Self-Adaptive Security Systems

Aimee Borda

Supervised by Dr. Vasileos Koutavas

Collaborators Prof. Matthew Hennessy (TCD), Prof. Bashar Nuseibeh (UL), Dr. Liliana Pasquale (UL)

Motivational Example

Access depends on the context. For example in the figure above:
1. Access Denied - from GPS, the user is not in the building
2. Access Allowed - location and identity are verified through GPS & fingerprint functionalities in the phone
3. Access Allowed - location and identity are verified through the IP Address and credentials.

Policy Enforcement needs to adapt to Context changes at runtime

Research Questions

- When is an SASS correct? Can we break down correctness to each sub-system?
  - For the Monitor phase, it may imply that every change is detectable.
  - For the Analysis, it may mean that all potential violations in the next k-steps are discovered
  - For Planning, it may mean that planned actions indeed guard against all violations
  - For Execution, it may mean that planning is implemented faithfully
- What verification techniques are suitable for SASS?
  - We envisage a combination of Static Analysis, Model Checking and Runtime Monitoring for different phases
  - How can we tackle complexity of verifying SASS Systems?
  - Is the approach modular?

We need the right formal model of SASS to be able to verify correctness effectively

Self-Adaptive Security Systems (SASS)

SASS is a prominent approach to address some of the limitations of traditional security systems. SASS can adjust itself to changes during runtime without human intervention [1]. A popular adaptation framework is the MAPE-K Feedback loop [2]:

- Monitor - The Adaptation System monitors the system to detect change
- Analyse - through look-ahead heuristics determine pro-actively whether a future violation is possible
- Plan - find action plan to avoid future violations
- Execute - apply the action plan to the system

SASS address policy Enforcement in Evolving Systems

Approach

A popular approach for modelling standard security systems is runtime-monitoring [3]:

- Violations are detected by looking at the history
- Countermeasures include suppressing operation or imposing alternative sequence of operations
- Similar to SASS except SASS takes into account the context

How can we extend these to model SASS? Will they be useful for modular verification?

References