Wallet

Notes on march 12 lecture, specifically minute 53+ where he talks about wallet

- The full way of logging into a database is by opening a command prompt and typing in…
  SQLPLUS [USER]/[PSW]@[HOST]:[PORT]/[SERVICE NAME]
  SQLPLUS PROD3/PROD3@12.334.1334.32:1554/SPRING2024PDB
- Instead of doing all of that we could do
  SQLPLUS [USER]/[PSW]@[SERVICENAME]
  SQLPLUS PROD3/PROD3@SPRING2024PDB
  - But you must have the tsa admin pointing to the tnsnames file which has all the information for connecting to the database
  - When you use the SERVICENAME it knows to go into the tnsnames file and retrieve the tns alias's data without the host, port, or servicename
- The tnsnames files is a file which describes the database connections
- If you ever decide to change the database from one server to another all you have to do is publish a new tnsnames file
- The only bad thing about this method so far is that the password and user name is still needed which can be a security problem

We want a way to store passwords in an encrypted file, the password for the [USER] who is authenticated to the database

- In this way we don't have to give the password and username away
- We can do this with a wallet
  - We first use the "mkstore -wrl C:\Oracle21c\network\admin -create" command
    - Mkstore -wrl [location of oracle and in what directory you want to store it in inside of oracle file] -create
  - We then get asked to set a password in which case we do
  - We then get 4 new files
    - Ewallet.p12.lck
    - Ewallet
    - Cwallet.sso.lck
    - Cwallet.sso
  - These 4 files are the wallet
  - Currently there is nothing in the wallet, we havent added any credentials yet
  - We will now add a USER to the wallet
    - We use the command "mkstore -wrl C:\Oracle21c\network\admin -createCredential SPRING2024PDB PROD3 PROD3"
      - Where the last three arguments are TNSENTRY(service name) USER PSW
      - What this command will do is add that credential to the wallet
    - It will then ask you for the wallet password which was set upon creating the wallet
  - Now we have a wallet, an encrypted file, that has the credentials for a USER, lets say PROD3, sitting in that wallet
  - Now we can connect to the database using the wallet
    - Sqlplus /@spring2024pdb
    - Now we are connected as PROD3

- The only limitation with this is that you can only have one user associated with a connection
  - For example if you have 17 users, who want to wallet in you need 17 different connections in the tsnnames which sucks
- One way around that is instead of running the command…
  "mkstore -wrl C:\Oracle21c\network\admin -createCredential SPRING2024PDB PROD3 PROD3"
  We do…
  "mkstore -wrl C:\Oracle21c\network\admin -createCredential SPRING2024PDB SSO_USER SSO_USER"
  - What happens now is that SSO_USER is now the user and psw
  - If we do sqlplus /@spring2024pdb we are now connecteed as SSO_USER
- So far we only have one user in the wallet called SSO_USER which doesn't help much
  - Remember We don't have PROD3 cause we essentially restarted, so forget about PROD3
- If we run the sql code ALTER USER [USERNAME] GRANT CONNECT THROGH SSO_USER we will see that we can do
  - Lets say we granted UD_JASONC connect through sso_user
  - We can in the command line do..
  - [UD_JASONC]/@SPRING2024PDB
- What this will do is grant us connection to the database as UD_JASONC
- What this does is the admin can allow you to grant through SSO_USER but you must have the wallet on your computer (personal computer)
- How does it work:
  - In sqlnet, I have an entry/text called WALLET_LOCATION which says that the wallet location is sitting user TNS_ADMIN and %TNS_ADMIN% is pointing to that directory
  - When we try to wallet in it will open up %TNS_ADMIN% and it will look for that credential based on the input/connection you are trying to make, then retrieve the user and password and log you in that way

- HOW does it work in the big picture?
  - We do what is called single sign on user
  - We grant proxy permission to that sign on
  - We add that sign on user to the wallet
  - We install it on the application server that needs the wallet
    - Essentially we will be allowing say an employee to be able to sign in using SSO_USER and their username say UD_VICVAS (you would use square brackets like [UD_VICVAS] to denote proxy)
    - We do this proxy thing because a wallet only allows us to authenticate one user and by having that user be SSO_USER and then using it to get into our "account" with it allows us to take advantage of the features of using a wallet but also we don't have to create thousands of wallets for each user
  - We change the web configurations needed to be able to authenticate/log on
  - On the top level of the database we have a table called something like proxy_logon and some triggers
    - This will help make it so not everyone can just proxy in only authenticated computers, servers, etc.
    - We will have a sort of whitelist