Project proposal

CS 858 | **Alexey Zhikhartsev** | **Zijin Li**
Prof. Urs Hengartner | azhikhar@uwaterloo.ca | z542li@uwaterloo.ca

Developing search spaces for automatic repair of vulnerabilities in mobile software

In the world of countless software bugs and limited resources to fix them, automatic software repair tools are a desirable instrument that can decrease the developers' workload. Current automatic repair tools aim at fixing simpler bugs and thus give developers the chance to concentrate on more sophisticated bugs. One of the important subsets of all the bugs is software vulnerabilities, as they can lead to substantial financial losses and great customers' dissatisfaction. Moreover, software vulnerabilities of mobile phones are of a particular importance due to the ubiquity of mobile apps. Several automatic software repair tools were proposed [4, 6, 8] that are able to fix some subsets of bugs (including software vulnerabilities). One of the most common approaches is the co-called Generate-and-validate (G&V) approach, in which the target project has a bug that is exposed in some failing tests; the automatic repair tool goes through the list of potential fixes (*the search space*), applies the fix and runs the tests; if all the tests pass, then the fix is considered to be the correct fix and it is presented to the developer.

Obviously, if the correct fix is not in the search space of this particular tool, then it cannot be found. In addition, there is a challenge of choosing a useful search space: if a search space is rich (i.e., it contains more correct patches), then most likely it is also large (i.e., it also contains many more plausible patches—incorrect patches that manage to pass the tests); plausible patches can precede the correct patch in the search space, so the correct patch will never be found [7]. That is why it is important to devise "targeted" precise search spaces that contain more correct patches and fewer incorrect patches.

The goal of the current project is to create promising search spaces for fixing software vulnerabilities in mobile software. The work on the project will consist of the following stages: (1) mining vulnerabilities of mobile software from open-source repositories; the information of interest is the description of vulnerabilities, their category and the source code of the fixing patch; (2) manual analysis of the retrieved patches; (3) devising the search spaces that contain correct patches for the analyzed vulnerabilities; and (4) evaluating the search spaces in terms of how many correct patches they contain and what is the density of correct patches compared to the total number of patches. The result of our project should be a set of targeted search spaces that aim at repairing mobile vulnerabilities. To the best of our knowledge, there are no prior work of building G&V automatic repair tools specifically for fixing *vulnerabilities* in *mobile* software; this project will be the first step in building such a tool.

We already successfully mined the required data from the Android vulnerability database [1]; this database contains 636 vulnerabilities in both application and system software of Android. The next step would be analyzing the data, constructing search spaces and subsequent evaluation of the search spaces. If the time allows, we plan to expand this data with mobile vulnerabilities from the National Vulnerability Database [2].

Related work. Automatic repair of specifically mobile software is its infancy; the most related work is by Azim et al. [3], their approach detects programs' crashes, immediately patches the bytecode of the program (to be certain that it won't crash again) and rolls the program state to

the nearest activity. Automatic repair of desktop applications received more attention; the tool Prophet [5] is a state-of-the-art technique that prioritizes potential fixes inside the search space by learning what correct *developer* fixes look like. This direction is promising, however it highly depends on the data to learn from, which is not always present. Our approach of targeted search spaces is orthogonal to the one of Prophet.

References

[1] Android security bulletins, 2016. http://source.android.com/security/bulletin/.

[2] National vulnerability database, 2016. https://nvd.nist.gov/.

[3] M. T. Azim, I. Neamtiu, and L. M. Marvel. Towards self-healing smartphone software via automated patching. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 623–628. ACM, 2014.

[4] C. Le Goues, M. Dewey-Vogt, S. Forrest, and W. Weimer. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 3–13. IEEE, 2012.

[5] F. Long and M. Rinard. Prophet: Automatic patch generation via learning from successful patches. 2015.

[6] F. Long and M. Rinard. Staged program repair in spr. 2015.

[7] F. Long and M. Rinard. An analysis of the search spaces for generate and validate patch generation systems. In *Proceedings of the 38th International Conference on Software Engineering*, pages 702–713. ACM, 2016.

[8] S. Mechtaev, J. Yi, and A. Roychoudhury. Angelix: Scalable multiline program patch synthesis via symbolic analysis. In *Proceedings of the 38th International Conference on Software Engineering*, pages 691–701. ACM, 2016.