Research Statement

Yue Duan
School of Computing and Information Systems, Singapore Management University
Tel: (65) 6826-4807; Email: yueduan@smu.edu.sg
18 Dec, 2023

Background

My research interests mainly lie in computer security, deep learning, and blockchain. As computer systems (e.g., mobile devices, IoT devices) and software have been deeply integrated into our daily lives, security has become a major concern. My goal is to make computer systems and software more secure by combining various static and dynamic program analysis techniques with other techniques, such as deep learning, model checking, and natural language processing (NLP). My research has enabled novel techniques to improve the security of systems and software in different domains, including computer systems, mobile devices, and blockchain, and has led to new techniques, algorithms, and important discoveries in the field. The following sections elaborate on my research in three areas and conclude with a future plan.

Research Areas

1. Blockchain Security

Smart contracts in blockchains are holding cryptocurrencies and tokens valued at billions of USD. One of its core applications is to support a variety of decentralized applications (DApps), such as NFT and DeFi. I have investigated two topics in this area: DApps safety vetting and upgradeable smart contract study.

VetSC [3]. Very little work has been done to target high-level semantic issues (e.g., unfair business models) in DApps. Existing techniques suffer from low efficiency and require extensive manual effort to create specific policies for each contract by understanding the source code. Hence, they are highly impractical. To overcome the limitations, we designed VetSC [5], a novel UI-driven, program analysis-guided model-checking technique that can automatically extract contract semantics in DApps and perform safety vetting. From the smart contract code, we extract business model graphs that capture its intrinsic business and safety logic to facilitate model checking. We further leverage textual semantics from DApp user interfaces to help validate the UI-logic consistency and detect any discrepancies. We have successfully discovered 19 new safety risks in the wild, such as expired lottery tickets and double voting. This work won the Best Paper Honorable Mention award at ACM CCS'22.

UscHunt [2]. Upgradeable Smart Contract (USC) has become the future of smart contract development as it allows developers to perform upgrades on the contract logic. To fully understand the status quo of USCs in the wild and reveal their security implications, we conducted the first-ever systematic study on USCs in the real world. To achieve this goal, we developed UscHunt, an automatic smart contract analysis framework that supports accurate USC detection and detailed behavior analysis. To systematically characterize USCs, for the first time, we proposed a USC taxonomy that comprehensively describes the unique behaviors of USCs at both syntactic and semantic levels. We analyzed 800K+ smart contracts with UscHunt and discovered 2,500+ real-world security and safety risks. The paper was published in USENIX Security'23.

DiffUSC [6]. Smart contract upgrades can accidentally introduce serious vulnerabilities, causing millions of USD losses. To detect security issues brought about by upgrades, we proposed a novel system named **DiffUSC** that discovers behavioral discrepancies between two versions of a smart contract by performing differential fuzzing that focuses on testing the changed code. Once the discrepancies are extracted, we further perform postprocessing to select and report the most important discrepancies to the developers. In the evaluation, we showed that our tool was able to effectively and efficiently detect all four known vulnerabilities related to upgrades. The paper will be submitted to ACM CCS'24 in Jan 2024.

2. Binary Analysis

Researchers have utilized different binary analysis techniques to solve security issues. The two topics I investigated are binary differential analysis and vulnerability discovery.

DeepBinDiff [4]. Binary diffing quantitatively measures the differences between two given binaries and produces fine-grained basic block matching. It has been widely used to enable different kinds of critical security analyses, including security patch analysis and malware analysis. However, all existing techniques need more accuracy, better scalability, and coarse granularity or require extensive labeled training data to function. To address the limitations, we designed DeepBinDiff [2], an unsupervised program-wide code-representation-learning technique that relies on both the code semantics information and the program-wide control flow information to generate block embeddings. We further proposed a k-hop greedy matching algorithm to find the optimal diffing results using the generated block embeddings. The evaluation showed that DeepBinDiff outperforms the state-of-the-art by a large margin for both cross-version and cross-optimization-level diffing. The paper was published in NDSS'20.

SigmaDiff [1]. Furthermore, we invented a novel technique called **SigmaDiff** to perform binary diffing at the pseudocode level for better precision and human readability. To this end, we first perform diffing at the IR level and map the IR-level diffing results up to the pseudocode level. We construct interprocedural program dependency graphs and extract a symbolic expression for each IR to capture the semantic meanings of a binary. After that, we model the program-wide IR diffing problem as a graph-matching problem on two IPDGs from the two given binaries by leveraging the deep graph matching consensus (DGMC) model to fully exploit the neighboring contextual information. The paper has been accepted in NDSS'24.

3. Android Security

Due to its openness and increasing popularity, Android has become a new frontier for the perpetual battle between cyber-criminals and those who want to stop them.

DroidUnpack. Although powerful, malware detectors can be thwarted by Android packing, a technique that encrypts the code and only decrypts it at runtime. The packing technique has become a new barrier to malware detection, but no comprehensive study has ever been conducted to help the community understand the status quo. Therefore, we conducted the first systematic study on mainstream Android packers in an attempt to understand their security implications. To do so, we developed **DroidUnpack**, a whole-system emulation-based analysis framework, which relies on only intrinsic characteristics of Android runtime. Running our tool on a dataset of nearly 100,000 malware, we found that (1) commercial packing services are abused to encrypt malicious or plagiarized apps; (2) these packing services also introduce security-critical vulnerabilities that impact more than 800 million people; (3) a rapid evolution of custom packers renders current unpackers obsolete. Our findings led to a vulnerability discovery reward of $8,000 from Qihoo Inc. The paper was published in NDSS'18.

DroidVMUnpack. Recently, the emerging virtual machine-based Android packers rendered existing unpacking techniques, including **DroidUnpack** ineffective. To tackle this new problem, we propose to combine dynamic analysis with deep learning to automatically and accurately distill the semantics from machine code, to facilitate semantic-based malware detection. To correctly identify translated machine code, we resort to the inherent behaviors of VM dispatchers rather than imprecise code structures. To capture instruction semantics in a context-aware fashion, we model bytecode operations using comprehensive ARM instruction traces and Android-specific JNI calls. To precisely recover the correlation between the generated machine code and its original Dalvik bytecode, we invent a custom neural network model, based on Bi-LSTM, Attention mechanism, and RNN, that can systematically distill intrinsic semantics and contexts of individual Dalvik opcodes from their unique machine translations, while tolerating nuances caused by specific runtime environments and packer implementations. The project is still in progress.

Future Directions

I expect to continue my current research and expand the effort to solve other emerging security issues

1. **Directed Fuzzing Scheduling**

Unlike traditional grey-box fuzzing that explores whole program states, directed fuzzing takes as input a list of pre-defined potential vulnerable code locations and dedicates all its resources to directing the testing to these locations and triggering potential vulnerabilities. However, one big research question: how to effectively schedule these locations, remains to be answered. Therefore, we propose to model the fuzzing campaign using statistical methods and estimate a probability for each location, to which fuzzers could successfully exploit the vulnerability, to schedule directed fuzzing to work on more promising locations and utilize limited resources more effectively.

2. **IoT Security**

The security and privacy issues in IoT devices, ubiquitous in our lives, have become huge concerns for end users. Analyzing IoT devices face unique challenges in that they are heterogeneous by nature: people only have access to the mobile apps that interact with the IoT devices, but the firmware controls the devices. Therefore, hidden behaviors in IoT could exist and users may be completely unaware of them. I plan to use the NLP technique to extract semantic information from the GUI components in mobile apps and leverage Android app analysis and fuzzing techniques to identify hidden behaviors in IoT devices.

Selected Publications and Outputs

[1] [NDSS’24] Lian Gao, Yu Qu, Sheng Yu, **Yue Duan**, and Heng Yin. "SigmaDiff: Semantics-Aware Deep Graph Matching for Pseudocode Diffing", to appear in Proceedings of the 31st Network and Distributed System Security Symposium, February 2024.

[2] [USENIX Security'23] William E Bodell III, Sajad Meisami, and Yue Duan. "Proxy Hunting: Understanding and Characterizing Proxy-based Upgradeable Smart Contracts in Blockchains", in Proceedings of the 32nd USENIX Security Symposium, August 2023.

[3] [CCS'22] Yue Duan, Xin Zhao, Yu Pan, Shucheng Li, Minghao Li, Fengyuan Xu, Mu Zhang. "Towards Automated Safety Vetting of Smart Contracts in Decentralized Applications", in Proceedings of the 29st ACM Conference on Computer and Communications Security, November 2022.

[4] [NDSS'20] Yue Duan, Xuezixiang Li, Jinghan Wang, and Heng Yin. "DeepBinDiff: Learning Program-Wide Code Representations for Binary Diffing", in Proceedings of the 27th Network and Distributed System Security Symposium, February 2020.

[5] [NDSS'18] Yue Duan, Mu Zhang, Abhishek Vasist Bhaskar, Heng Yin, Xiaorui Pan, Tongxin Li, Xueqiang Wang, and Xiaofeng Wang. "Things You May Not Know About Android (Un)Packers: A Systematic Study based on Whole-System Emulation", in Proceedings of the 25th Network and Distributed System Security Symposium, February 2018.

[6] [To be submitted] William E Bodell III, Josselin Feist, Gustavo Grieco, and Yue Duan. “Differential Fuzzing of Smart Contracts in Solidity”