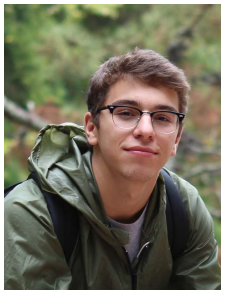
3ª Parte do Projeto de Sistemas Distribuídos

A47

URL do repositório do GitHub: https://github.com/tecnico-distsys/A47-SD18Proj

83521

Mariana Mendes



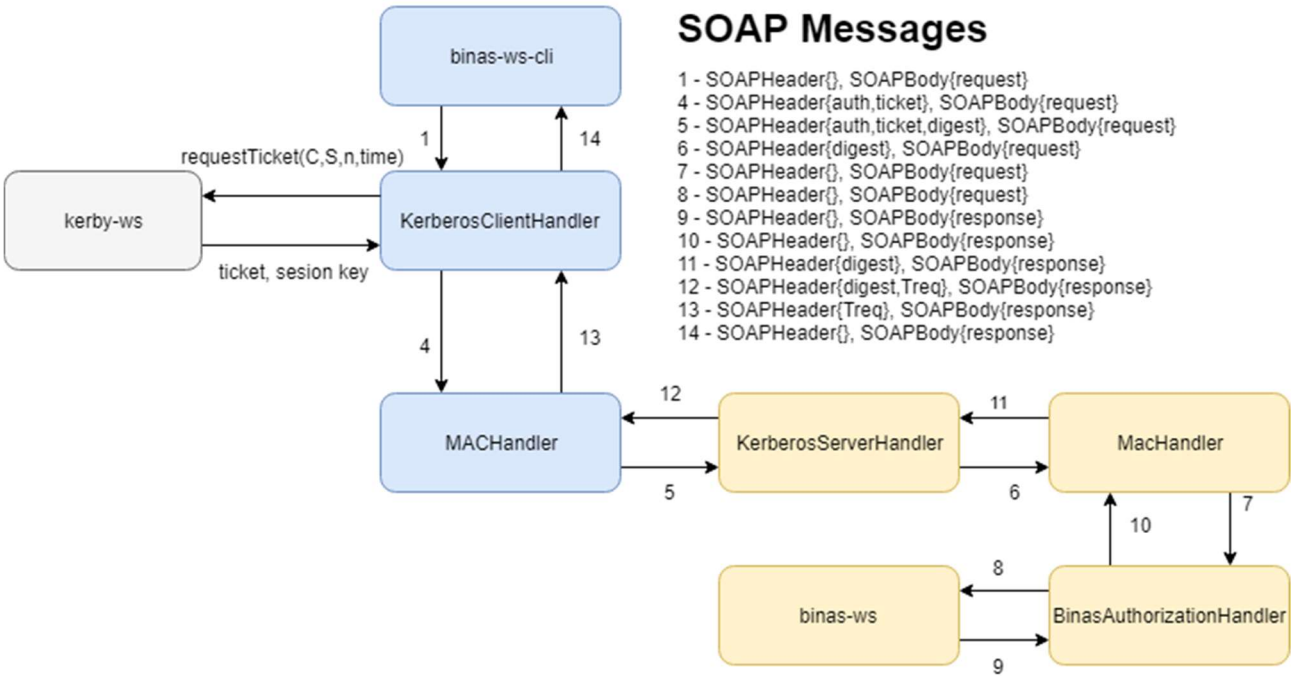
83539

Pedro Caldeira

83540

Pedro Lopes

Figura da solução de Segurança



Descrição da Figura

A figura mostra o fluxo das interações desde o binas-ws-cli até ao servidor binas-ws, de forma a garantir autentificação do cliente, integridade dos pedidos/respostas e o controlo de acessos às informações no binas-ws. Para tal, como mostra a figura, foram implementados handlers que estão encarregues da implementação do protocolo utilizando headers no envelope SOAP, que contém a informação necessária para o funcionamento do protocolo. As caixas em azul são client-side enquanto as caixas amarelas são server-side. A interação 5 representa o pedido do binas-ws-cli enviado pela rede enquanto a interação 12 representa a resposta do binas-ws pela rede.

Protocolo detalhado

Autentificação

(1) Para garantir a autentificação, são usado os moldes do Kerberos simplificado. Assim, o cliente ao fazer um pedido é feito um request de um ticket ao kerby-ws onde o cliente é identificado pelo seu email e o servidor é identificado com o e email do binas, adicionando ao request um nounce, um número gerado aleatoriamente no momento do request.**(2)** O kerby-ws responde com o **ticket={cliente,servidor,t1,t2,kcs}ks** e a **session key = {Kcs , n}kc** . Assim, o cliente tendo o Kc(chave gerada a partir da password associada ao email) decifra o session key para obter o nounce e verificar se é igual ao nounce enviado no requestTicket. **(3)** Caso o nounce seja igual, o cliente gera o **auth=(cliente,timestamp do pedido)** cifrando com o Kcs que vem no session key. **(4)** No lado do servidor, o pedido vem com o auth e o ticket. Decifra o ticket com o Ks(gerado a partir da password do servidor) para obter o Kcs. Decifra o auth e faz as seguintes verificações: o cliente que vem no auth é igual ao cliente que vem no ticket, o Treq que vem no auth é verificado se está entre t1 e t2, verifica se a hora do servidor está entre t1 e t2, se Treq não é anterior à hora atual do servidor menos 2 segundos (estes 2 segundos deve ser o menor tempo possível para evitar replay-attack). **(5)** Na resposta, o servidor envia juntamente o Treq que foi recebido pelo auth cifrado com Kcs para garantir a frescura. **(6)** O cliente depois ao receber a resposta, decifra o Treq e compara com o Treq que este enviou no auth do pedido que fez.

Integridade

(7) Para garantir a integridade, tanto nos pedidos como nas respostas, ao enviar uma mensagem SOAP é calculado um resumo que é obtido através da aplicação de SHA-256 no SOAPBody do envelope da mensagem

e de seguida é cifrado com Kcs. Este é enviado juntamente com a mensagem. **(8)** Ao receber as mensagens, o resumo cifrado calculado no outro lado é extraído e é calculado novamente o resumo do SOAPBody da mensagem recebida através da aplicação de SHA-256. Ambos os resultados são comparados de forma a averiguar se houve interceção e alteração da mensagem.

Controlo de Acesso

(9) Para garantir o controlo de acesso, o servidor verifica se o argumento que identifica o cliente(email) dos pedidos é igual ao identificador do cliente (email) recebido pelo ticket. Caso contrário o pedido é negado.

Breve Explicação da Solução

O Cliente efetua um pedido ao servidor. Este pedido primeiramente passa pelo KerberosClientHandler que solicita o ticket ao kerby-ws e gera a partir deste o auth (tal como descrito nos pontos 1,2 e 3 do protocolo). São acrescentados dois headers ao envelope: auth e o ticket, com a tag "security:auth" e "security:ticket" respetivamente. De seguida a cadeia passa pelo MacHandler que calcula o digest do body e o encripta com o Kcs (tal como descrito em 7). Este valor é adicionado ao header do envelope com a tag "security:digest". Ao chegar ao servidor, o pedido passa primeiro pelo KerberosServerHandler, obtendo os seguintes headers: "security:auth" e "security:ticket". Efetuando então o que está descrito em 4 e removendo os headers do envelope. Posteriormente, avança para o MACHandler onde obtém o header com a tag "security:digest" e executa o que é dito em 8, sendo que no fim este header é também removido do envelope. O próximo estágio é o BinasAuthorizationHandler, onde é pesquisada a tag "email" no SOAPBody e é verificado se o valor dentro da mesma é igual ao identificador do cliente recebido pelo ticket, tal como dito em 9. Por fim chega ao binas-ws onde o pedido é executado.

O servidor ao enviar a resposta, passa primeiro pelo BinasAuthorizationHandler onde nada é efetuado. Seguidamente passamos para o MacHandler onde é executado 7 e onde é adicionado o digest encriptado no header com a tag "security:digest". Por fim a cadeia passa pelo KerberosServerHandler, onde é colocado o Treq que veio na session key do pedido tal como dito em 5, adicionando o valor num header com a tag "security:requestTime". Agora na parte do cliente, ao ser recebida a resposta, passa primeiramente pelo MACHandler onde se obtém o header "security:digest" e é executado 8 sendo este header removido do envelope. De seguida avança para o KerberosClientHandler onde se obtém o header "security:requestTime" e efetua como em 6. Por fim chega ao binas-ws-cli.

Para passar informação entre handlers foi usado as funções get() e put() do messageContext. No KerberosClientHandler é guardado o Kcs (para cifrar depois o digest) e no KerberosServerHandler é guardado o Kcs(para decifrar depois o digest) e o email do cliente (para o BinasAuthorizationHandler).

Conteúdos das mensagens SOAP

```
<SOAP:Envelope>
    <SOAP:Header>
        <security:auth></security:auth>
        <security:ticket></security:ticket>
        <security:digest></security:digest>
    </SOAP:Header>
    <SOAP:Body>
        REQUEST
    </SOAP:Body>
</SOAP:Envelope>
```

```
<SOAP:Envelope>
    <SOAP:Header>
        <security:digest></security:digest>
        <security:requestTime></security:requestTime>
    </SOAP:Header>
    <SOAP:Body>
        RESPONSE
    </SOAP:Body>
</SOAP:Envelope>
```

Na figura da esquerda está representado a estrutura da mensagem SOAP do pedido enviada do cliente na rede para o servidor. Na figura da direita está representado a estrutura da mensagem SOAP da resposta do servidor enviada pela rede. A explicação para introdução dos headers adicionais estão explicados nas secções anteriores e na figura da solução.