Identity and Access management Section Notes

LDAP (Only on objectives acronym section)

1. Most vulnerabilities related to input validation
2. How it works is just an LDAP server that gets queries and returns directory info
   1. Can do authorization to specific objects rather than just authentication to the whole thing
   2. TCP or UDP

TACACS+

1. AAA protocol
   1. Authorization, authentication, and accounting
   2. Uses TCP
      1. Replay attacks work well cause every sequence starts with 1
      2. Session IDs too short too

RADIUS

1. Only encrypts passwords
2. Uses UDP
   1. Can’t use over unreliable connection
   2. Easier to forge packets and spoof servers
3. Allows use of shared secret
   1. Could breach the whole network with just that info
4. Sometimes susceptible to buffer overflow
5. Use with IPSec to make it secure

OAuth2 (Not on objectives)

1. AAA protocol

OpenID (Not on objectives)

1. Doesn’t do authorization

**Kerberos** (Not in objectives but book suggests I need to know a little about it’s potential problems)

The point is to authenticate both the client and the service

Session keys are what they client gets in return

Basically the AS has a shared secret code with both the TGS and the clients on it’s network. And the TGS has a shared secret with the Service. Everything is sent in two packets either to or from the client. The client can decrypt one of them either with the shared secret with AS or a key it got from that. The other one is left encrypted so the AS can send it to the TGS without the client being able to modify it and the TGS can do the same for the Service.

1. Endpoint (client) requests TGT from AS which checks the KDC database
2. KDC authenticates user in KDC and sends encrypted TGT and a TGS session key
   1. TGT has endpoint info + copy of TGS session key + timestamp
   2. TGT is symmetrically encrypted with key that servers have but endpoint does not.
   3. Message with TGS session key is encrypted with endpoint’s secret key (do they mean password or private key? - I think password)
      1. User on the endpoint has to be able to decrypt it to get the TGS session key
3. Endpoint sends “authenticator” (client ID encrypted with TGS session key) and TGT to the TGS server (both are sent encrypted)
   1. TGS decrypts the TGT with the “secret key” (probably a password or decryption key shared with the AS)
   2. TGS uses copy of the TGS session key inside the TGT to decrypt the “authenticator” which proves the client ID in it was encrypted with the TGS session key (client wouldn't have had access to that unless it had the user’s secret key)
   3. TGS checks both messages (now both decrypted) for expiration and that client ID wasn’t a duplicate
4. TGS sends two messages to client
   1. One with service session key encrypted with TGS session key
      1. Client can decrypt this one since it has the TGS session key from before
   2. Other has service ticket with copy of service session key. It’s encrypted with the service secret key (known by both TGS and service)
      1. Client can’t decrypt this one but service will be able to
5. Client sends 2 messages to network service
   1. First is service ticket (which client couldn’t decrypt)
   2. 2nd is another authenticator
      1. Has client data and is encrypted with service session key (client got service session key from TGS)
6. Service uses service secret key (which TGS knew) to decrypt the service ticket
   1. This shows service session key to service
      1. It uses this to decrypt the authenticator and checks all its timestamps
7. Service sends its own authenticator to client\
   1. Authenticator has service ID encrypted with service session key
      1. Since client can decrypt with service session key, it proves that service was able to obtain it (meaning it knows the shared secret with the TGS)
8. Client keeps using service ticket until it expires

I guess the reason for a TGT is so that KDC doesn’t have to be queried every time a service is needed? It just authenticates one user and then TGS can check that person has authorization to that service?

* TGT=Ticket Granting Ticket
* KDC=Key Distribution Center (the database of the AS)
* TGS=Ticket Granting Server
* AS=Authentication Service
* Other two things in play are endpoint (user’s computer) and network service (file server or whatever)
* Authenticator
  + Data from the thing sending it encrypted with a key that it could only have gotten by knowing a shared secret with whatever step came before it

1. Weaknesses
   1. KDC database must not be compromised
   2. Every node in a network has to support it or it won’t work
   3. Relies on timestamps to verify everything
      1. All clocks must be synchronized

