Sistemas Distribuidos

TÉCNICO LISBOA

Relatório da 3ª Entrega sobre Segurança

Grupo T06

Francisco Aguiar
84718

Gonçalo Marques
84719

Manuel Sousa
84740

https://github.com/tecnico-distsys/T06-SD18Proj

Segurança

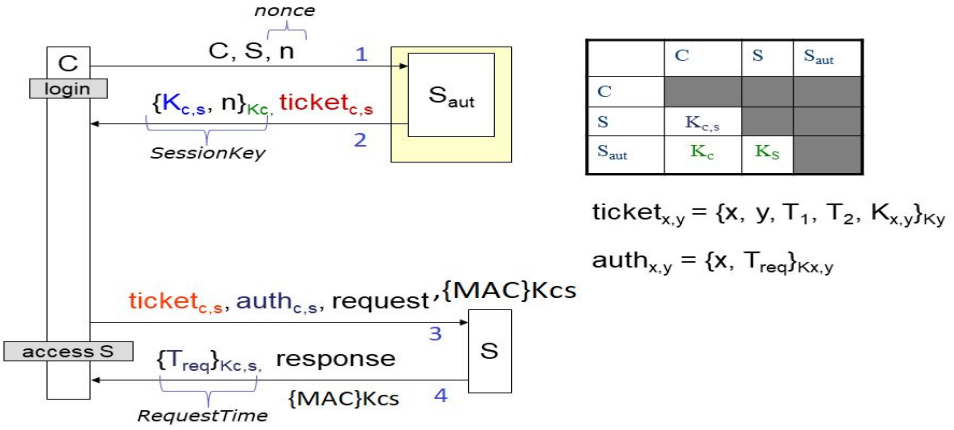
Nesta entrega temos de garantir a integridade dos pedidos e das respostas. Para isso em cada pedido adicionamos uma variação de Message Authentication Code (MAC). Neste caso específico decidimos usar um Hash-based Message Authentication Code (HMAC) onde utilizamos a função de resumo estudada nas aulas, SHA-256 juntamente com uma chave. Esta chave será a chave partilhada (Kcs) do ticket kerberos. Escolhemos esta função de resumo por se provar como segura, e por usar uma representação de bits consideravelmente grande. Até ao momento esta função de resumo não conhece qualquer colisão e dada a sua robustez, não permite que ataques Brute-Force consigam decifrar o conteúdo original de uma hash em tempo útil.

Em cada pedido cliente-servidor existe um MACHandler que é responsável por calcular o HMAC do body da mensagem SOAP usando a chave partilhada (Kcs) do ticket correspondente. Este HMAC é então colocado no header da mensagem e segue viagem para o seu destino. O destinatário desta mensagem, vai validar se o seu conteúdo foi ou não adulterado. Para isso no MACHandler do lado do servidor é retirado o HMAC e o conteúdo do corpo da mensagem. É então criado um novo HMAC, desta vez usando a chave partilhada que foi transportada no ticket (apenas o receptor sabe abrir o conteúdo deste ticket) e o body da mensagem recebida. Se o HMAC calculado for diferente do HMAC transportado na mensagem, então podemos concluir que a mensagem foi alterada por um atacante.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <e:clientTicketHeader xmlns:e="http://ticket.com">lxtsP87+0GD0MZdgPUXMM9fbbU3ZC8lGrEiYAUusGh+xCBxeWQdo6JXNofHrQHg7eAmNzKh+0
    <e:clientAuthHeader xmlns:e="http://auth.com">fNJif+4c4qcs4eRweHObVKj0PJ6nTVtMxvBfgsAHh54vxwSJ2V7xMBce2ws1UxBu3h4KDAcx0VKGb
    <m:MACHeader xmlns:m="http://macheader.com">qlevTfgGfvFJK8V2dKm9ISBZMrZF1vIEfSfHYSgtH+U=</m:MACHeader>
  </SOAP-ENV:Header>
  <S:Body>
    <ns2:activateUser xmlns:ns2="http://ws.binas.org/">
      <email>alice@T06.binas.org</email>
    </ns2:activateUser>
  </S:Body>
</S:Envelope>
```

Figura 1 - Mensagem SOAP recebida pelo binas-ws

Na figura 1 observamos um exemplo de uma mensagem SOAP enviada de um cliente para o servidor. É transportado o ticket (clientTicketHeader) que será usado para retribuir a chave Kcs. É também transportado o Auth usado pelo kerberos (clientAuthHeader) e o HMAC (MACHeader) calculado pelo emissor, que será usado pelo receptor para verificar a integridade da informação recebida. Para representação de dados binários em texto, ciframos o texto em BASE64.



Vamos agora explicar o modelo de comunicação segura aplicado ao projecto, todas as referências **(Img x)** correspondem a legenda de uma das mensagens acima esquematizadas.

Cliente-Servidor

Nesta entrega foram implementados mecanismos de segurança usando o servidor Kerberos e os SOAP handlers.

Quando o cliente (binas-ws-cli) faz um pedido ao servidor (binas-ws), o KerberosClientHandler é o primeiro a interceptar a mensagem. Se o cliente estiver a fazer um pedido pela primeira vez, ou se o ticket guardado está inválido, pede um ticket e uma chave cliente, ambos encriptados, a chave cliente servidor em conjunto com o nouce, encriptados com a chave do cliente, e o ticket encriptado com a chave do servidor, ao servidor Kerberos **(Img 1/2)**. Se esse não for o caso, usa os dados que estão guardados localmente.
De seguida, usando o SOAPMessageContext, é metido o ticket recebido e a autenticação criada criada pelo cliente e encriptada com a chave cliente servidor no cabeçalho da mensagem SOAP. O nome do pedido e o email do utilizador estão também contidos no corpo da mesma mensagem,caso o pedido em questão requeira um email para ser efectuado.**(Img 3)**Por fim pedido segue então para o servidor.

Quando o servidor recebe um pedido do cliente, o primeiro handler a intercetar o pedido é o BinasAuthorizationHandler, que extrai o ticket e a autenticação encriptados no cabeçalho da mensagem SOAP, introduzidos pelo cliente, e verifica se o ticket é válido (se a data é mais tarde do que o limite mínimo e mais cedo do que o limite máximo, com uma ligeira margem de erro) e se o ticket, a autenticação e o email no corpo da mensagem são coincidentes, verificando assim que o ticket e autenticação pertencem ao mesmo utilizador e que o corpo da mensagem não foi adulterado. Se alguma destas condições não se verificarem, o BinasAuthorizationHandler manda uma RuntimeException, e o pedido do cliente não é concretizado pelo servidor binas-ws.

Se as condições anteriores se verificarem, o pedido continua e chega ao KerberosServerHandler, que guarda a chave cliente-servidor e o TimeRequest contido na autenticação.

Servidor-Cliente

(Img 4) Quando o servidor concluir o processamento do pedido do cliente, no KerberosServerHandler, é criada uma nova autenticação com o TimeRequest recebido pelo cliente, essa autenticação é metida no cabeçalho da mensagem SOAP, e é iniciada uma resposta ao cliente.
Quando o cliente recebe a resposta do servidor, no KerberosClientHandler, é verificado se o TimeRequest contido na autenticação que vem no cabeçalho da mensagem SOAP é o mesmo que o TimeRequest mandado na autenticação para o servidor. Se não for, é mandada uma RuntimeException, e a resposta não é processada pelo cliente.