Symbolic Execution of Shell Scripts

Mihaela Sighireanu

December 13, 2016

Shell scripts are programs that are typically relatively short and follow a simple algorithmics. They act on the current state of the operating system: they create, remove or rename files, start or stop services, put bits of information into configuration files, etc. Scripts are everywhere on UNIX machines. A typical example are *maintainer scripts* which are executed during the installation, removal, or upgrade of software packages.

Incorrect scripts may have consequences that range from mildly annoying to catastrophic. For example, incorrect maintainer scripts may cause data loss by removing files that they are not supposed to touch, may make the installation fail, or may even lead the package management system into an inconsistent state. Furthermore, shell scripts must work correctly in a diversity of situations that probably have not been planned by the author of the script. For instance, maintainer scripts have to work correctly regardless of the choice of packages that are installed on the system. They also have to work correctly in abnormal situation, for instance when the installation is relaunched after a failure or an interruption by the system administrator. The formal verification of maintainer scripts may provide guarantees of correctness or at least the satisfiability of some properties which are important in the context of package maintenance, like commutation of two scripts, idempotency, preservation of some parts of the file system, etc.

The ongoing ANR project CoLiS has the objective of developing techniques and tools for the verification of UNIX maintainer scripts in Debian GNU/Linux packages. The internship is part of this research project.

The particular use case of maintainer scripts allows us to focus on features of POSIX-shell (which may be seen as a core of the GNU bash shell) that are actually useful in this context. Indeed, the main purpose of these scripts is the transformation of a file system through basic file system operations like adding, removing, or renaming files. Therefore, an abstract model of these scripts is considered in the CoLiS project. In this model, a

file system is a tree-like hierarchical structure, and we abstract away from some quirky features of the POSIX shell language.

The objective of this internship is to develop a method to find bugs (violations of the aforementioned properties) in such scripts based on *symbolic execution*. The method should be able to abstract inside a symbolic representation the most important features of the file system. On the proposed symbolic representation of the shell configurations, the method should define the transformers corresponding to the post-condition computation of shell statements for basic file tree manipulations like removing and renaming files and directories. The expressivity of the symbolic representation should be complemented by good decision properties for checking emptiness. For this, the intern could develop new decision procedures or translate these problems into problems for existing automatic provers.

We expect from the intern to have operational knowledge of program verification in general, and of the symbolic execution or abstract interpretation in particular. The candidate should also have at least a basic knowledge of UNIX and its shell.