A Reusable, Secure Reference Monitor Based on the Aura Programming Language

Luke Zarko

FACULTY ADVISOR: Steve Zdancewic

Abstract

Aura (Jia et al., 2008; Vaughan et al., 2008) is a dependently typed higher-order programming language. It was designed to facilitate the automation of institutional access control policy. Following the Curry-Howard isomorphism, Aura types correspond to logical propositions and its expressions correspond to proofs. At runtime, Aura programs manipulate evidence to construct these authorization proofs. Aura's typechecker is used to determine if proofs provided to support access to a restricted resource are properly formed in accordance to policy. Furthermore, proofs used to support these accesses may be logged for later auditing.

Policy Refinement and Fine-Grained Delegation

Knitter may want to allow advertising affiliates to add themselves to any user's friends list. To do this, they merely need to refine the policy to allow friends to add friends:

```
MayAddFriend p target affiliateId
```

Defining Policy and Manipulating Evidence

The equivalent Aura type for this proof object would be:

```
pf (openlogin says (IsLoggedIn p uid)) ->
  (acl friendId) ->
  MayAddFriend p target affiliateId
```

Conclusions and Future Work

The system I implemented proved to be up to the task of running the Knitter service—and service providers to produce evidence by combining assertions. Clients may do this without going back over the network—the reference monitor will check to make sure the type system's rules are followed when RPCs are made with evidence objects as arguments.

```