A48 – Terceiro Entrega do Projeto de Sistemas Distribuídos

 83420 – Alexandra Figueiredo

 83443 – Denis Voicu

 83520 – Mariana Loureiro

GitHub: https://github.com/tecnico-distsys/A48-SD18Proj

Kerberos

A abordagem usada tem base no protocolo *Kerberos*. Em seguimento da entrega anterior, é assumido que existem três réplicas *Station*, assim como uma instância *Binas* a correr, e também os dois clientes, *binas-ws-cli* e *station-ws-cli*. Faltas de seguranças devidamente identificadas pelo processo detalhado em baixo são reportadas com instâncias de uma *RuntimeException*, método pelo qual é garantida a integridade e frescura da mensagem.

O cliente começa por se autenticar com o servidor *Kerby* ao requisitar um *ticket*, com a sua identificação (*email*), a identificação do servidor a contactar (*email* do *Binas*), um valor *nounce* (um valor único e aleatório que servirá depois para efeitos de validação, e a duração da sessão. Recebe duas estruturas de dados, uma *SessionKey* e um *Ticket*, em resposta. Tanto o cliente como o *Kerby* têm conhecimento da *password* do utilizador a que corresponde o *email* enviado, pelo que a chave do cliente (*Kc*) é conhecida por ambos, e pode ser usada para cifrar e decifrar a mensagem enviada.

O cliente valida o *nounce* devolvido na *SessionKey*, e, agora conhecendo a chave (também no *SessionKey*) que irá partilhar com o *Binas*, envia-lhe o *Ticket*, que não consegue abrir. O *Binas* decifra o *Ticket* com a sua chave *Ks*, que o *Kerby* conhece por ter acesso à *password* usada pelo *Binas* também. Passa a conhecer a chave partilhada *Kcs*, e verifica que o endereço do servidor no *Ticket* corresponde à sua identidade. Para ter a certeza da frescura da mensagem, os dois *timestamps* no *Ticket* devem ser inclusivos do momento atual.

Em adição ao *Ticket* (e ao pedido – *request* – propriamente dito), o cliente envia também uma estrutura de dados *Auth* ao *Binas*, que contém uma instância temporal e o seu *email*. O *Binas* valida que os *emails* (no *Ticket* e no *Auth*) são idênticos, e que o *timestamp* não pertence a uma listagem de *timestamps* previamente recebidos (o *Binas* guarda, para cada utilizador, o *timestamp* da última mensagem recebida com o respetivo *email*). Sendo tudo corretamente validado, envia então de volta para o cliente – novamente cifrado com a *Kcs*, e acompanhado da resposta ao pedido – o *timestamp* recebido no *Auth*, para que, do seu lado, o cliente valide a resposta por comparação com aquele que recorda enviar.

Handlers

Todo o processo acima é realizado pelos *handlers* do lado do cliente e do *Binas*. No processo de envio – dos dois lados – são primeiro feitas as ações referentes ao protocolo *Kerberos*, e de seguida é aplicado um *MAC* à mensagem, cifrado com a chave partilhada *Kcs*. Neste segundo *handler*, a *string* obtida para garantir a incorruptibilidade da mensagem é simplesmente o *request*, a partir do qual tanto o cliente como o *Binas* devem gerar *MAC*s idênticos.

Portanto, do lado do cliente, o primeiro *handler*, *KerberosClientHandler*, trata de validar a mensagem de acordo com o protocolo *Kerberos*, descrito explicitamente em cima. O segundo, *MACHandler*, faz a comparação dos *MAC*s gerados a partir da *string* do *request*. Do lado do *Binas*, o primeiro, *KerberosServerHandler*, faz o mesmo papel do seu homólogo, enquanto que o segundo, *BinasAuthorizationHandler*, tens as mesmas funções que o *MACHandler*, e adicionalmente valida que o *email* do utilizador no *request* corresponde ao *email* do utilizador no *Ticket*.

SOAP

Ilustração

Existe um cliente(C) e um servidor(S). O cliente vai efetuar um *activateUser(alice@.binas.org)*. Esse pedido é apanhado por um *handler* do cliente que coloca dentro de um envelope um *Ticket*, correspondente ao *TicketHeader* na figura, um *Auth(AuthHeader)* e o *MAC(MACHEADER)*. O pedido feito pelo cliente ao servidor é colocado dentro do envelope e posteriormente o envelope é enviado para o servidor. O servidor recebe o envelope efetua várias verificações para garantir a integridade da mensagem e a sua frescura. Gera um novo envelope com um *RequestTime*, *MAC* e a resposta ao cliente.

O cliente decide depois fazer um *getCredit(alice@.binas.org).* São efetuados os mesmos passos descritos anteriormente, mas o envelope é intercetado por um agente malicioso (*hacker*) e este reenvia o envelope, substituindo o *email* do *getCredit.* O servidor, ao receber esse pedido, e durante as suas verificações, compara o *MAC* que se encontra no *Header* do envelope com o *MAC* gerado a partir da mensagem que recebeu. Concluindo que não são iguais, é lançada uma *RunTimeException*, sendo que o agente malicioso não recebe qualquer resposta**.**