Mitigating Authorisation Flaws with Hybrid Enforcement of Category-Based Access Control (CBAC)

Dr. Asad Ali, Innovation Manager, asad.ali@identitymethods.co.uk

Context & Problem

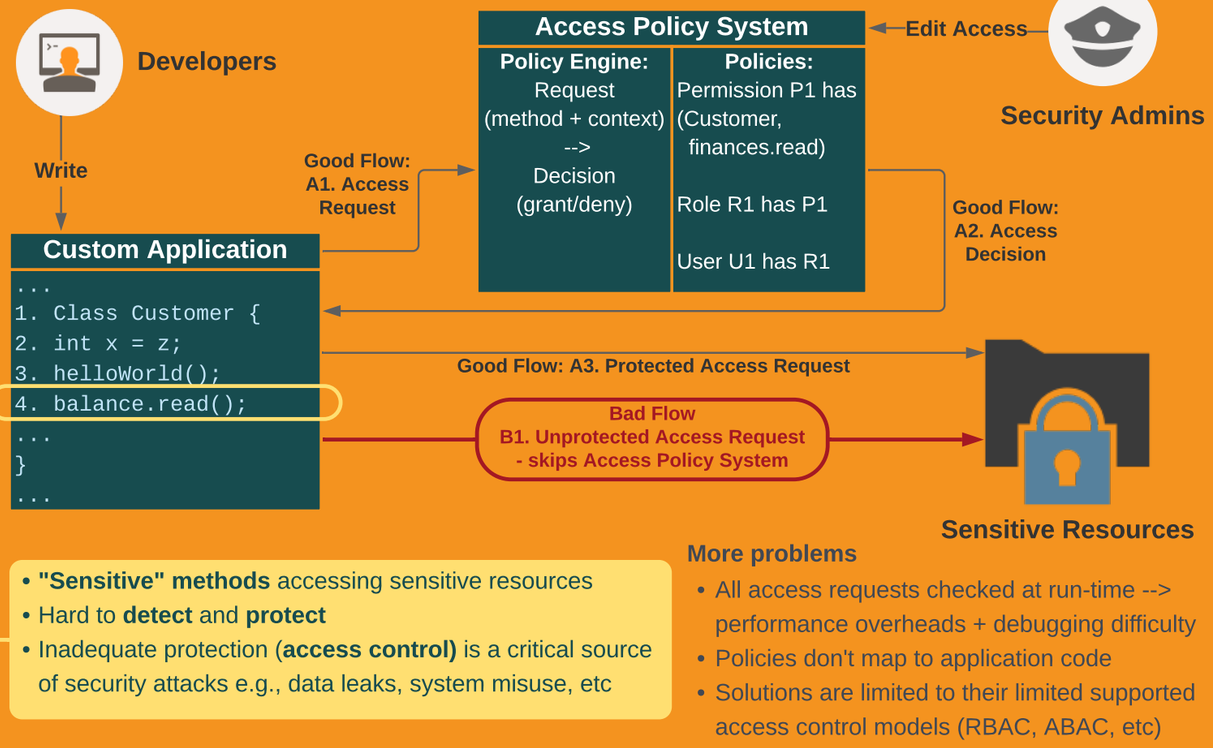
Problem Statement

Authorisation flaws in software are an IEEE Top 10 Security Design Flaw

How can we ensure each sensitive method can run only when allowed by an access policy?

-Is access queried prior to every "sensitive" call?

-Is the method only run if the decision is to grant access?

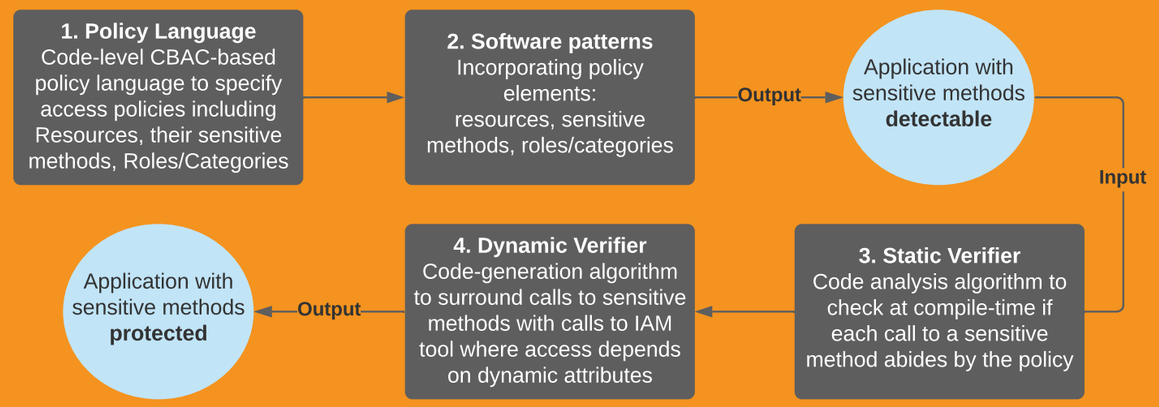


- **"Sensitive" methods** accessing sensitive resources
- Hard to **detect** and **protect**
- Inadequate protection (**access control)** is a critical source of security attacks e.g., data leaks, system misuse, etc

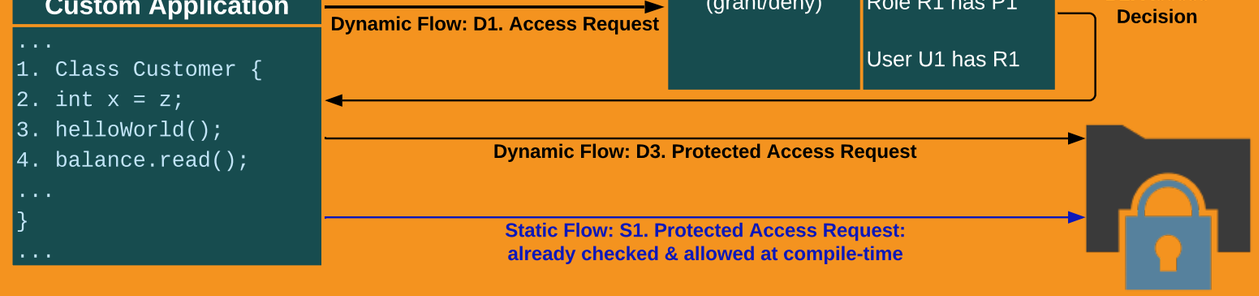
More problems

- All access requests checked at run-time --> performance overheads + debugging difficulty
- Policies don't map to application code
- Solutions are limited to their limited supported access control models (RBAC, ABAC, etc)

Solution



- **Dynamic Flow:** for requests requiring run-time information --> always routed to Access Policy System
- **Static Flow**: for requests not requiring run-time information --> only those that are always allowed will exist in the app



Results & Outcomes

Authorisation flaws mitigated by detecting and protecting all sensitive methods

Using the CBAC meta-model enables solution to be adaptable to all access models (ABAC, RBAC, etc)

Catch many errors at compile-time, aiding debugging and reducing run-time overheads

Contributes to "security-by-design" and "shifting security to the left" movements

Future Work

Build a fully-fledged CBAC-based policy system & simplify the process of writing code-level policies

Experiments with real applications in a variety of domains

Build highly-usable tools for all four parts of the solution

Enhance Design Patterns to support widely-used architectures e.g. inheritance, microservices. etc.

IDENTITY METHODS®

+448452 411836

INFO@IDENTITYMETHODS.CO.UK

LINKEDIN.COM/COMPANY/IDENTITY-METHODS