Identity As a What?

Mark Sitkowski
Design Simulation Systems Ltd
http://www.designsim.com.au

Most of the so-called 'Identity As a Service' systems are, in reality Identity-as-a-Project systems. The magnitude of the project varies with the implementation with the worst, possibly being any of the proprietary SAML-based systems, which provide a Single Sign-On capability.
Single Sign-On is a facility whereby a user logs in with UserID and Password, or other credentials, to one application and can access other, related applications without needing to login again.
Let's call this group of applications a Single Sign-On Group.

Many authentication systems pass a token to the authenticated client, which gets passed around by the client applications, and is used to determine thedegree of access granted to the user, whenever he requests another service.

Probably the best, and most secure of such tokens is the Json Web Token, or JWT.
It has the advantage that its important content is a SHA256 hash, protected by an encryption key.

One obvious disadvantage, is that this encryption key must be known to both the authentication server, and the application client, with all the attendant risks that this carries from The Enemy Within.

The main claimed advantage, however, is that a user wishing to access a single sign-on group only needs to access the authentication server once, as opposed to connecting once per member of the group, to confirm authentication.

This freedom comes at a price, which is that all this functionality must be executed by the client.

Firstly, each application must know that it has received the token, and must send it somewhere for verification.
Most applications will do this with a few lines of Javascript, executed when the page loads.

At the back end, is the verification process which must, at the very least:

1. Know who the user is
2. Know all the members of the Single Sign-On group.
3. Know when the user logged in
4. Know whether, and to which application, the user is logged in
5. Have the ability to encode and encrypt a JWT for comparison with that received.
6. Know the JWT encryption key.
   It is common to use the user's password as the JWT encryption key, since this is known by both the authentication provider, and the application provider. This prevents the compromise of a single user's account from turning into a full-scale company-wide security breach.

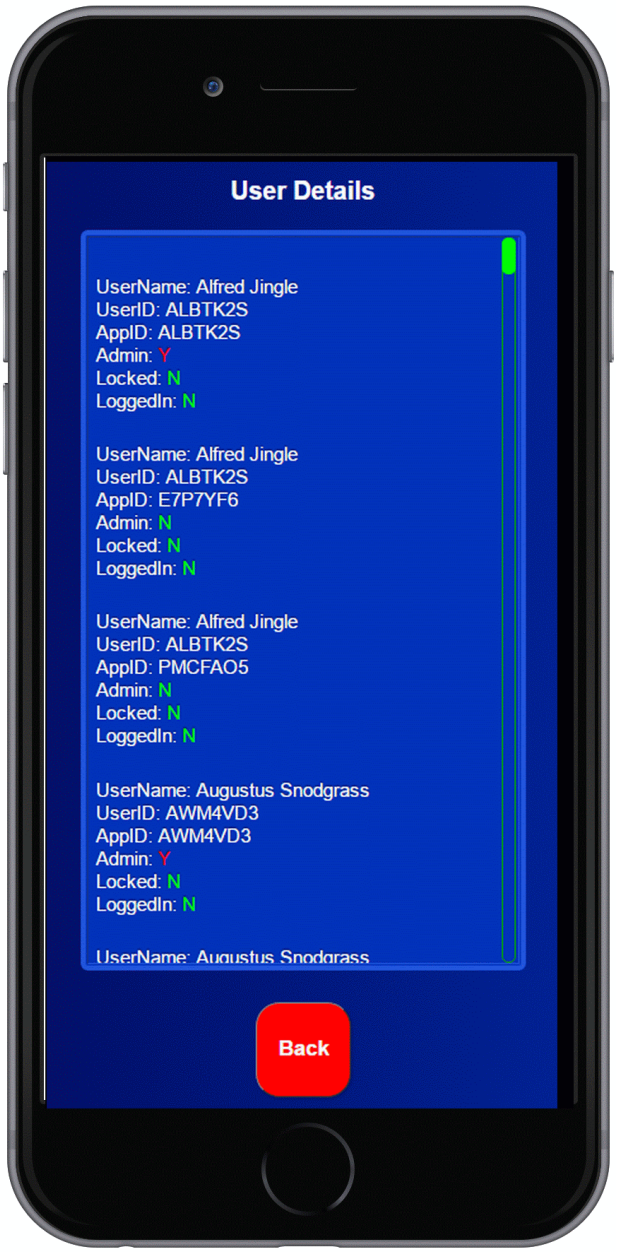
At this stage, it is becoming obvious that the 'Identity as a Service' is a non-trivial development project. The application provider will, of course, already have a database,

containing user data and application data, but this will be made more complex by the need to store and maintain login times and flags. We must add to this the complication that authentication systems using SAML must encapsulate the token operations in a message format which can require up to 100 XML statements, with all the attendant parsing requiremnents.

Even if the cost of development is ignored, the fallacy of performance improvement is highlighted by the fact that the application server is accessed as many times as that of the identity provider would have been, and that the client must perform a significant amount of processing with each access.

Identity as a Service should be just that, and should follow the paradigm:

1. Open the box
2. Install the software on your authentication server
3. Configure it from your smartphone, by the pool, with a margarita in your hand
4. Add users and applications to the database



5. Register their devices

6. Add links to your web page to use the system.

```
<form method="POST" name="bform" id="bform" target="_self"
action="http://authserver/cgi-bin/enterprise.cgi">
    <input type="hidden" id="MsgCode" name="MsgCode" value="CREQ">
    <input type="hidden" id="UserID" name="UserID" value=>
    <input type="hidden" id="ApplicationID" name="ApplicationID" value="CBHEVJC">
    <input type="hidden" id="SessionID" name="SessionID" value=>
    <input type="hidden" size="10" name="DeviceID" id="DeviceID" value=>
    <input type="submit" class ="css_button" value="Blue Diamonds">
 </form>
```

By all means issue a JWT with each authentication, but the authentication system should internally keep track of each login, permitting the client to do as much or as little token processing as the application provider sees fit.

Hardware and network bandwidth are no longer precious, but time is.

A delay of three to six months, incurred by having to wait for the completion of a totally unnecessary development project, may make the difference between getting a company's applications online on time, or not.

DSS Enterprise is a true Identity as a Service system. Although it issues a JWT, which is used internally as a SessionID, all session validity and timeout tracking is performed internally. If a user accesses an application outside the Single Sign-On group, that session is terminated.
No need to start a development project - unless you really want to.
In addition to the Session tracking, Enterprise routinely and automatically checks the device signature and IP address of the user's device, to make sure that it's not seeing a session hijack or replay attack.
The technical details are described in full in the manual, at
http://designsim.com.au/DSS_Enterprise_Manual.pdf which also describes the architecture of the JWT, and the reasons for the uniqueness of the device signature.