Chapter 12
Source Code Analysis With Deep Neural Networks

Rebet Keith Jones
https://orcid.org/0009-0008-0487-1301
Capitol Technology University, USA

ABSTRACT

In recent years, deep learning techniques have garnered considerable attention for their effectiveness in identifying vulnerable code patterns with high precision. Nevertheless, leading models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks require extensive computational resources, resulting in overhead that poses challenges for real-time deployment. This study presents VulDetect, an innovative transformer-based framework for vulnerability detection, developed by fine-tuning a pre-trained large language model (GPT) on a variety of benchmark datasets containing vulnerable code. Our empirical analysis demonstrates that VulDetect achieves a vulnerability detection accuracy of up to 92.65%, surpassing SyseVR and VulDeBERT, two of the most advanced existing techniques for identifying software vulnerabilities.

1. INTRODUCING

Cybersecurity aims to safeguard systems from increasingly sophisticated cyber-attacks, a growing challenge as businesses become more interconnected through technological advancements. The ability of organizations and individuals to defend against such widespread attacks is being strained, as highlighted in the 2023 Verizon Cost of Data Breach Report, which shows that companies take an average of

DOI: 10.4018/979-8-3373-0588-2.ch012

197 days to detect a security breach and 69 days to contain it. This delay exposes organizations to significant financial and operational risks, unscheduled downtime, and reduced productivity. Consequently, the need for automated systems capable of processing and analyzing vast volumes of language data—particularly for applications like software vulnerability detection—has become essential. Traditional methods of vulnerability detection, which depend on human experts to define vulnerabilities, are time-consuming and resource-intensive. Machine learning techniques, particularly Natural Language Processing (NLP) models like CodeBERT, present a more efficient alternative, enabling faster and automated detection of software vulnerabilities without extensive feature engineering (Healthcare, Business, & Technology, 2024; Basharat & Omar, 2024; Zangana, 2015).

Knowledge distillation (KD) is a technique that reduces neural network sizes for execution on devices with limited computational resources. KD involves training a smaller, lightweight model (the "student") to replicate the outputs of a larger model (the "teacher"), thus transferring the high performance of the teacher to a more compact architecture (Ahmed et al., 2023; Al-Sanjary et al., 2018; Arulappan et al., 2023). Remarkably, the student model may even surpass the teacher model in performance due to the "dark knowledge" embedded within the teacher's learned representations, such as insights into class similarities. This knowledge, often hidden in the teacher's outputs, can be more effectively utilized by the student than by the original labels (Dawson et al., 2019; Zangana, 2013; Zangana & Abdulazeez, 2023). As depicted in Figure 1, the student model is trained to mirror the teacher model's behavior, capturing its knowledge to achieve similar or even superior accuracy. The following section provides a comprehensive overview of the framework and components of knowledge distillation.

Transformer-based models like GPT-2 offer significant advantages for vulnerability detection, including improved accuracy and advanced natural language processing capabilities. These models reduce the need for manual inputs in static analysis tools, streamlining and automating the detection process (Basharat & Omar, 2024; Gholami & Omar, 2023; Jones & Omar, 2023). The proposed system, VulDetect, leverages the capabilities of Large Language Models (LLMs) such as GPT-2 to detect vulnerabilities in C and C++ source code. Specifically, VulDetect utilizes the power, speed, and precision of LLMs for software vulnerability detection. By leveraging benchmark datasets and a GPT-based large language model, VulDetect effectively identifies vulnerable code in multiple programming languages, including C/C++ and Java. Our empirical results demonstrate that VulDetect outperforms two leading state-of-the-art techniques in software vulnerability detection (Hamza & Omar, 2013; Huff et al., 2023; Zangana & Zeebaree, 2024).