Data Driven Malware Detection

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Andrew Rocco Gagliostro

5 May, 2021

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Andrew Rocco Gagliostro

Capstone advisor: Tian Y. Professor, Department of Computer Science

Motivation for Research

Although vulnerability detection is one of the most pressing challenges in software security, developers are still required to identify and patch vulnerabilities through processes that are mostly manual and extremely time consuming (Chen, 2017).

There have been many different techniques proposed to help increase the efficiency and accuracy of vulnerability detection: white-box symbolic execution, black-box random fuzzing, and path-sensitive software analysis are just a few of them (Salimi, 2020). These techniques have proven to be effective to some degree in controlled environments, but are not robust enough to efficiently identify vulnerabilities in real-world scenarios. Without further development, they lack the accuracy and scalability required to perform any effective vulnerability detection in complex code.

There are clear limitations of current vulnerability detection techniques; however, the growing importance of software, and inherently software security, has led to a spike in the number of vulnerabilities detected in real-world applications in recent history. In turn, we now have much more data and information at our disposal: this can provide us with insights regarding the nature of different vulnerabilities in unique contexts (Salimi, 2020). These insights can be used to better pinpoint zero-day vulnerabilities in large sets of code.

Along with these insights have come new initiatives: learning as much as we can from existing vulnerabilities to understand how to convert programs into vectors. The goal is to use these vectors with machine learning frameworks in order to automatically detect vulnerabilities in code. One of the only known implementations of such a framework is called SySeVR (Li, 2021). This framework introduces the use of Syntax-based Vulnerability Candidates (SyVCs) and Semantics-based Vulnerability Candidates (SeVCs), which are portions of the code that

reflect syntax-based vulnerability features as well as associated semantic-based (data and control dependency) vulnerability features. The SySeVR Framework shows us that it is possible to use deep-learning to efficiently detect vulnerabilities, but is very limited in terms of the types of systems and vulnerabilities current implementations are able to handle.

The goal of our work is to extend the above research and develop a more comprehensive and robust deep-learning program to detect zero-day vulnerabilities in real-world code.

Challenges

There is no existing tool or program that is able to accurately and efficiently detect vulnerable code in any software or namespace. There are two important limitations in the functionality of existing vulnerability detection tools that explain why there is not yet an answer to such a pressing issue (Grieco, 2018).

First of all, current VDTs use a significant amount of approximation in their evaluation of the program context. They are not able to understand or model vulnerable behavior in complex programs: this results in issues with slow execution and low success rates. Current VDTs are also unable to understand where a vulnerability is likely to occur in a program: this means that they need to evaluate all possible program states when looking for bugs. As a result, current techniques lack scalability and are therefore unable to process programs with large amounts of code.

The SySeVR framework is a good step towards mitigating the above issues in current VDTs, but is only effective in the context that it was designed for. The current implementation is very limited in terms of the vulnerabilities it is able to detect, programming languages it works

with, and the syntactic/semantic information it uses for vulnerability detection. In turn, SySeVR is unable to perform in real-world scenarios.

These limitations of current VDTs along with SySeVR lead us to further investigate how we can use deep learning frameworks in our research. We are attempting to overcome the current challenges associated with automatically detecting malicious code in large, diverse sets of data.

Current Solution and Results

In our progress towards a solution to this problem, we have been studying the nature of vulnerabilities in TensorFlow: an open-source machine learning library. We chose to explore this machine learning framework for two reasons: it is widely used and it shares some vulnerabilities that are semantically similar to the SySeVR dataset.

We have been using a clustering approach on TensorFlow files to narrow down the potentially vulnerable code for further analysis. The purpose of this process, along with the analysis of all past reported vulnerabilities in TensorFlow, is to gather data regarding vulnerabilities in different namespaces to identify what domain knowledge is pertinent to effective vulnerability detection. We hope this information can help to develop an intelligent machine learning approach for vulnerability detection in any given namespace.

In terms of implementation, our current solution uses the SySeVR framework for static analysis of code snippets: this allows us to summarize the control and data dependencies of some given code. We then use this data, in the form of code “slices,” along with specific domain knowledge in order to vectorize the program slices in a way that makes sense for the specific namespace: Tensorflow in our testing.

With the program slices tailored to their specific domain and vectorized, we plan on using a standard graph neural network in order to determine if each code chunk should be classified as benign or vulnerable.

Potential Future Developments

We have been working towards using the limited but conceptually-sound SySeVR framework to detect vulnerabilities in complex, real-world situations. Our work with TensorFlow has been a manual effort in order to understand the nature of vulnerability detection in one specific library. We will then use that understanding along with the other tools discussed: taking in some code as well as domain knowledge specific to TensorFlow in order to accurately label vulnerable and benign codes.

Regardless of any existing tools, determining the necessary domain knowledge for a namespace and how it should be correctly translated into a set of vectors is no trivial task. Once we have a framework that can process complex libraries and identify vulnerable code through the use of existing tools and domain knowledge, we may then work to automate the process for collection and application of domain knowledge in the context vulnerability detection.

References

Chen, Y., Khandaker, M., & Wang, Z. (2017). Pinpointing vulnerabilities. *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. doi:10.1145/3052973.3053033

Grieco, G., & Dinaburg, A. (2018). Toward smarter vulnerability discovery using machine learning. *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*. doi:10.1145/3270101.3270107

Li, Z., Zou, D., Xu, S., Jin, H., Zhu, Y., & Chen, Z. (2021). Sysevr: A framework for using deep learning to detect software vulnerabilities. *IEEE Transactions on Dependable and Secure Computing,* 1-1. doi:10.1109/tdsc.2021.3051525

Salimi, S., Ebrahimzadeh, M., & Kharrazi, M. (2020). Improving real-world vulnerability characterization with vulnerable slices. *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering*. doi:10.1145/3416508.3417120