Grupo 3



André Pires, nº 76046 Miguel Cruz, nº 76102 Mauro Teles, nº 70200

ATENÇÃO – É preciso que o servidor ID seja lançado antes do servidor STORE.

SD-ID.A

Características gerais da implementação do ”Kerberos”:

- Todas as mensagens que contêm informação do protocolo são XML verificado com ficheiros XSD pelo servidor.
- O algoritmo de cifra usado em todo o protocolo é o “AES” com chaves de 128 bits com “Padding PKCS5 “.
- As chaves “Server Keys” são partilhadas num ficheiro serverKeys.txt para que ambos os servidores possam aceder-lhes.

“Kerberos” 1º ronda de mensagens (Pedido de autenticação ao SD-ID)

- O cliente envia abertamente uma mensagem de pedido de autenticação com um “Nonce” e o serviço que quer usar, o servidor guarda o “Nonce” durante o período que o ”ticket” é valido. De 5 em 5 horas o Servidor limpa os “Nonces” que já podem voltar a ser usados novamente.
- Nesta fase usamos o algoritmo de resumo MD5 para resumir as palavras-chave e criar as “Client Keys” usada para o servidor enviar a “Session Key” e o “Nonce” seguramente para o cliente.
- O cliente ao receber o “ticket” e o “authenticator” verifica se o “Nonce” recebido foi o mesmo que enviou, caso seja cria uma “Credential” que contém o ticket e a “Session Key” devolvendo ao cliente num “byte[]”.

“Kerberos” 2º ronda de mensagens (Pedidos de serviços usando as credenciais)

- Nesta fase sempre que existe um pedido é passado ao SD-CLIENT a “Credential”, para que possa com os dados dela construir um “Authenticator” que contem o tempo de envio e o cliente que envia mais o ticket (ainda cifrado com “Server Key”). Para além disto no handler final, faz-se um MAC da SOAPMessage (Usando a função HmacSHA256) que usa o algoritmo de resumo SHA256 e a chave de sessão), adiciona-se o MAC ao “Soap Header” e envia-se.
- Do lado do servidor (O “Handler” passa para o servidor para além do que veio do cliente a mensagem SOAP em bytes) antes de executar os pedidos o servidor verifica pela seguinte ordem:

1. Desencripta o ”Ticket”, verifica se o cliente que pediu é igual ao que está no ticket, verifica se o ticket ainda não expirou, caso algo falhe lança “RunTimeException”.
2. Retira a “Session Key” do “Ticket”, e aplica a mesma função MAC (que o Cliente) a mensagem que chegou ao servidor, se os 2 forem iguais a mensagem é válida.
3. Por fim verifica-se com o autenticador se o tempo de pedido não é anterior ao último pedido desse cliente (“Replay Attack”), e se o cliente que está no autenticador é igual ao do pedido e do ticket, caso esteja tudo certo guarda-se o novo tempo (num “HashMap”) e a chave de sessão se ainda não estiver armazenada (num “HashMap”).

- Na resposta o servidor adiciona ainda no “Header” o tempo de pedido que o cliente enviou cifrado com a chave de sessão. O cliente ao receber vai comparar se estiver mal a mensagem conta como errada.

SD-STORE.B

Características da implementação da replicação:

- Funciona com um protocolo de quoruns simplificado, como sugerido, em que se espera por Q>N/2 respostas de servidores (inclui excepções causadas por intrusão nos canais de comunicação).
- Sendo consistência relaxada, não se faz operação de load para receber as tags de cada réplica, apenas enviando-se o pedido.
- Os pedidos são feitos assincronamente, para aumentar a eficiência, e deixa de se receber resposta ao fim de Q respostas ou de 3 segundos, cancelando-se os pedidos pendentes.
- No caso de createDoc:
  - Todas as respostas obtidas são contadas.
  - Caso haja mais excepções de DocAlreadyExists do que as Q respostas de confirmação necessárias, o procedimento é cancelado e lança-se essa excepção DocAlreadyExists, pois não se terá recebido uma maioria de respostas de confirmação.
  - Se vier uma excepção causada pelo handler, cancela-se o todo o procedimento de createDoc.
- No caso de listDocs:
  - As excepções do handler não são contadas.
  - Todas as respostas válidas, que incluam uma lista de documentos de um servidor, são aceites e guardadas.
  - Se houver mais excepções de UserDoesNotExist do que respostas válidas (listas), o procedimento de listDocs é terminado com excepção de UserDoesNotExist.
  - Caso não haja listas nenhumas, é lançada uma excepção de KerberosInvalidRequest.
  - No caso de sucesso, todas as listas são *merged* numa nova list de modo a mostrar todos os ficheiros existentes nas réplicas de Store que havia durante o envio do pedido de listDocs; essa lista com todos os documentos não tem duplicados e é ordenada posteriormente pelo FrontEnd, antes de retornada ao cliente.