Federal State Autonomous Educational Institution for Higher Education
National Research University Higher School of Economics

Information Security

BACHELOR'S THESIS

Research project

"Hybrid Fuzzing of the PyTorch Framework"

Prepared by the student of group 191, 4th year of study,
Larionov-Trichkine Theodor Arsenij

Supervisor:
Petrenko Alexander Konstantinovich

Consultant:
Kuts Daniil Olegovich, ISP RAS

Moscow 2022

Contents

Annotation

Your annotation in English.

Аннотация

Ваша аннотация на русском языке.

Keywords

Hybrid Fuzzing, Program Security, Dynamic Analysis, PyTorch, AI Frameworks Security

1 Introduction

Software security is a growing concern in the modern world. With the rapid development of information technologies, the number and complexity of software systems have increased drastically. This has led to an increase in the number of software vulnerabilities as well as an increasing need for secure software development practices.

1.1 Memory Safety Vulnerabilities

Memory safety vulnerabilities are a particularly significant concern in software security. They refer to programming errors that can cause a program to access memory in unintended ways, potentially leading to system crashes, data leaks, or even full system compromise. Memory safety vulnerabilities are especially prevalent in large codebases written in memory-unsafe languages such as C and C++.

According to [4], for codebases with more than one million lines of code, at least 65% of security vulnerabilities are caused by memory safety issues in C and C++. The Chromium project security team also highlights the same point in their report [1]. This alarming statistic underscores the importance of addressing memory safety vulnerabilities in software development. Especially, for critical software systems, such as operating systems, web browsers, machine learning frameworks, and beyond.

1.2 AI and Security

In recent years, AI (Artificial Intelligence) has emerged as a key technology in many domains, including banking, healthcare, transportation, and more. With the rise of AI-powered applications, there is an increasing need for secure AI models and software systems that can withstand cyber threats, as these systems are often used to make critical decisions that affect human lives.

Of particular interest is the security of AI frameworks. Often, these systems are the foundation of AI applications. As such, vulnerabilities in AI frameworks can have a significant impact on the security of applications built on top of them.

One of the most popular AI frameworks is PyTorch [2]. PyTorch is an open-source machine learning framework developed by Meta (formerly Facebook). It is used by many companies and organizations, including Microsoft, Uber, Twitter, and more. Despite its popularity, PyTorch is not immune to security vulnerabilities, especially given that it is written in C++, a memory-unsafe language.

Considering the importance of PyTorch in the AI ecosystem, it is crucial to ensure that PyTorch is secure and robust against cyber threats.

1.3 Objective

Our objective in this work is twofold: to perform a comprehensive security analysis of PyTorch with the goal of detecting and addressing any memory safety vulnerabilities present in the framework, and to enhance our hybrid fuzzing tool, sydr-fuzz [3].

2 Hybrid Fuzzing

2.1 Dynamic Analysis

2.2 Symbolic Interpretation

2.3 Hybrid Fuzzing

3 PyTorch Fuzzing

3.1 Attack Surface Mapping

3.2 Fuzzing Harness Development

4 Hybrid Fuzzer Improvements

4.1 Scheduling Symbolic Pointers Modelling

4.2 Utilizing Debug Information to Improve sydr-fuzz

5 Results

5.1 PyTorch Bugs

5.2 1 in 25

5.3 Annotate

6 Conclusion

References

1. *Chromium project memory safety report.* https://www.chromium.org/Home/chromium-security/memory-safety/. Accessed: 2023-02-12.

2. Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32.* Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

3. Alexey V. Vishnyakov et al. "Sydr: Cutting Edge Dynamic Symbolic Execution". In: *CoRR* abs/2011.09269 (2020). arXiv: 2011.09269. URL: https://arxiv.org/abs/2011.09269.

4. *What science can tell us about C and C++'s security.* https://alexgaynor.net/2020/may/27/science-on-memory-unsafety-and-security/. Accessed: 2023-03-14.