Context-Aware Attack Classification: Using Temporal Object Allocation Trace

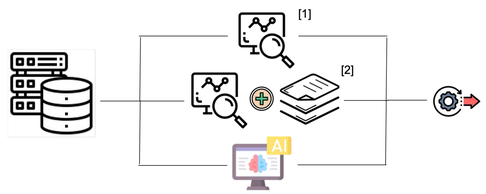
Nicholas Di(nd56), Kimia Saedi(ks152), Claire Huo(ch107)

Abstract

Our work introduces an advanced detection method leveraging Long Short-Term Memory (LSTM) for identifying security vulnerabilities within the Linux kernel. Focusing on low-level context analysis and detailed per-process runtime behavior, it is meticulously designed to detect a broad spectrum of security threats while maintaining minimal impact on system performance.

Motivation

The Linux kernel's increasing complexity has amplified its exposure to sophisticated cyber threats. Traditional mechanisms, such as static analysis[1], are ineffective against dynamic runtime exploits due to their inherent static nature. Dynamic taint analysis[2], while more adaptive to runtime conditions, suffers from significant limitations like high false positive rates and is not scalable to complex kernel environments. These traditional methods also struggle to fully comprehend the intricate interactions and behaviors within kernel components, leading to inefficient and often inadequate threat detection capabilities.



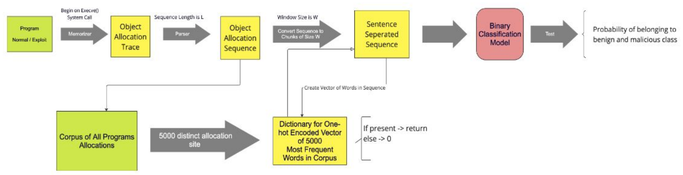
To overcome these substantial shortcomings, our research introduces a novel approach utilizing LSTM. This method is designed to offer a real-time analysis of kernel activities, which significantly enhancing the detection accuracy of security threats while maintaining optimal system performance. We pick 5 real-world CVEs collecting 23 unique kernel operations to form 265 traces to train and test our model.

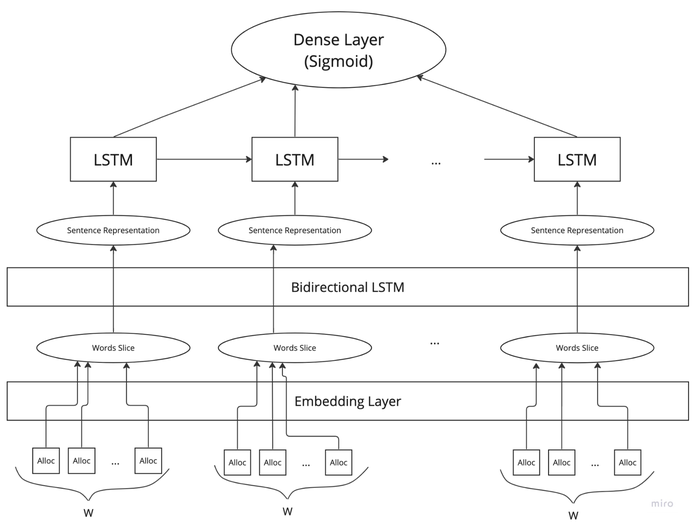
References

- [1] Chess, Brian, and Gary McGraw. "Static analysis for security." *IEEE security & privacy* 2.6 (2004): 76-79.
- [2] Newsome, James, and Dawn Xiaodong Song. "Dynamic taint analysis for automatic detection, analysis, and signaturegeneration of exploits on commodity software." *NDSS*. Vol. 5. 2005.
- [3] Roessler, Nick, et al. "Lossless instruction-to-object memory tracing in the Linux kernel." Proceedings of the 14th ACM International Conference on Systems and Storage. 2021.

Model

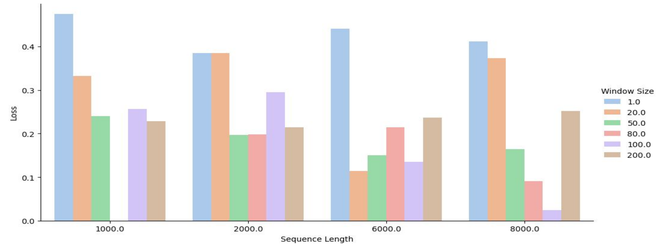
- Memorizer[3] is a tool designed to track object allocations within the Linux kernel. It captures intricate details such as the process name, memory address, and size, storing this information in a file to aid in the identification of abnormal system behavior.
- The underlying model for analyzing this data hinges on the assumption of consistent allocation sites across different environments. To achieve this, a global hashmap is employed to assign unique identifiers to each site. The input to the model consists of a sequence of object allocations, each represented by a unique number encoded in a one-hot vector. The model possesses the capability to adjust the sequence length to effectively detect delayed malicious activity.
- A significant enhancement to the model is the incorporation of an embedding layer. This layer transforms the one-hot vectors into denser, context-rich vectors, thereby improving efficiency.
- The model's hidden layer utilizes a two-stage LSTM network, processing data in smaller segments to grasp the context and relevance of each part. This approach bolsters the model's ability to classify security threats accurately. The overall method of mapping the input into embedded vectors is illustrated in the first figure. The second figure shows the architecture of the simple classification model utilizing LSTM and sliced sequence representation.

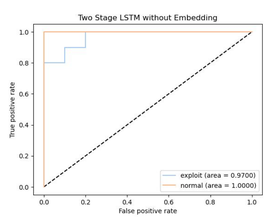


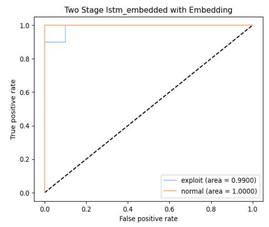


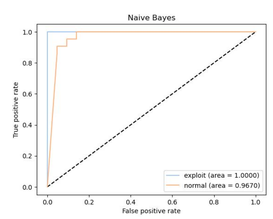
Results

- We evaluated the performance of our proposed model using three metrics: receiver operating characteristic (ROC) curve analysis, the area under the curve (AUC).
- We collected multiple traces for all the programs to build a robust model. For our experiment, we traced 23 common Linux commands, i.e., ls, route, ps, etc., as normal samples. For exploit samples, we use the proof of concept of 5 distinct CVEs: CVE-2022-0185, CVE-2022-0847, CVE-2022-2639, CVE-2021-3490, CVE-2021-31440. Overall, we obtained 215 normal traces and 50 exploit traces and used 80% of the whole traces for training and the rest 20% for testing. Among the 20% test samples there are completely new types of attack and benign program which are not learned in the training set.
- We benchmarked our model against Naive-Bayes.









Conclusion

In conclusion, understanding the behavior of a running program and identifying potential malicious activities is challenging due to the multitude of events occurring within seconds of program execution. Our approach introduces a straightforward LSTM model that leverages the sequential nature of runtime events for effective classification. Through careful selection of detailed features, their transformation into a condensed vector representation, and fine-tuning of model parameters, we achieved an outstanding accuracy of %99. These results highlight the success of our method in accurately representing program behavior and detecting malicious patterns.