Identity Management and Authentication

CMPU4023 - Enterprise Application Development

Identity Management

- The term *identity management* describes the process of managing individuals, entities or group authentication, authorisation, roles, privileges, etc across the enterprise information systems suite
- The majority of enterprise applications will require the identity of their users and entities to be verified and to restrict what actions particular users and entities can carry out within them
- A typical enterprise user will need to access and use dozens of different applications all of which will have this authentication and authorisation requirement

Centralised

- Intuitively, it makes sense that identity management would be largely centralised within the Enterprise for at least three compelling reasons

  - **Identity** - The same user or entity (having a single identity everywhere) can be managed and tracked across all of their application accesses and
  - **Security** - Centralisation enforces consistency as to how privileges are applied and standards are maintained across the enterprise IS suite; In addition, it is easier to take remedial action in the event of a security breach
  - **Cost Control** - Centralisation allows for the reduction or elimination of duplication which should be more cost effective

- The presumed tradeoff here the that the benefit of the increased complexity and inflexibility of a centralised approach outweighs the downsides of a looser, distributed autonomous approach

Activation

- User identities are created using a process often known as activation
- This is a one-time event where a user is created within the identity management system and assigned the appropriate privileges
- For example, onboarding a new employee or creating a materials supplier account
- In the following discussion, we will assume that users are already activated and have been provided their credentials and privileges

Authentication

- The term *authenticate* means to establish and verify the identity of some user
- Activated users authenticated through *credentials* usually comprising a publicly-known part and one or more secrets, known only to the user
- The base assumption is that the activated user who can successfully pass a credentials check is authenticated
- The authenticated user then assumes some preassigned identity within the application(s) for subsequent access and actions

Authentication Strength

- Security is second only to correctness in enterprise information systems because of the sensitivity of the data being managed and processed
- As authentication is the first line of defence of the access network and the application software itself, enterprises increasingly employ ever more sophisticated credential schemes to make them harder to compromise
- In general, the more secrets that are required, the more secure the scheme will be
- Multi-factor authentication implies two or more secrets are needed

  - Something you know and something you have
  - Includes a so-called air-gap

Credentials Distribution

- The chicken and egg problem for authentication systems is how the would-be authenticating user has possession of their credentials before they need to use them (e.g. a shared secret or two-factor device)
- This is typically solved by using some out-of-band distribution mechanism, i.e. over some other interface which is not the API itself
- Credentials distribution can be a complex problem and is the concern of the wider identify management system itself
- We will assume, for our discussion, that the user already has somehow obtained the necessary credentials prior to authentication

Authenticating Authority

- We will also assume the existence of some authenticating authority that we can programmatically consult to authenticate a user
- In the context of a SOA, we can imagine this authority is just another service that exposes a secure API allowing the users credentials to be presented and a response as to whether that user is authenticated
- The authentication authority is itself an abstraction of the underlying identity management system’s security mechanism and policy framework

Authenticating APIs

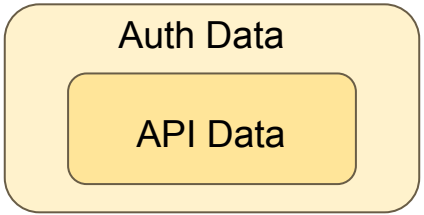
- In SOA application stacks, each service makes requests of some other service to carry out some function on its behalf
- In reality, a service request is typically carried out on behalf of some identity (ultimately a user) which, in most cases, will require the request to be authenticated and authorised before being executed
- There are two approaches for authenticating API requests, namely:

1. Authenticate each and every service request at request time
2. Preauthenticate in advance and use some temporary access token for each subsequent request verified at request time

- In both approaches, the requesting service must possess the identity credentials (shared secret) in advance by some out-of-band means

API Authentication Independence

- It is generally good practice to decouple the API authentication specification from the API specification itself
- The principal advantage is that different mechanisms (or none) can used with the same API in different contexts
- By implication, the API call syntax is therefore not directly polluted with authentication syntax but rather as a wrapper around API messages



- In context of HTTP-based API like REST, the authentication data would generally be carried in one or more specialised header fields

Summary

- Identity management is concerned with authentication and credentials management for users of an enterprise system
- It is typically centrally managed for because it is easier to enforce consistency and manage costs
- In the context of SOA systems, we expect that services will need to authenticate the requests to other services before they can be carried out
- There are a number of different approaches to doing this which we will consider in detail separately