Simplifying Reverse Engineering of Software for Security Analysis

Steven H. H. Ding* Benjamin C. M. Fung* Philippe Charland+

*Data Mining and Security Lab, School of Information Studies
McGill University, Montreal, Canada
+Mission Critical Cyber Security Section,
Defence R&D Canada - Valcartier, Quebec, Canada

1 Research Problem

Figure 1. The binary analysis process.

Assembly code is one of the critical processes for detecting and proving software plagiarism and software patent infringements when the source code is unavailable. It is also a common practice to discover exploits and vulnerabilities in existing software systems. However, it is a manual and time-consuming process.

2 Challenges

Figure 2. An example of cloned subgraph.

However, this task is challenging. The number of assembly functions scales up to millions. As a clone search engine, it needs to achieve the following requirements:

- Scalable: the repository can index millions of assembly functions.
- Efficient: the average clone search response time should be around 1 second.
- Accurate: low false positive rate for query that has lots of results; high recall for query that has less results.

Figure 3. A shared repository.

3 Technical Details

The Kam1n0 engine is designed for general key-value storage and Apache Spark computation framework. Its solution stack consists of three layers. The data storage level and the computational level provide flexibility of deploying on a single workstation or on a cluster.

Figure 4. The solution stack.

4 Evaluations

We construct a new labeled one-to-many clone dataset by linking the source code and assembly code level clones.

Figure 5. The proposed new LSH and graph search algorithm.

We seek to design new data mining techniques that address the challenges of assembly clone search. We design an Adaptive Locality Sensitive Hashing algorithm to mitigate the imbalanced data distribution problem of traditional LSH. Moreover, we propose a new MapReduce algorithm to search for the cloned subgraphs.

Kam1n0 is open-source and available on GitHub. Scan the code to checkout Kam1n0 and the publication.