Critique on

Joseph P. Near and Daniel Jackson. 2016. Finding Security Bugs in Web Applications using a catalog of Access Control Patterns.

Astha Sharma
asharma6@uno.edu

SUMMARY:
The authors of the paper, here, have proposed a technique that involves finding of the unseen security checks in web applications using a series of access control patterns. Each of these pattern imitates a common access control use-case, and finds application specific bugs. The implementation, aka SPACE - Security Pattern Check, checks if the application's code (the one they are testing for bugs) matches the security pattern with those present in their catalog. The patterns defined are of different types ranging from ownership, public objects, authentications, to explicit permissions, user profiles, administrators and explicit roles. This is a highly automated technique for finding out the security bugs based on the idea of matching extracted access pattern. Here, they have evaluated a total of 50 most watched open source Ruby on Rails applications on GitHub and found 33 possible bugs out of which 23 were previously unknown and the remaining 10 were false positives.

STRENGTH:

- The time to computation and pattern finding is very less. Hence it gives out results in short period of time. Therefore, fast. For example: For a bound of 8 (with 8 objects of each type), it required less than 10 seconds to finish the verification for all applications.
- The tool uses a bounded model finder to find violations of its pattern library.
- The default 8 scales of Space is good and covers most of the bugs.

WEAKNESSES:

- Covers large number of non-specific fields, thus, leading to less accurate results
- No mentions about how the bounds are set for bounded model finder
- The finite bound of the model finder may not be able to address the counterexamples exposing bugs having large number of objects.
- Space assumes the environment at the time of analysis is same as that of the production time - which is not true. Ruby allows the change in implementation of methods at any time.
- Detects the information flowing only from ActiveRecord calls to rendered pages and misses out the leakage through native code, direct network connections, or non-ActiveRecord data sources.

- It requires manual determination of the false positives.