Today 95% of all mobile devices run Android, Symbian, iOS or RIM, all of which share the same security model for third party applications. Each time a new application is installed the OS asks the user if he or she grants the application a set of permissions. Such permissions typically allow the application to access internet, the GPS hardware, address book data, camera etc. Unfortunately this model is quite crude and most useful and innocent tasks require a combination of permissions which could just as well be used maliciously. For this reason it is important to study techniques, such as the one I will present, which allow policies to be expressed at a finer level of granularity.

The technique combines *data flow monitoring* with *control flow monitoring* by (a) keeping track of how each value in the execution has been computed and by (b) preventing certain function calls from being made based on this information. This data-centric approach to runtime monitoring allows for a wide range of policies to be expressed, including API protocol policies restricting which methods may be applied to what arguments and data flow policies stating how data must have been processed before being passed to certain functions.

The approach differs from traditional runtime monitoring on 3 key points: (1) While ordinary techniques, perform monitoring globally on the level of an application, our technique operates on the level of data. This means that different computations and processings of data are monitored in isolation, despite being arbitrarily interleaved on an application level. (2) Existing runtime monitoring frameworks are typically based on DFAs, edit automata, LTL, context free grammars, or a variations thereof, all of which rely on a model of *linear* monitoring. Since we in our work focus on how data is processed i.e. which functions are applied to what arguments, rather than in what order actions are performed, we work with a *tree based* model of monitoring. (3) Similarly to work by others we let the function calls be the actions observable by the monitor. The fact that monitoring is performed on the data level however, allows the observable actions to include the computational history of the arguments of the function calls, which makes our definition of observable actions unique.

The presentation will cover the theoretical formalization of the framework including the program model and policy semantics and a demonstration of our implementation.