Paper review

MOPS: an Infrastructure for Examining Security Properties of Software
by Hao Chen and David Wagner

This paper introduces MOPS a tool that checks for vulnerabilities in program codes. MOPS uses finite state automata to model possible vulnerabilities ("unsafe" programming practices), and pushdown automata to model control flows through the program. These are then passed to a model checker which verifies whether some program flow contains unsafe operations.

The emphasis is put on the soundness and scalability of MOPS. Soundness is ensured by the model checker which verifies these safety properties of all possible execution paths. Scalability is achieved by efficient model checking algorithms, and restricting the check to control flows (not including data flows which may render some execution path infeasible). This results in some false alarms, which the authors claim to be tolerable in practice. It would have been more convincing if they had shown the number of warnings, and the number of real vulnerabilities.

Another strong point of MOPS is that it does interprocedural check. Many tools do not perform interprocedural check since it is quite costly. However, one may agree that interprocedural vulnerabilities are much more difficult to detect manually, and hence MOPS is a useful tool.

I feel that the properties being checked in the paper seem somewhat limited and specialized to programming using system calls. Maybe because this is their initial test on MOPS, and further properties will be checked in subsequent versions. Also, the lack of new vulnerabilities discovered by the tool raises doubt to whether it can detect potentially new vulnerabilities, opposed to detecting vulnerabilities it uses as a learning base. In conclusion, MOPS uses quite rigorous modelling, and exploits powerful tools available in the model checking context, which makes it sound and scalable. However, the lack of newly discovered vulnerabilities by the tool somewhat degrades the impact of the paper.