LEARNING REPRESENTATIONS OF OPERATING SYSTEM LEVEL DATA AND DESIGNING INTERPRETABLE AI FOR DEFENSIVE CYBERSECURITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Frontier models have developed the emergent capability to formulate cybersecurity exploits and use them agentically with many recent high profile incidents. In large part due to these incidents, many surveys indicate a majority of Americans now think of AI as an existential safety risk. With frontier models speeding up the formulation and execution time of exploits beyond human reaction time in multiplicative effect with increasing the scale of attacks through agentic swarms, the value of high precision detection of exploits as a defensive counterbalance becomes greater; both to safeguard valuable computing systems and to allow the larger field of Artificial Intelligence to safely advance. Toward these ends, vector representations of cybersecurity exploits were learned on operating system level data, including command lines and executables used in host operating systems. These representations were trained on by interpretable ensemble machine learning approaches to make interpretable detectors. Some of the detectors had greater than 90% precision when tested on reserved data, even when the targets were a minority class with other logs dwarfing the numbers of true positives by orders of magnitude. This high precision for detecting minority classes in large swaths of operating system level data is very valuable in defensive cybersecurity because most networks on average generate thousands of logged events per computer per day with many false positives using up responders' time thereby delaying responses to and often completely burying the true positives. In the new era of AI and cybersecurity where agentic AI swarms are conducting offensive cyberattacks faster than current defensive systems can respond, in large part due to the lower precision and lack of interpretability of previous systems, it seems imperative for safety to use AI, and preferably interpretable AI, defensively as a counterbalance.

1 INTRODUCTION

Cybersecurity exploits are constantly evolving and new ones being generated, the rate of both of which have been sped up by generative A.I.; therefore, it is necessary to have some framework and grouping system to communicate the nature of the exploit. This is what the MITRE ATT&CK framework, as described in Strom et al. (2018), does. The "why"s, or the broad goal of an attack is characterized by different tactics in the framework, with hundreds of different techniques and sub-techniques, which can be considered the "how"s.

In order to build a machine learning based cyberattack detector, it is good to have labeled instances. Splunk Threat Research Team (2026) has helpfully made logs of red-teaming attacks labeled with MITRE ATT&CK techniques and sub-techniques public.

In terms of previous work building detectors, many private companies have detectors but they generally do not publish their detection rates and moreover not with the same Splunk Attack dataset as a consistent benchmark. The closest comparable was Mărmureanu & Oprisa (2023), but it focused on predicting the broad tactic and not the technique. Therefore, this paper could act as a benchmark for future detection work.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 METHODOLOGY

Splunk Threat Research Team (2026) was used for reproducibility. The models were trained on the Integrity, User, ParentUser, Image (executable), ParentImage, CommandLine, and ParentCommandLine fields, which were extracted from the JSON and dictionaries in the Splunk Attack logs, and the technique and sub-technique folder the logs were in was considered the ground truth. It is important to note that the logs may contain extra events that may happen during an attack but are not part of the attack though, potentially causing some events to be mislabeled.

There was a limit of 100 labels for Pyspark's implementation of RandomForestClassification, so the techniques were indexed and techniques with an index of 100 or greater dropped.

Notably, rows with NULLs in the feature columns were not dropped. The thinking behind this was to train a detector resilient to NULLs, but this would also have the effect of lowering the detectors ability to discern between techniques compared to keeping only entries with non-NULL data.

80:20 train:test splits were used, first with random splits and later with stratification to more consistently cover minority classes.

When indexing, vectorization, or calculations of conditional probabilities were done, they were fit on the training dataset, then the frozen models or probabilities were used, after model fitting, on the test set. This was done to ensure there would be no information leakage from the test set before model fitting.

The optimization metric was the average of the F1 scores between the different techniques. Four fold cross validation was used.

The reported results are those on the reserved test set.

3 EXPERIMENTS

3.1 SETUP

Integrity, User, ParentUser, Image, and ParentImage were indexed to positive integers.

CommandLine and ParentCommandLine were tokenized by spaces and vectorized into vectors of 300 dimensions using Word2Vec.

Pyspark's RandomForestClassifier was then fit with 100 trees with maxDepth of 30.

3.2 RESULTS

Table 1: MITRE ATT&CK technique detection precision and recall on reserved 20% of Splunk Attack data

Description	**Technique**	**Precision** %	**Recall** %
Scheduled Task	T1053	24	89
Process Injection	T1055	65	0.05
Scripting Interpreter	T1059	55	1
Binary Execution	T1218	55	0.2
Server Software Component	T1505	21	94
Event Triggered Execution	T1546	14	0.08
Boot Autostart Execution	T1547	85	0.04
Abuse Elevation Control	T1548	60	0.1
Impair Defenses	T1562	43	3
Phishing Attachments	T1566	50	0.1
Protocol Tunneling	T1572	83	100

4 CONCLUSION

While there is still much room for improvement for detectors of hundreds of techniques, it was found that some could be detected with high precision using indexes of the Integrity, User, ParentUser, Image, and ParentImage, and the CommandLines and ParentCommandLines represented as vector embeddings of 300 dimensions.

5 FUTURE WORK

An alternative experiment was setup where Image and ParentImage where instead vectors of 15 dimensions with each dimension being the probability of the Image or ParentImage being in an entry with a technique within one of the 15 MITRE ATT&CK tactics. The idea behind this was to give the vectors interpretable meaning correlating with the MITRE ATT&CK tactics 'Reconnaissance', 'Resource Development', 'Initial Access', 'Execution', 'Persistence', 'Privilege Escalation', 'Stealth', 'Defense Impairment', 'Credential Access', 'Discovery', 'Lateral Movement', 'Collection', 'Command and Control', 'Exfiltration', and 'Impact'. Similiarly, each token in the CommandLine and ParentCommandline was turned into a vector based on the correlation between the command token and the technique of that entry being within an tactic group. The vectorized command tokens were then added. These vector representations were then to be put into an Explainable Boosting Machine. Unfortunately, as of the submission time of this paper, the model was still fitting and notably had been for hours. While this suggets such a model may be hard to train, there is hope that it will be more interpretable and possibly have higher Area under the Precision - Recall curve.

AI USE STATEMENT

In this work, we used generative AI tools to implement some methods for converting textual data to floats. We have not used generative AI tools for generating synthetic data sets, help developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, to design research methodology or experiments, assist with translation, clear or reformat datasets, supporting qualitative and thematic data analysis, interpreting the results, research idea origination or selection, the original Random Forest detector implementation, and writing this paper and formulating questions for surveys or interviews are not applicable to this work. Additionally, we used generative AI tools for sourcing/searching for information, identifying relevant literature, and formatting references. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

This paper was made with reproducibility in mind, with the data (Splunk Attack) in the public domain and the algorithms used described in the Methods section.

REFERENCES

Marius Mărmureanu and Ciprian Oprisa. Mitre tactics inference from splunk queries. *2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing*, pp. 277–283, 10 2023. doi: 10.1109/ICCP60212.2023.10398612.

Splunk Threat Research Team. Splunk attack data: A repository of curated datasets from various attacks. `https://github.com/splunk/attack_data`, 2026. Accessed: 2026-09-25.

Blake Strom, Andy Applebaum, Doug Miller, Kathryn Nickels, Adam Pennington, and Cody Thomas. Mitre att&ck: Design and philosophy. *The MITRE Corporation*, 2018.