Push Authorization / Transactional Authorization

Quick overview of the AM policy framework:

AM decides what is / is not permitted but it's down to the policy agent / REST client to enforce these decisions. In this regard, AM is sometimes described as the Policy Decision Point (PDP) and the agent/client is described as the Policy Enforcement Point (PEP).

AM comes to these decisions by evaluating a set of policies (identified by the agent/client). These policies are defined by AM administrators.

Resources to be protected are modelled by URI and actions - Out of the box, AM comes with a web URL resource type with HTTP verbs (GET, POST, etc) as possible actions; it's easy to map other types of resources to URIs.

Each policy defines:

- Resource URI - constrains which resource(s) the policy applies to (can include wildcards)
- Permitted / explicitly denied actions - defines what is allowed / denied if this policy matches
- Subject - constrains who the policy applies to
- Environment Conditions - constrains when the policy applies; supports logical operators so multiple conditions can be applied
- Response Attributes - Key-Value map of data which can be returned to the agent/client requesting policy evaluation; this data may be statically defined as part of the policy or dynamically looked up from the subject's profile

Certain policy conditions, such as requiring authentication to a specified authentication module, may not currently be met by a given subject but there are steps the subject can take in order to fulfil this requirement. When this happens, AM returns "advice" to the agent/client so that the user can be redirected back to AM's authentication framework to fulfil the requirement.

Session Upgrade

In versions of AM earlier than 5.5, upon completing the requirements of the policy advice, the user's session would be upgraded and thereafter the policy condition would always be met. This is not always the desired behaviour. Sometimes, we'd prefer to protect each invocation of a given action by an additional authentication/authorization step. For example, when making a bank transfer, I'd like to use 2nd factor authentication to protect each transfer.

Transactional Authorization

In AM 5.5.0, we've introduced a new "Transaction" policy condition which allows policies to require some re-authentication step every time they wish to perform the action protected by the policy. This new condition type can be used with complex policy conditions - For example, you may wish to require push authorization for bank transfers but only if the transferred amount is over $10.

- First person
- Allow single-use access to resource
- Works with stateful and stateless sessions
- Works with existing ForgeRock Authenticator iOS and Android apps (and by extension, will work OOTB with any apps written by ForgeRock customers and partners)
- Works with existing ForgeRock policy framework
- Can be used with Policy Agents (5.0.0+) and REST clients
- Is fully audited - interactions made by various clients/devices are audited and related events can be tied together

Push Authorization

Adding "Push Authorization" is as simple as configuring a policy with a transaction condition which requires the user to complete an authentication chain, module or tree which includes push authentication. No changes are required to the mobile app.

Questions

- *Do we have to use the ForgeRock / Amazon SNS service to deliver push messages?*
    - It depends. If you want to use the "official" ForgeRock Authenticator app then yes you must. In order to send a push notification to an iPhone, the sender must have a private key (tied to the specific iOS app ID) in order to authenticate to Apple APNS servers. In order to keep this private key safe, we cannot share it with customers and partners. Unfortunately, Apple does not offer a way for us to mint keys per customer / partner. By using Amazon SNS, we can generate separate credentials per partner / customer and keep the private key safe. If you want to use a app which you have submitted to the iOS and Android app stores yourself, then you cannot use the ForgeRock / Amazon SNS service - For the same reason as is outlined above. You can either, use an instance of Amazon SNS which you manage yourself (requiring no code change to AM), or you can use something else (in which can you will need to plugin an alternative implementation of the org.forgerock.openam.services.push.PushNotificationDelegate and configure AM to use this).

Feedback

- Would like skinning / forking Authenticator apps to be easier