CS846-001 Week 10 Reviews

Jun Qing Lim
University of Waterloo
jq3lim@uwaterloo.ca

1 Mining Apps for Abnormal Usage of Sensitive Data

1.1 Problem Being Solved

The paper presents a static analysis tool to detect apps with malicious or suspicious features. Most of the existing malware apps detectors check an app against a list of malicious features or patterns and if a new malicious pattern is not known, the detector would fail. As such, the authors proposed this tool aiming to address the issue of problem in apps misusing sensitive data which results in security and privacy breaches, as some apps collect sensitive data and misuse it by sharing with third parties. The authors hope to develop a method to automatically detect abnormal pattern(s) in source code in handling sensitive data to improve the security and privacy of apps.

1.2 New Idea Proposed

The authors built a tool named MUDFLOW where it mines for patterns of "normal" data flow in the code and use the remaining mined patterns to detect suspicious behaviour. The process work as follow:

1. Extracts all data flows from the app (sensitive data sources)
2. Using the information extracted, it leverages the differences with normal apps and classify them if they are malicious apps or not in multiple categories and steps

The authors then used data of both benign and malicious apps to evaluate the performance of the tool and obtained the following findings:

- it recognizes 86% of malware with a false positive rate of 18.7%
- it recognizes 90.1% of malware leaking sensitive data with a positive rate of 18.6%
- in a sample of 96 repackaged apps (good apps + malicious behaviour), the tool identified 97% of those apps correctly as malicious
- using the data flow from sensitive sources result in best and accurate classification results

1.3 Positive Points

Firstly, the paper proposed a new approach of massive mining of collection of patterns data flow to identify malicious behaviour, this makes an impact to both the software and security field. Secondly, the proposed method in the paper has high practical impact as it is a static analysis tool that could be generalized for other range of apps (open / non-open source), making it a valuable tool in the enforcing security and privacy of an app. Thirdly, the paper gave a good discussion and evaluation of the proposed method, leaving little ambiguity of the ideas.

1.4 Negative Points

Firstly, the proposed tool may have some degree of computational overheads as it analyzes the trace execution of a source code and thus running this tool over a heavy app may be resource-intensive. Secondly, the method is only usable for Android apps as it was trained and evaluated over Android sources, limiting the generalization of the idea. Thirdly, the proposed tool is solely dependent on the data flow of the app, and did not consider the (logic of any possible runtime app), as such it may not handle/cover enough edge cases.

1.5 Future Work

One extension would be extending to other apps platform such as iOS, making it more generalizable for the larger apps community. Besides that, we could conduct an experiment or survey to determine the practicality and accuracy of the tool in actual practice/industry, giving a more realisitc representation of its usefulness.

1.6 Rating

4 out of 5. A well-written paper with a new method proposed.

1.7 Discussion Points

- What are some common popular ways of mishandling sensitive data?
- Any ethical concerns of this method?
- What are the possible future research directions for this paper?

2 Software Bertillonage: Finding the Provenance of an Entity

2.1 Problem Being Solved

The paper proposes a technique to identify the origin of a software entity in a software repository. Often the origin and evolution of software entities are not clearly stated, leading to uncertainty on a number of technical and ethical concerns. Software repositories also contain large amount of code, which are created and modified over a long time, making it difficult to understand the origin of a software entity. The authors hope that through the use of this proposed method, it could provide better understanding in the provenance of a software in faciliating and addressing software-related issues.

2.2 New Idea Proposed

The authors first adopted the general concept of software Bertillionage in reducing the search space of locating a software entity. It then used anchored signature matching technique to determine the version of a file within a larger repository with the following approaches:

1. extracting the class signature (metadata of a class) from *source code*
2. extracting the class signature from *bytecode* (compiled file information)

The authors then used matching techniques to match a desired artifact to determine its similarities. This method is then evaluated through a case study of 84 open source libraries and the following findings were obtained:

1. the similarity index in finding original *binary archives* matches up to 96%
2. the similarity index in finding original *source archives* matches up to 68%

2.3 Positive Points

Firstly, the author used an amazing idea of Bertillionage approach in finding the origin of a software entity, based on the metadata of software entities and matching with other artifacts, which is a great novel contribution. Secondly, the paper was clearly written with organized presentation of its ideas and implementation, making readers easy to understand regardless of its software expertise. Thirdly, the paper did not just propose a novel approach, but also discussed how the proposed method could be expanded for future research, laiding out a good foundation for “Bertillionage-based" software matching.

2.4 Negative Points

Firstly, the evalutation of the proposed method was limited to a single application/repository of an e-commerce, as such it is unclear on how useful this method would be on a larger complex application/repository. Secondly, the paper lacks thorough discussion or comparison with other existing methods of software provenance analysis, making it difficult to assess the relative strengths of the available techniques.

2.5 Future Work

We could extend this project by doing a comprehensive survey/comparison with other approaches in terms of effectiveness, accuracies, and generalizability. On top of that, we could also incorporate machine learning techniques to improve the effectiveness and accuracy of the approach.

2.6 Rating

3 out of 5. Although written thoroughly, it lacks a clear discussion on the objective of the research: “Why do we need to find the origin of a software?”, “What will we get if we determined the origin of a software?”.

2.7 Discussion Points

- What are some real challenges in industry pertaining to software origins?
- Comments about the proposed technique?
- What other alternative methods (or tools) do we have to determine the origin of a software?