Review 21

4/10/12

CS6V81.502

Seungtack Baek

**Automatic Input Rectification**

In this Paper, Long et al. demonstrated how to protect an application from crashing via atypical inputs, such as too large inputs, etc. They achieve this through following steps.

1. Identify benign inputs from buggy inputs by monitoring application’s execution status.
2. If it does not fail, classify the input as benign, and extract the input’s characteristics, such as length of the string within the data, and negativity of the integer input, etc.
3. If fails, try to rectify the input so it will have the characteristics seen from benign inputs.

They don’t quite “extract” the requirements for the typical inputs. Yet, they start with pool of requirements and start marking each requirement as false, whenever they see a benign input that act as a counterexample. As opposed to machine learning, this method can produce a feature set limited by the initial feature set. Still, this method seems to serve its purpose when you know you will only need to care about certain aspects of the inputs.

The most interesting aspect in this paper is that they actually allow buggy inputs to go through the application (which I think can result in catastrophic outcome – but just as any other methods, such as filtering might result in). If they falsely classify buggy inputs to be benign, this can lead to something similar to SQLIA.

Another interesting contribution is that they are trying to infer requirements for the inputs to the application without a priori knowledge. I really would like to see the quality of this method, in comparison to manually inferred requirements.

**Question:**

1. Do you think this is safe enough to be used in very information-sensitive application?