This paper was about using a tool called CQUAL to analyze the use of LSM authorization hooks in the Linux kernels. CQUAL is a tool that adds qualifiers to standard variable declarations and assignments, the consistency of which can be checked using standard type checks. In order for a section of code to be considered correct, whenever controlled objects, ie objects which are security sensitive, are modified, there must be some LSM hook that authorized the object. When an object is created, it is given the unchecked qualifier. Once it has passed an LSM hook, it is given the checked qualifier. In order to instrument the kernel code to have these qualifiers, perl scripts were used to modify the source.

This seemed to be a well written paper. All of the topics it discussed were explained without invoking to many specialized terms, and were lingered on long enough to explain fully without being repetitious. I liked the simplistic approach the authors took towards analyzing the code. Using only more basic tools, they were able to perform complex analysis.

One thing this paper didn't address was that the kernel code, as stated in the paper, can flow arbitrarily. It may tend to use C function call semantics, but if it doesn't in significant portions of code, this could seriously damage the analysis. Also, this scheme didn't seem to catch that many errors, while throwing a lot of false positives. The paper states near it's start that there hasn't been a concerted effort to verify the placement of these hooks, so it would be very suprising if the only errors are the ones this setup identified. It seems like this system may have caught only a few of the potentially numerous errors, and collected a lot of false positives because of an inherent insensitivity to flaws. In order to catch any errors at all, the sensitivity of the test had to turned up so much that the real results were almost drowned out by “noise” errors.