**Kerberos**

**What is Kerberos?**

* Protocol for authentication
* Uses tickets to authenticate and identify
* Uses symmetric crypto

**AD uses Kerberos by default**

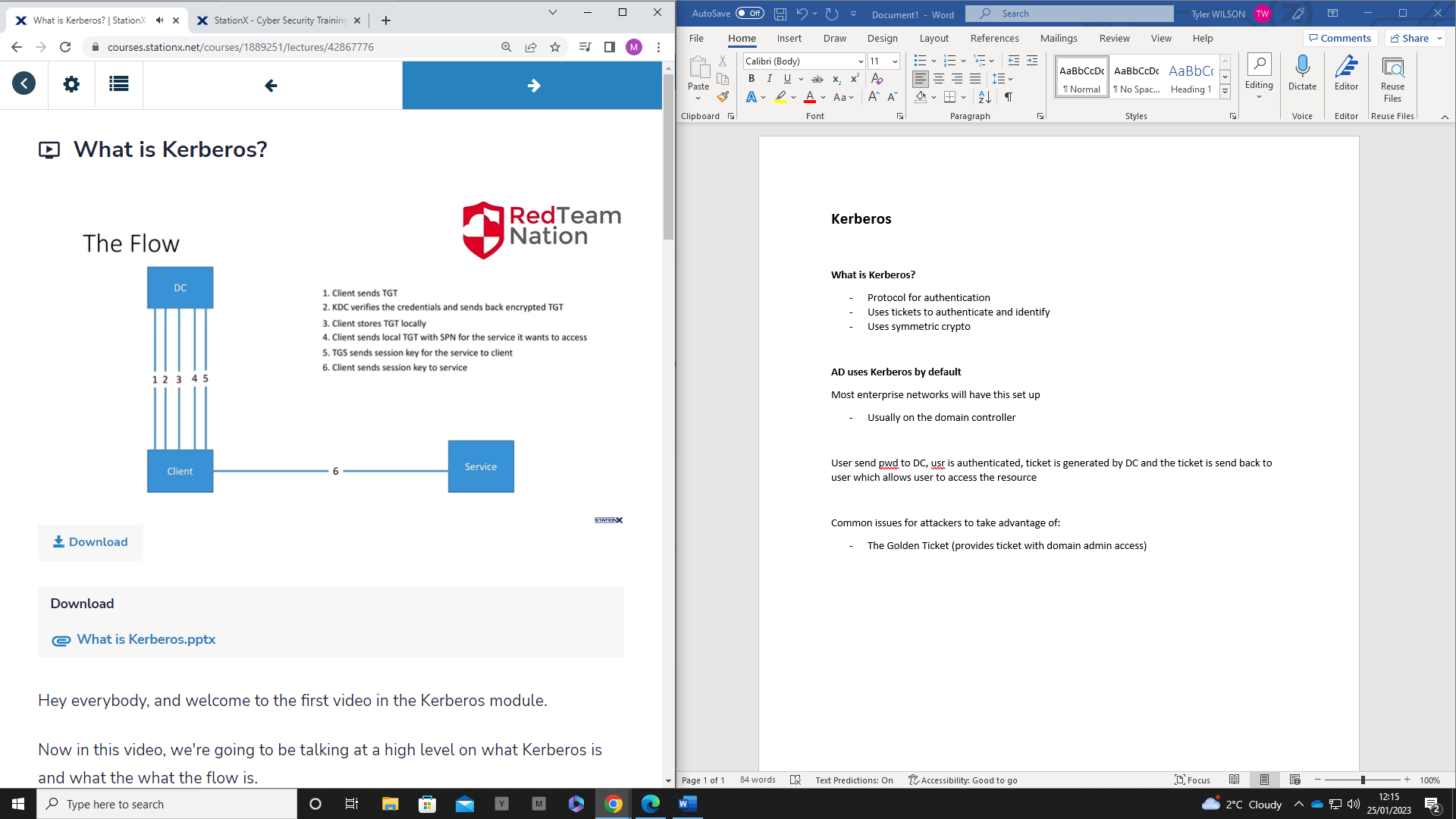
Most enterprise networks will have this set up

* Usually on the domain controller

In simple terms: User sends pwd to DC, usr is authenticated, ticket is generated by DC and the ticket is sent back to user which allows user to access the resource

Common issues for attackers to take advantage of:

* The Golden Ticket (provides ticket with domain admin access)



1. Client sends ticket generated ticket (TGT) to DC
2. KDC verifies the credentials and sends back an encrypted TGT to the client
3. The client stores the encrypted TGT locally
4. Client sends local TGT with the SPN for the service it wants to access
5. The Ticket Generating Server (TGS) will then send a session key for that service to the client
6. Client sends session key to service

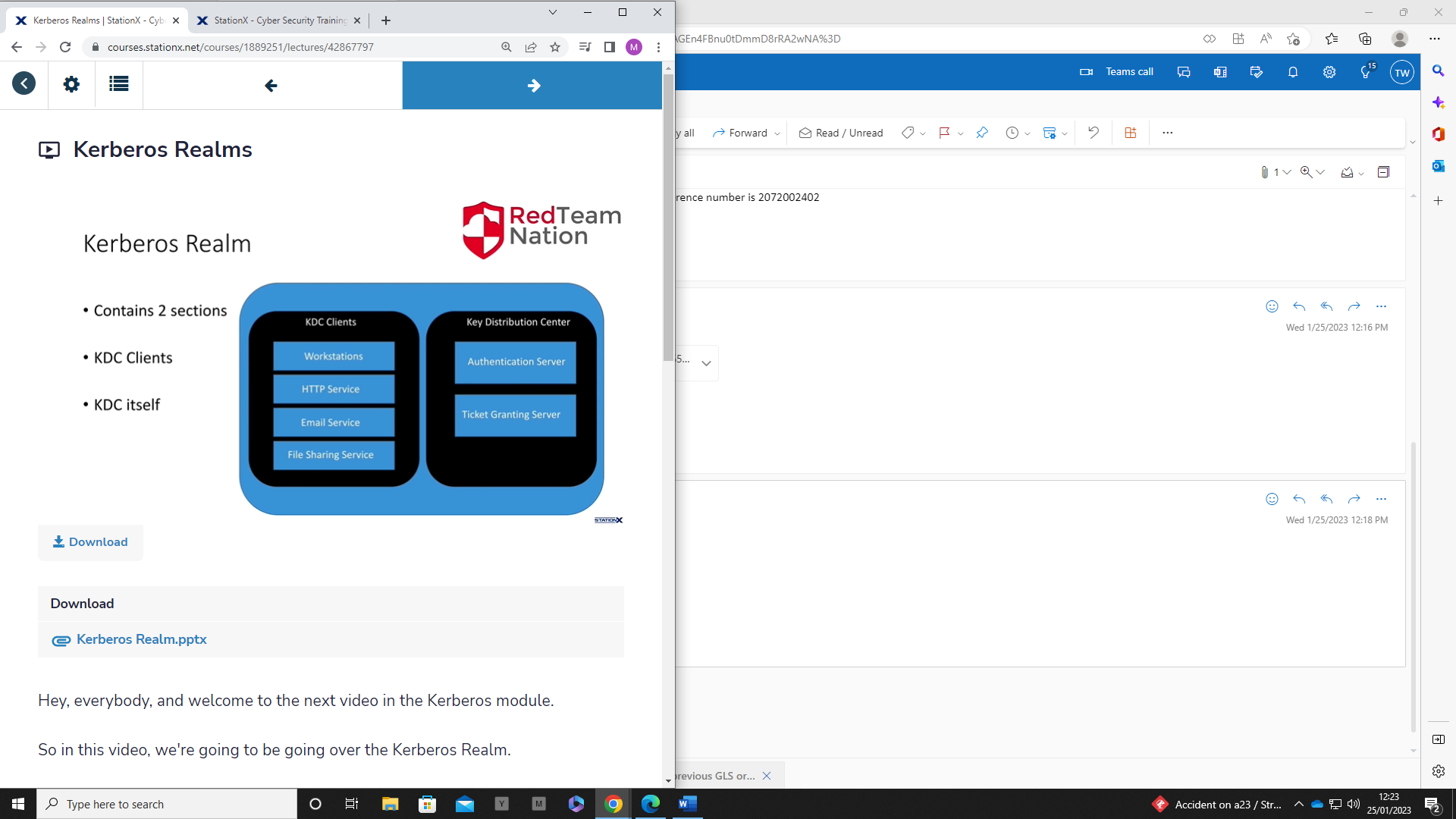
**The Kerberos Realm**

An admin created logical base network containing the resources needed for Kerberos to function

* All machines available for access, need to reside within Kerberos realm
* Policy restrictions

Kerberos realm contains 2 sections:

1. KDC clients
2. KDC itself – Key Distribution Centre



**The Authentication Server**

Handles the authentication before a ticket is generated

* Doesn’t generate the ticket
* Generates sub-ticket that is used to talk with the TGS to generate a Ticket

**Requesting a TGT**

Client will send plaintext request for TGT

* Name
* Name of Service (SPN) Service Principal Number
* IP
* Lifetime of ticket before expiry

The authentication server will then check the local db to see if user is present (exists)

* No issues = session key generated
* This key is used to communicate with other KDC servers in a more secure manner

The AS usually generates 2 messages:

1st (Encrypted with TGS secret key)

* Your name
* TGS name
* Timestamp
* Your IP
* Lifetime
* TGS session key

2nd (Encrypted with your client secret key)

* TGS name
* Timestamp
* Lifetime
* TGS session key

CAN decode 2nd message, CAN’T decode 1st

**The Client Key**

Generated by appending the usr (username@realm) with the pwd and encrypts it

* This becomes your client key which can be used to decrypt the 2nd message containing the TGS key

Now the client can communicate with the TGS with the session key provided by the AS

**Ticket Granting Server (TGS)**

Client will now send 3 messages:

* 1st (encrypted with TGS session key) AKA Authenticator
  + Name
  + Timestamp
* 2nd (Plaintext)
  + SPN Service name
  + Lifetime
* TGT from Authentication Server (encrypted with TGS secret key)
  + Contains decryption key for TGS session key

**TGS Response**

* Upon receiving the 2 messages from the client, the TGS will look up the service to see if it can find it
  + Does it exist
  + Does client have access to it
* TGS then decrypts the Clients TGT message with the TGS secret key
* Once decrypted, the TGS session key can now be extracted
* The TGS session key is now used to decrypt the 1st message or authenticator

There are a few things that occur on the TGS before its response

* Compares the UID sent to what the TGS has stored
* Timestamp check (normally 2 minute delay is max)
  + Prevents MITM and spoofing
* Check lifetime (expiry)
* Check if authenticator was not already sent before
  + Can’t send twice
* Validates client IP

**Service Key**

If all checks out, the TGS will generate a Service session key

Then generates the Service ticket for the client (Encrypted with the service secret key)

* Name
* Service name
* IP
* Timestamp
* Lifetime
* Service session key

**TGS Response**

Prepares 2 responses

* 1st sends Encrypted service ticket to client
* 2nd send encrypted ticket with the TGS session key

**Clients Turn**

Client will need to interact with service now

Client receives both messages

* CAN’T decrypt service ticket yet
* CAN decrypt the other message with TGS session key

Client will generate new Authenticator request with the service key

Client will then send the service ticket from the TGS and the new authenticator ticket to the service itself

**The Services Turn**

Service will decrypt the service ticket with its service secret key

* Extract service session key

Validates the Authenticator, Service ticket and User

* Same process as TGS

The service sends back an Authenticator of its own with its ID and timestamp encrypted with the service session key

* The client stores this ticket locally and can now authenticate to the service