ThreatX Project Final Report

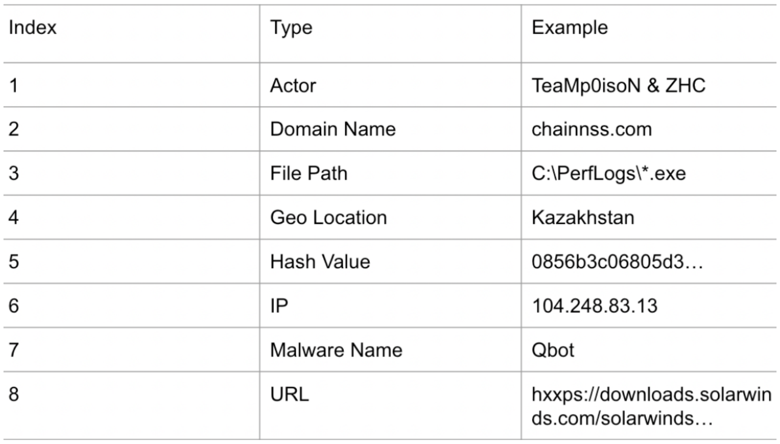
Xuehan Liu (1008401301) Wenxuan Zhang (1009230388)

ACKNOWLEDGEMENT

Word count: 1931
Compute penalty: 0%

INTRODUCTION

Cyber crimes are evolving, but thankfully, security experts are sharing their security insights online through security posts to help protect the community. This shared information is invaluable for companies to stay informed and for security analysts to keep up with latest cyber threats. Although there's a huge amount of security knowledge online, it's usually scattered, unstructured and in various topics. To simplify the task of extracting security knowledge, engineers tend to focus on specific threat signals, known as indicators of compromise (IoCs) from each security post. However, getting these IoCs from posts isn't easy, and companies pay a lot to manually sort through them. Our project, ThreatX, is designed to leverage Large Language Models (LLMs) to automatically detect IoCs from security reports, which could save time and money. Our project starts with identifying 8 particular IoC types to cut down the scope. Table 1. lists all 8 IoC types we handle with our model. Amongst all types, types like file path, IP address, and URL are simpler to identify because they are structured and follow specific patterns, while the rest can be tricky since they don't have consistent patterns.



Index	Type	Example
1	Actor	TeaMp0isoN & ZHC
2	Domain Name	chainnss.com
3	File Path	C:\PerfLogs\*.exe
4	Geo Location	Kazakhstan
5	Hash Value	0856b3c06805d3…
6	IP	104.248.83.13
7	Malware Name	Qbot
8	URL	hxxps://downloads.solarwinds.com/solarwinds…

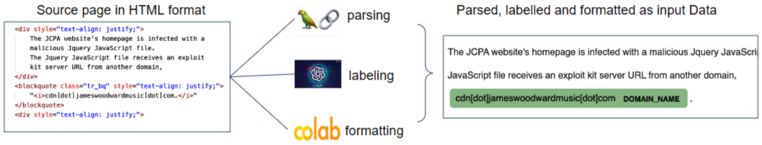
Table 1. IoC Types with Examples

BACKGROUND AND RELATED WORK

Indicators of Compromise (IoCs) are important in cybersecurity for detecting and responding to security threats. Consequently, researchers use various techniques for collecting IoCs from open source security references to identify patterns of malicious activity and take steps to prevent further attacks. Gao et al. Gao et al. (2022) propose an advanced system. In their work, they utilized a method which combines rules, neural networks, and convolutional neural networks for content processing. With the use of these deep learning models, both accuracy and efficiency have been improved significantly. However, their method struggles with novel, unseen threat patterns, and is less accurate with patterns with less training data. Our proposed method overcomes these limitations by using Large Language Models (LLMs). LLMs continually adapt to new cyber threats, making our approach more versatile in handling various cybersecurity challenges.

DATA AND DATA PROCESSING

We've acquired a repository with over 17,000 latest security reports from 31 authorized vendors. To simplify our project, we select 426 references from them, all in HTML format. Our raw data undergoes a three-stage pre-processing phase: parsing, labeling, and formatting. The entire data flow process is outlined in Figure 1.


Source page in HTML format
parsing
labeling
colab formatting
Parsed, labelled and formatted as input Data
DOMAIN_NAME

```
<div style="text-align: justify;">
    The JCPA website's homepage is infected with a
    malicious Jquery JavaScript file.
    The Jquery JavaScript file receives an exploit
    kit server URL from another domain,
</div>
<blockquote class="tr_bq" style="text-align: justify;">
    "<i>cdn[dot]jameswoodwardmusic[dot]com.</i>"
</blockquote>
<div style="text-align: justify;">
```

The JCPA website's homepage is infected with a malicious Jquery JavaScri

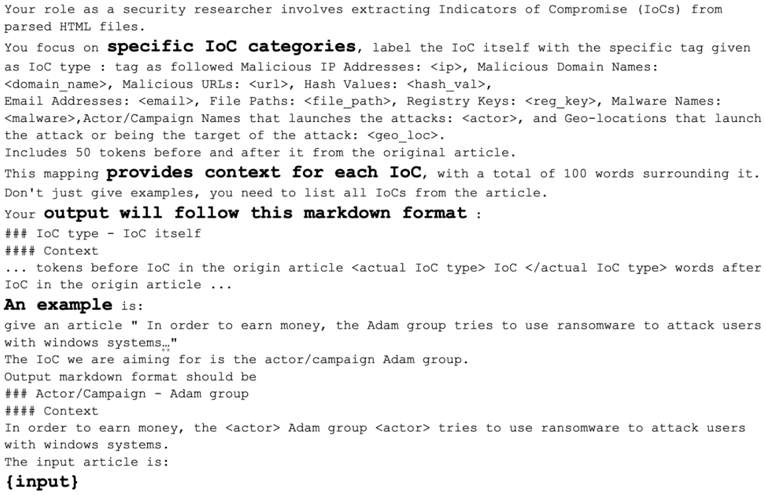
JavaScript file receives an exploit kit server URL from another domain,

cdn[dot]jameswoodwardmusic[dot]com DOMAIN_NAME .

Figure 1. Data Processing Flow

In the first step, we leverage a set of functions in Langchain for text parsing. We start by loading a given reference document, then utilize BSHTMLparser to extract text content based on HTML tags, converting it into plain text. Since there's a limit on input length, we break this plain text into segments, ensuring a preset overlap of tokens between these consecutive segments.

In the second step, we utilize the GPT API for data labeling. Each text segment is fed into GPT-4, guided by the prompt outlined in Figure 2.



```
Your role as a security researcher involves extracting Indicators of Compromise (IoCs) from
parsed HTML files.
You focus on specific IoC categories, label the IoC itself with the specific tag given
as IoC type : tag as followed Malicious IP Addresses: <ip>, Malicious Domain Names:
<domain_name>, Malicious URLs: <url>, Hash Values: <hash_val>,
Email Addresses: <email>, File Paths: <file_path>, Registry Keys: <reg_key>, Malware Names:
<malware>,Actor/Campaign Names that launches the attacks: <actor>, and Geo-locations that launch
the attack or being the target of the attack: <geo_loc>.
Includes 50 tokens before and after it from the original article.
This mapping provides context for each IoC, with a total of 100 words surrounding it.
Don't just give examples, you need to list all IoCs from the article.
Your output will follow this markdown format :
### IoC type - IoC itself
#### Context
... tokens before IoC in the origin article <actual IoC type> IoC </actual IoC type> words after
IoC in the origin article ...
An example is:
give an article " In order to earn money, the Adam group tries to use ransomware to attack users
with windows systems…"
The IoC we are aiming for is the actor/campaign Adam group.
Output markdown format should be
### Actor/Campaign - Adam group
#### Context
In order to earn money, the <actor> Adam group <actor> tries to use ransomware to attack users
with windows systems.
The input article is:
{input}
```

Figure 2. Five Components of Prompt to GPT-4

After we retrieve the response from GPT, we manually verify the accuracy of the extracted data. This involves a quick review of the context provided by GPT around each IoC. Using our domain expertise, we eliminate any inaccurately labeled data.

The last step involves the formatting of our input dataset in our own code. Each model requires a unique data format, so we independently create a data formatter for each one. The specific format for each model is detailed in the model section.

We evaluated 600 IoCs flagged by GPT-4, selecting 155 as our dataset. Figure 3 shows the distribution of these data points across various labels. Notably, our dataset is imbalanced, with 'malware name' being the most common, while 'file path' and 'IP' are the least represented categories.

To ensure all labels are represented in both training and testing data, we use stratified sampling to allocate 20% of the data points as a reserved test set for model evaluation. Figure 4 illustrates the distribution of training and test data across all labels. Notably, categories like IP and field path have limited data, which might impact the performance of the NER models.

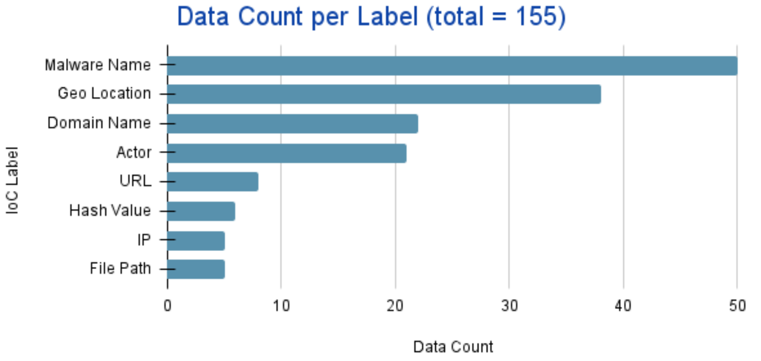


Figure 3. Distribution of Data Points over Labels

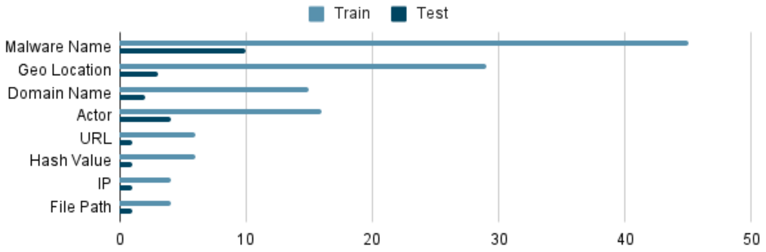


Figure 4. Distribution of Data over Labels for Train/Test Set

ARCHITECTURE AND SOFTWARE

Figure 5 illustrates our system's architecture, mainly comprising two components: a data parser and an NER model. The process starts with obtaining the provided HTML file, which is then processed by the same data parser mentioned in the data pre-processing stage. This file is then broken down into smaller chunks for efficiency consideration. Each chunk is fed into our Threat Named Entity Recognition (NER) model to extract specific types of IoCs. We employed three different NLP tools in our threat NER models, balancing cost and performance.

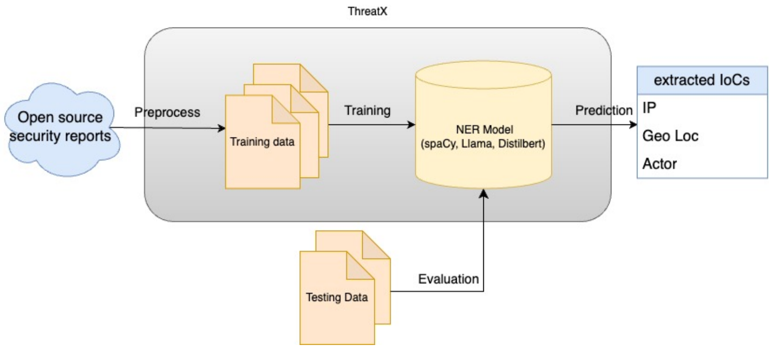
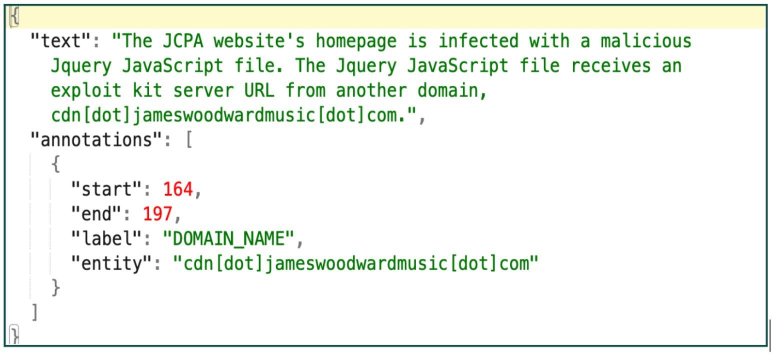


Figure 5. System Data Flow

Baseline Model

We construct a custom NER model using spaCy v2.0 as our baseline model. This model incorporates a deep convolutional neural network with residual connections, along with a transition-based approach for named entity parsing. We choose it as a baseline model aiming at an optimal balance between efficiency, accuracy, and adaptability.

Input data format Figure 6 shows a sample input for our baseline model. It includes a text chunk for extraction and annotations of all IoCs identified in the text. Each annotation details the start and end indices of the IoCs in the text, along with their label and IoC value.

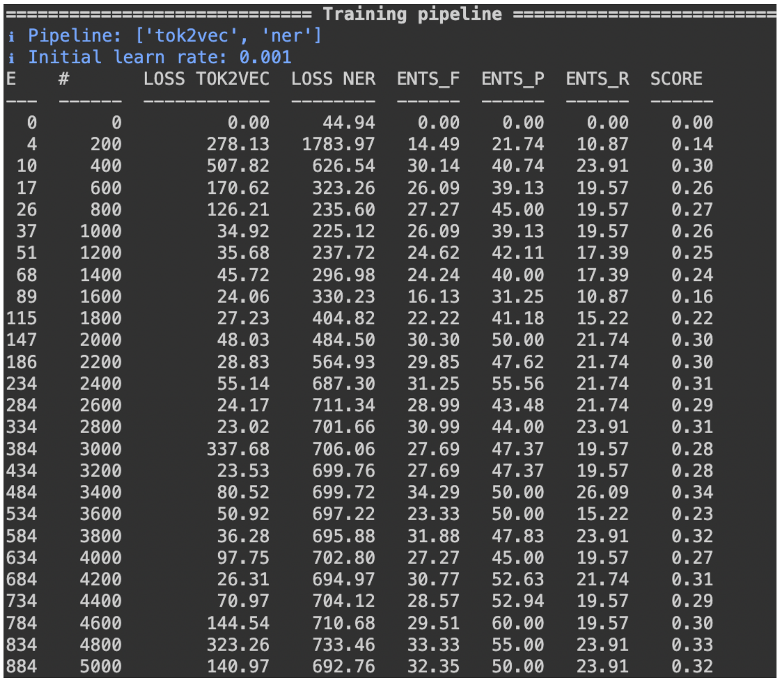


```
{
  "text": "The JCPA website's homepage is infected with a malicious
    Jquery JavaScript file. The Jquery JavaScript file receives an
    exploit kit server URL from another domain,
    cdn[dot]jameswoodwardmusic[dot]com.",
  "annotations": [
    {
      "start": 164,
      "end": 197,
      "label": "DOMAIN_NAME",
      "entity": "cdn[dot]jameswoodwardmusic[dot]com"
    }
  ]
}
```

Figure 6. Input Data for SpaCy Model

Experimental Setup And Outcome When training data is provided, it first undergoes tokenization by spaCy's tokenizer. Later, tokens are passed to the NLP training pipeline consisting of tok2vec and NER. Tok2vec is crucial for converting tokens into vector representations, and NER and classifies named entities in the text. The loss function for those components involve a combination of cross-entropy loss and a localization loss.

Figure 7 demonstrates the training process for our baseline model. Notably, there's fluctuation and difficulty in optimizing the loss for both tok2vec and NER components, possibly due to insufficient training data.



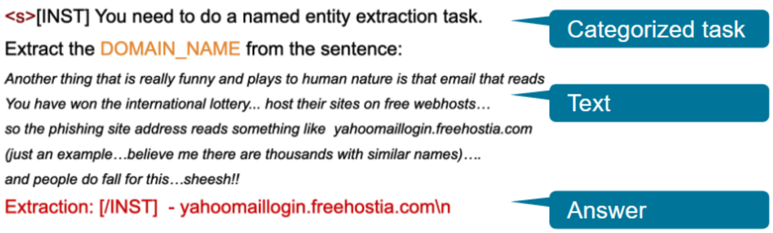
```
============================== Training pipeline ==========================
ℹ Pipeline: ['tok2vec', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE 
---  ------  ------------  --------  ------  ------  ------  ------
  0       0          0.00     44.94    0.00    0.00    0.00    0.00
  4     200        278.13   1783.97   14.49   21.74   10.87    0.14
 10     400        507.82    626.54   30.14   40.74   23.91    0.30
 17     600        170.62    323.26   26.09   39.13   19.57    0.26
 26     800        126.21    235.60   27.27   45.00   19.57    0.27
 37    1000         34.92    225.12   26.09   39.13   19.57    0.26
 51    1200         35.68    237.72   24.62   42.11   17.39    0.25
 68    1400         45.72    296.98   24.24   40.00   17.39    0.24
 89    1600         24.06    330.23   16.13   31.25   10.87    0.16
115    1800         27.23    404.82   22.22   41.18   15.22    0.22
147    2000         48.03    484.50   30.30   50.00   21.74    0.30
186    2200         28.83    564.93   29.85   47.62   21.74    0.30
234    2400         55.14    687.30   31.25   55.56   21.74    0.31
284    2600         24.17    711.34   28.99   43.48   21.74    0.29
334    2800         23.02    701.66   30.99   44.00   23.91    0.31
384    3000        337.68    706.06   27.69   47.37   19.57    0.28
434    3200         23.53    699.76   27.69   47.37   19.57    0.28
484    3400         80.52    699.72   34.29   50.00   26.09    0.34
534    3600         50.92    697.22   23.33   50.00   15.22    0.23
584    3800         36.28    695.88   31.88   47.83   23.91    0.32
634    4000         97.75    702.80   27.27   45.00   19.57    0.27
684    4200         26.31    694.97   30.77   52.63   21.74    0.31
734    4400         70.97    704.12   28.57   52.94   19.57    0.29
784    4600        144.54    710.68   29.51   60.00   19.57    0.30
834    4800        323.26    733.46   33.33   55.00   23.91    0.33
884    5000        140.97    692.76   32.35   50.00   23.91    0.32
```

Figure 7. Training Pipeline of SpaCy Model

Llama 2

We leverage Llama-2-7b-chat as our pre-trained generative text model for completing our NER task. This model has 7 to 70 billion parameters and is known to be optimized for dialogue use cases.

Input data format Llama 2 requires input data in a question prompt format, similar to ChatGPT. As shown in Figure 8, the input for Llama 2 should clearly specify the type of IoCs to be extracted. It must include the text from which the extraction is to be made, along with the correct answer. A key aspect of this approach is that each extraction task focuses on only one type of IoC. Therefore, we iterate through all IoC types to comprehensively extract various IoCs from the provided text.



<s>[INST] You need to do a named entity extraction task.
Extract the DOMAIN_NAME from the sentence:
Another thing that is really funny and plays to human nature is that email that reads
You have won the international lottery... host their sites on free webhosts…
so the phishing site address reads something like yahoomaillogin.freehostia.com
(just an example…believe me there are thousands with similar names)….
and people do fall for this…sheesh!!
Extraction: [/INST] - yahoomaillogin.freehostia.com\n

Figure 8. Input data for Llama 2

Experimental Setup And Outcome While this pre-trained model has more than 7 billion parameters, we can't fine tune this model in full due to resource limitation. Therefore, we decided to use QLoRA where Q stands for quantization of each parameter to be presented in 4 bit precision. LoRA allows us to tweak just a small part of relevant input when it makes predictions, making the training process quicker and less resource-intensive.

The learning curve in Figure 9 shows the model is learning as expected because both training and testing loss decrease over the steps. The blue line represents the loss on the training data and it's going down, which means the model is getting better at predicting or fitting the training data as it goes through more steps. The orange line is the loss on the testing data, which also generally trends downwards, indicating the model is improving on new, unseen data as well.

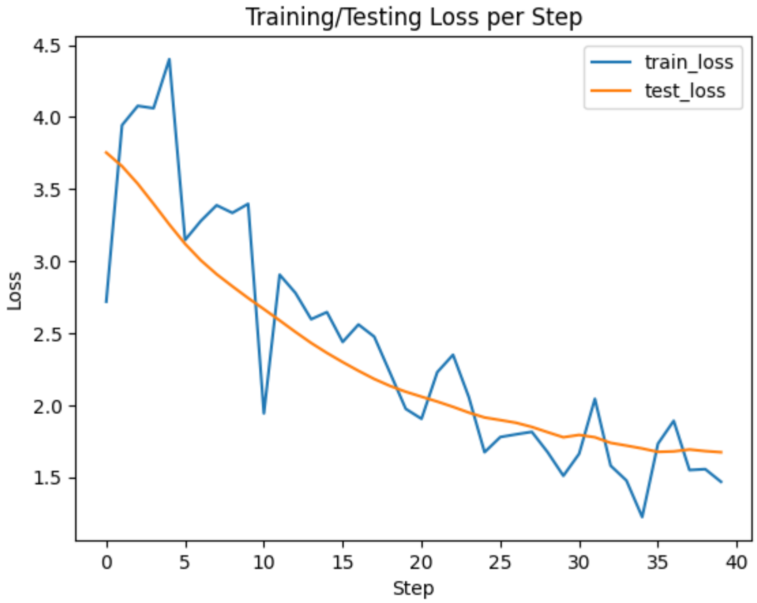
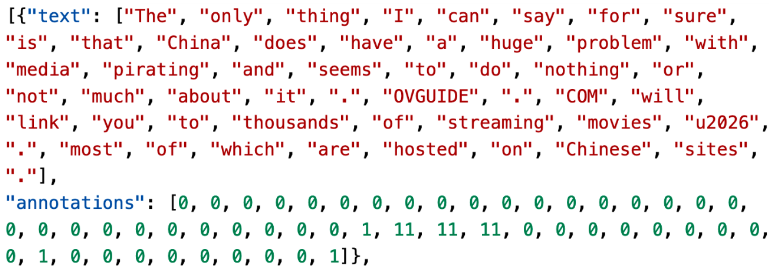


Figure 9. Learning Curve of Llama 2

Distilbert

We use a lighter and faster LLM DistilBERT-base-uncased as the pre-trained model for our NER model in a full fine-tune manner. DistilBERT-base-uncased is trained on a huge amount of text and is great for understanding the meaning of sentences and words in tasks like translation, question-answering. It is a smaller model compared to Llama 2.

Input data format The Distilbert model uses the BIO (Begin-Inside-Outside) format for data to train effectively. In this format, every word or punctuation is a token with a specific label. Non-IoC elements are marked as 0. An IoC at the start of a sentence or following a non-IoC token (labeled 0) is labeled as B-label_name. If it follows another IoC token (non-0), it is labelled as I-label_name. This format helps Distilbert to recognize IoCs in text, as illustrated in Figure 10.



```
[{"text": ["The", "only", "thing", "I", "can", "say", "for", "sure",
"is", "that", "China", "does", "have", "a", "huge", "problem", "with",
"media", "pirating", "and", "seems", "to", "do", "nothing", "or",
"not", "much", "about", "it", ".", "OVGUIDE", ".", "COM", "will",
"link", "you", "to", "thousands", "of", "streaming", "movies", "u2026",
".", "most", "of", "which", "are", "hosted", "on", "Chinese", "sites",
"."],
"annotations": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 11, 11, 11, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1]},
```

Figure 10. Input Data of Distilbert Model

Experimental Setup And Outcome The Distilbert model initially focuses on identifying one specific type of IoC in the early epochs (in Figure 11), gradually broadening its scope with progression. For example, by epoch 2, it specializes in recognizing DOMAIN_NAME, and by epoch 5, it also starts to identify MALWARE_NAME and IP. The performance of Distilbert, detailed in the results comparison section, is based on evaluations conducted after 50 epochs.

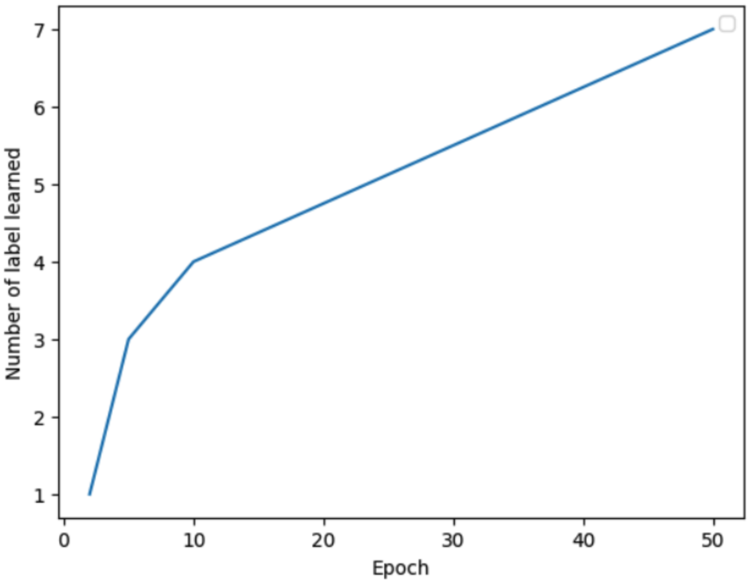
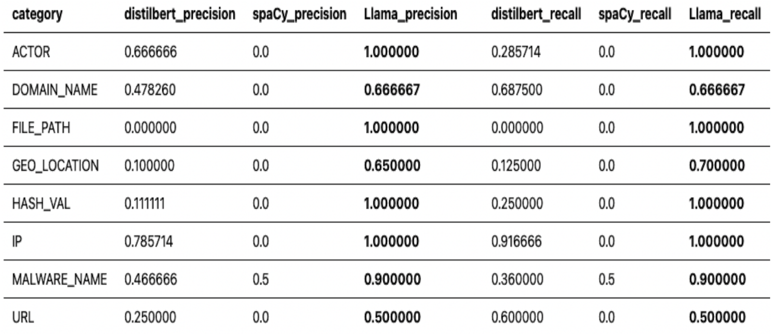


Figure 11. Number of label learned of Distilbert

OBSERVATIONS AND DISCUSSIONS

Quantitative Results

Table 2 presents the quantitative results of three models across eight IoC categories, focusing on precision and recall. The Llama model excels in identifying all IoC categories, achieving 100% precision in actor, file path, hash value, and IP, and 50% in its weakest area, URL prediction. The distilbert model shows strength in most categories, particularly with 78% precision in IP, but struggles with file path IoCs. In contrast, spaCy is limited to the malware name category, reaching 50% accuracy, and underperforms in other IoC categories, indicating its comparative limitations against Llama and distilbert.

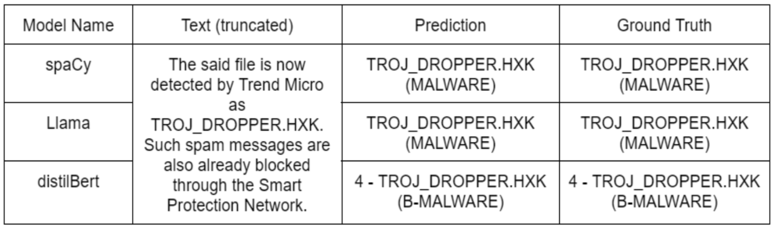


category	distilbert_precision	spaCy_precision	Llama_precision	distilbert_recall	spaCy_recall	Llama_recall
ACTOR	0.666666	0.0	**1.000000**	0.285714	0.0	**1.000000**
DOMAIN_NAME	0.478260	0.0	**0.666667**	0.687500	0.0	**0.666667**
FILE_PATH	0.000000	0.0	**1.000000**	0.000000	0.0	**1.000000**
GEO_LOCATION	0.100000	0.0	**0.650000**	0.125000	0.0	**0.700000**
HASH_VAL	0.111111	0.0	**1.000000**	0.250000	0.0	**1.000000**
IP	0.785714	0.0	**1.000000**	0.916666	0.0	**1.000000**
MALWARE_NAME	0.466666	0.5	**0.900000**	0.360000	0.5	**0.900000**
URL	0.250000	0.0	**0.500000**	0.600000	0.0	**0.500000**

Table 2. Quantitative Results of Three Model

Qualitative Results

Table 3 is an example of qualitative results, The table compares the performance of three models—spaCy, Llama, and distilbert—in extracting malware names from text. All models successfully identified the malware name ”TROJ_DROPPER.HXK” as an IoC (Indicator of Compromise). spaCy and Llama provided the correct label ”MALWARE” without specifying the position in the text. distilbert also correctly identified the malware name, but additionally specified its position with a ”B-” prefix, indicating the beginning of the entity, aligning with the BIO (Begin-Inside-Outside) tagging format used in NER (Named Entity Recognition). This demonstrates that each model is capable of accurately extracting malware names, with distilbert providing more detailed positional information.



Model Name	Text (truncated)	Prediction	Ground Truth
spaCy	The said file is now detected by Trend Micro as TROJ_DROPPER.HXK. Such spam messages are also already blocked through the Smart Protection Network.	TROJ_DROPPER.HXK (MALWARE)	TROJ_DROPPER.HXK (MALWARE)
Llama		TROJ_DROPPER.HXK (MALWARE)	TROJ_DROPPER.HXK (MALWARE)
distilBert		4 - TROJ_DROPPER.HXK (B-MALWARE)	4 - TROJ_DROPPER.HXK (B-MALWARE)

Table 3. Qualitative Results of Three Model

Discussion And Learning

The mentioned results show that Llama outperforms distilbert, with spaCy being the least effective. Conclusively for our task, larger models tend to perform better with the same data. However, all models have difficulty differentiating between domain names and URLs due to the similarity between those two types and inaccurate labeling may also contribute to the noise for this category. This confusion is particularly notable when using chat-GPT4 for labeling, which can lead to undetected errors and dataset contamination. Additionally, despite GEO_LOCATION being a common category, the models often incorrectly extract all geo-locations instead of just the one relevant to the attacks, which is our primary interest. This project highlights several important lessons. First, the training data's format is crucial, different models require different formats of input data, and data format can impact model training effectiveness. Second, the extraction cannot perfectly align with human intention. Such as identifying all geographic locations instead of just the attack's related. Lastly, the project's success is highly dependent on data quality and quantity. Limited data is a major factor in model performance and evaluation.

INDIVIDUAL CONTRIBUTIONS

Xuehan Liu in the following table 4 is addressed by Liu and Wenxuan Zhang is addressed by Zhang. Though not mentioned in report, but UI is implemented with reference of https://github.com/ece1786-2023/ThreatX/blob/main/1786gradioProj.ipynb

Item	Assignee
Raw Data Acquisition	Liu
Data Labelling	Liu
Data Formatting	Zhang
Baseline Model	Liu And Zhang
Fine-tune Model - Llama	Liu
Fine-tune Model - distilBert	Zhang
Gradio UI	Zhang

Table 4. Project contribution.

Member	Post Video?	Post Report?	Post Code?
Xuehan Liu	No	Yes	Yes
WenXuan Zhang	Yes	Yes	Yes

Table 5. Project permission.

PERMISSIONS

REFERENCES

Gao, P., Liu, X., Choi, E., Ma, S., Yang, X., Ji, Z., Zhang, Z., and Song, D. (2022). Threatkg: A threat knowledge graph for automated open-source cyber threat intelligence gathering and management. *arXiv preprint arXiv:2212.10388*.