make a direct comparison between the CRF method and the proposed LSTM-CRF architecture, effectively isolating the impact of word embeddings. Employed CRFSuite with its default settings to train a CRF model.

Gold standard corpora: The assessment encompassed the evaluation of approximately 40 different entity types across three corpora. Furthermore, a performance analysis was conducted specifically for a subset of seven highly relevant entities within the cybersecurity domain. The text in the corpus was auto-labeled and pertained to the field of cybersecurity.

Text Preprocessing: All the corpora were consolidated into a single JSON file. The file was converted into the CoNLL2000 format, which served as the input for the LSTM-CRF model. In this newly unified annotated corpus, the distinctions between the three original corpora and annotated each word individually on a separate line were eliminated. Each line contains the word from the text and its associated entity type.

Regarding the CRF model, CRFSuite necessitates the training data to adhere to the CoNLL2003 format, which encompasses Part of Speech (POS) and chunking details, with the NER tag taking precedence.The process began with a conversion of the data back to its initial structure, which consisted of paragraphs. Ultimately, the text was reverted to its anticipated format. Regarding the CRF model, CRFSuite mandates that the training data adheres to the CoNLL2003 format, where Part of Speech (POS) and chunking details are provided, and the NER tag is positioned as the primary element.Eventually, the text was transformed to conform to the specified format.

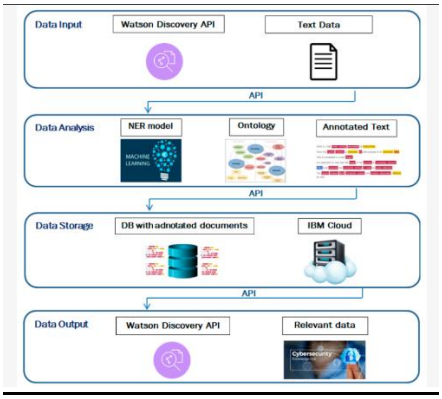
Evaluation Metrics: The annotated corpus was segmented into three separate, non-overlapping subsets. Specifically, 70% of the data was used for training the model, 10% for a holdout cross-validation set (or development), and the remaining 20% for model evaluation. We conducted a comparison between the two models, LSTM-CRF and CRF, based on metrics such as accuracy, precision, recall, and F1-score.

Results: In the course of 100 training iterations, accuracy values were assessed for the test set. LSTM-CRF began with an initial accuracy of 95.8% after the first iteration and showed a gradual improvement, ultimately stabilizing at values between 98.2% and 98.3% from the 23rd iteration onward until the conclusion of training. Conversely, the CRF method had a slower start with accuracies around 65% but quickly escalated to reach accuracies of 96%, where it plateaued, eventually reaching 96.35% at the end of the training process. The performance metrics in terms of precision, recall, and F1-score show that the results for LSTM-CRF are better than their CRF counterparts. Conclusion: It is seen that the overall item accuracy of the resulting LSTM-CRF model is higher by 2% than the CRF model. The notable feature of the LSTM-CRF method is its independence from feature engineering, and it is entirely neutral to entity types.

Named-Entity-Recognition-Based Automated System for Diagnosing Cybersecurity Situations in IoT Networks

Aim: The aim of this paper is to create a system that helps detect vulnerabilities in Internet of Things (IoT) devices more easily. It analyses and understands information related to security threats, vulnerabilities, and risks automatically using a technology known as named entity recognition (NER). This system gathers data from different sources, like articles and reports, and then organizes it to help security experts. By doing this, it makes it simpler for users to identify possible issues with their IoT devices. Additionally, the system refreshes itself frequently with fresh data to ensure that it is up to speed on the most recent developments in cybersecurity. This helps in identifying vulnerabilities in IoT devices before they can be exploited by hackers.

Methodology: The methodology of this paper involves creating a specialized system for gathering information about the security of Internet of Things (IoT) devices. it involves creating a tool called "Cybersecurity Analyzer" that automatically finds and organizes security information for IoT devices. It does this by reading text documents and using a special vocabulary for IoT. The tool has four parts: one for getting data, one for understanding it, one for storing it, and one for searching it. They used technology from IBM to make this tool and created a special list of words to help it understand IoT security.



Model: Developing the NER model

Named Entity Recognition (NER) is a technique in natural language processing that identifies specific types of information in text, like names, dates, and organizations.

Training the Model:

- Trained NER model using a mix of data sources, like CVEs, research articles, technical reports, and online articles.
- This training process involved tagging entities (specific pieces of information) in the text.

How the Model Works:

- Building the model: Created a smart computer program focused on IoT security terms from their ontology, which is like a big list of IoT-related words and phrases.
- Recognizing Entities: The model was trained to identify specific types of information, such as IoT device names, security terms, and vulnerabilities.
- Flexibility: The model can also figure out things that are similar, even if they are not in the vocabulary.

Data Storage:

- The trained model is integrated into a service called Watson Discovery, which can automatically identify entities in documents.
- These documents, including IoT security-related data, are stored in the IBM Watson Discovery cloud platform and can be accessed through a REST API.

Evaluation: The standard metrics of F1-Score, Precision and Recall are used for performance. The model achieved a F1 score of 0.68, indicating its reliability. A high precision score (0.74) suggests it is fairly accurate, though recall (0.63) could be improved with more training data. Finally, stored relevant documents in JSON format, allowing users to easily access critical security information about their IoT devices.

Conclusion: This paper enhanced the process of identifying vulnerabilities in IoT systems by creating a domain ontology for IoT cybersecurity, selecting relevant CVEs, and implementing an entity recognition system. Users,

especially security experts, can now access and stay updated on IoT system vulnerabilities. Unlike other approaches, this system actively identifies security issues in IoT systems.

Proposed Method against Cyberattacks:

To begin with the idea, we shall take a look at a set of pre-defined attacks that are prevalent in the cyberworld

Oftentimes, many of these "attacks" are simply an exploit of a bug that is present within the system. These are very specific to each system and software configuration and are usually not applicable on the large scale, given the wide configurations and OS of systems we have today.

One of these is SQL Injection Attack, in which the hacker simply exploits the fact that the programmer had not put in place guards against SQL Query mishaps. Here, the application of ML is very limited & enough data cannot be gathered in this domain to avoid day-zero attacks. The main responsibility of security here lies only with the programmer themselves.

Drive-By Download Attack is a method in which a malicious program gets automatically downloaded by a website when you visit the page. The only way to fight against such attacks is for the Web-Browser to contain a global repository of legitimate websites & to ask the user each time before beginning an unsolicited download. Again, ML does not play any role here.

Many of these attacks, if present, are often mitigated by using encryption & vulnerability-free code. However, most of this is not possible in practice, and oftentimes, the industry & academia cannot trace the bits transferred in everyone's devices in-order to understand the nature of the attack. Even if done so, it is essentially ineffective as there is a different encryption used in every message which hinders the Model from being able to learn anything.

In order to implement ML as a means of security against Day-Zero attacks, one must violate the privacy policy of the user and assume that the model is able to see the decrypted version of the data. We propose a method through which there is some possibility of using ML as a viable source against recurring attacks, and even day-zero attacks, to the extent possible.

First and foremost, comes the issue of data collection. Herein, there is unfortunately not much one can do to obtain non-encrypted data on a large scale as it is considered a security breach and a violation of policy. However, what can be done is that the bits (i.e. the signals) that are coming through and fro from the receiver can be recorded at an intermediate node. Since encryption guarantees that none of it will be read by a middle-man, there is no harm to the User.

Second, the data is captured as a timeseries data in which the unrecognizable bits are stored in a sequential order. The times when no data is sent is left untouched, as it also helps to determine the network load without any additional meta-data.

Finally, the data is segregated into timestamps where known attacks had occurred. Since we do not have any knowledge of which exact bits of the code are a threat, we mark the entire range to be malicious with a confidence level of 100%

For the training, a two-fold model is used, in which a global database is queried to check whether a site is trusted, and other several factors such as the domain name, the URL, the security (HTTP/HTTPS) method are taken into consideration to provide a score from 0 to 100. There are several pre-existing tools that already do this. We normalize it between 0 to 1.

Coming to the model, there are several models already available in the market for all the domains. As such, there is little need to create a new model and experiment around, as the results will be very similar to current benchmarks and that a large carbon footprint is created for generating a viable model. For this experiment, we stick to BERT to perform NER.

The key idea we require here is the concept of SSL. Self-supervised learning (SSL) is a machine learning technique where a model learns to understand and represent data without explicit labels. It does this by using the data itself to create supervisory signals. In self-supervised learning, a portion of the data is used to define a task, such as predicting missing parts of the data or ranking data points. The model then learns to perform this task by extracting useful features from the data. This approach is particularly useful when labeled data is scarce or expensive to obtain. Self-supervised learning has been successful in natural language processing, computer vision, and other domains, as it enables models to pre-train on large datasets and then fine-tune for specific tasks, leading to improved performance in various applications.

Here, we simply jumble up the dataset while maintaining the labels along with the respective data-stream. More specifically, if we define one unit of data to be 100 bits of stream, we can shuffle the blocks of 100 bits while maintaining their classification. One key thing to note here is that we are sacrificing learning long distance relationship between the blocks of bits in-order to understand the functioning of each of these 100 bits. Since we are talking of a theoretical approach, we will proceed with the given idea. However, there are over 256^100 possible combinations, which are impossible to learn practically. The only way to mitigate this issue would be to have non-encrypted data alongside a label for the packet.

These can then be used to train the model as usual wherein the model makes a prediction and is corrected depending on its generated output. At the end of the training, assuming we have infinite compute capacity, data and time, there will be a model with a holistic idea of what each block of word means.

After this step, we now have two distinct methods, one being a hard-set rule based method with another one being an ML model. The outputs of which can then be fed to a neural network which can then be fine-tuned to assign weightage to each of method appropriately.

This model can then be put before the encryption layer that takes place before the data is sent out. The data is first read to check if it contains any sensitive content. This responsibility falls primarily on the OS. If it's sensitive, the model is triggered and the previous 10-20 kilobytes of data is cross-checked to see if there's any attempt of malice. This way, the overhead is extremely limited.

Conclusion: We've covered a few papers and a novel approach of detecting breaches with a minimal overhead. The idea can be further continued if a scope for it is found in the near future.

References:

1. Bridges, R, Jones C, MD. Iannacone KT, Goodall J (2013) Automatic labeling for entity extraction in cyber security. arXiv preprint arXiv:1308.4941.
2. Collobert, R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537
3. Dionísio, N, Alves F, Ferreira P, Bessani A (2019) Cyber threat detection from twitter using deep neural networks In: 2019 International Joint Conference on Neural Networks (IJCNN), 1–8.. IEEE, Budapest.
4. Gasmi, H, Bouras A, Laval J (2018) Lstm recurrent neural networks for cyber security named entity recognition In: Proceedings of the Thirteenth International Conference on Software Engineering Advances, Nice.
5. Hochreiter, S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780.

6. Joshi, A, Lal R, Finin T, Joshi A (2013) Extracting cybersecurity related linked data from text In: Proceedings of the 2013 IEEE Seventh International Conference on Semantic Computing, 252–259.. IEEE, Irvine.
7. Lee Jinhyuk, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang BioBERT: a pre-trained biomedical language representation model for biomedical text mining Bioinformatics, 36 (2020), pp. 1234-1240
8. Athira Gopalakrishnan, KP Soman, B. Premjith, A Deep Learning-Based Named Entity Recognition in Biomedical Domain
9. Beltagy Iz, Kyle Lo, Arman Cohan, SciBERT: A Pretrained Language Model for Scientific Text, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP) and the 9th International Joint Conference on Natural Language Processing, , Hong Kong, China (2020), pp. 3615-3620
10. G Veena, Deepa Gupta, S Lakshmi, J.T Jacob, Named Entity Recognition in Text Documents Using a Modified Conditional Random Field, Advances in Intelligent Systems and Computing, 709, Springer, Singapore (2018)
11. T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, Advances in neural information processing systems (2013), pp. 3111-3119
12. HudaS. et al. Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data
13. KimJ.-Y. et al. Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders
14. SameeraN. et al. Deep transductive transfer learning framework for zero-day attack detection
15. WangS. et al. Hyperparameter selection of one-class support vector machine by self-adaptive data shifting
16. K. Pohl, G. Bockle, and F. J. van Der Linden, Software Product Line ¨ Engineering: Foundations, Principles and Techniques. Springer Science & Business Media, 2005.
17. C. Kastner, S. Trujillo, and S. Apel, “Visualizing software product line ¨ variabilities in source code.” in SPLC (2), 2008, pp. 303–312.
18. K. Berg, J. Bishop, and D. Muthig, “Tracing software product line variability: From problem to solution space,” 2005, pp. 182–191.
19. J. Bosch, Design and use of software architectures: adopting and evolving a product-line approach. Pearson Education, 2000.
20. S. Apel and C. Kastner, “An overview of feature-oriented software ¨ development.” Journal of Object Technology, vol. 8, no. 5, 2009, pp. 49–84.
21. Ashton, K. That ‘internet of things’ thing. RFID J. 2009, 22, 97–114. [Google Scholar]
22. Palermo, F. Information Week. Available online: https://www.informationweek.com/strategic-cio/executive-insights-and-innovation/internet-of-things-done-wrong-stifles-innovation/a/d-id/1279157 (accessed on 3 March 2018).
23. Georgescu, T.M.; Iancu, B. An IoT architecture that uses semantic reasoning based security. In Proceedings of the 17th International Conference on Informatics in Economy (IE 2018), Iasi, Romania, 17–20 May 2018. [Google Scholar]
24. Lin, H.; Bergmann, N.W. IoT Privacy and Security Challenges for Smart Home Environments. Information 2016, 7, 44. [Google Scholar] [CrossRef]
25. Jurn, J.; Kim, T.; Kim, H. An Automated Vulnerability Detection and Remediation Method for Software Security. Sustainability 2018, 10, 1652. [Google Scholar] [CrossRef]