TÉCNICO LISBOA

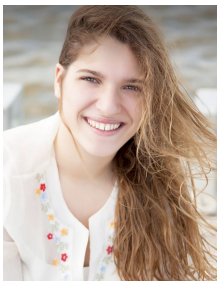
Sistemas Distribuídos

2.º Semestre 2017/2018

SD-Binas

https://github.com/tecnico-distsys/A58-SD18Proj

Relatório - Segurança

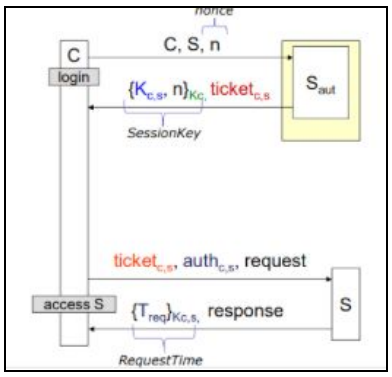


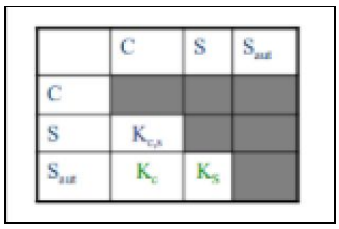
79758 Luísa Santo

83517 Sofia Martins

87823 Kevin Vos

Protocolo Kerberos Simplificado





Ticket c,s = { C, S, T1, T2, Kc,s }Ks

Auth c,s = { C, Treq }Kc,s

Nomenclatura:

C: Cliente Binas
Kc: Chave do Cliente C
S: Servidor Binas.
Ks: Chave do Servidor S
Saut: Servidor de autenticação Kerberos
Kcs: Chave de sessão do cliente C e do servidor S
T1 e T2: Intervalo de validade do Ticket.
n: Nounce, número aleatório gerado no cliente antes do envio de cada pedido.

Fig. 1 -Protocolo Kerberos Simplificado

Assume-se que as contas do **C** e do **S** encontram-se pré-registradas no **Saut.** A chave Kc é conhecida apenas pelo **C** e pelo Saut, a chave Ks é conhecida apenas pelo **S** e pelo **Saut**. O protocolo Kerberos permite os utilizadores autenticarem-se e fazer pedidos ao servidor Binas de forma segura, confidencial e manter a integridade das mensagens enviadas através de funções digest.

Troca de mensagens

O procedimento de autenticação consiste no **C** enviar para **Saut** o nome do cliente, o nome do servidor que pretende comunicar e um nounce.
O **Saut** gera um Ticket, cifrado com a chave Ks, e cria uma chave de sessão Kcs cifrada com a chave Kc. O Ticket e a chave de sessão são enviados do **Saut** para o **C** como resposta.
Após resposta do **Saut**, o **C** necessita de abrir a chave de sessão Kcs com a sua chave Kc, permitindo construir um pedido a **S**. Antes de enviar o pedido, é construida uma mensagem SOAP, cujo corpo possui informações sobre o pedido, e o cabeçalho possui o Ticket e um Auth, cifrados com a chave de sessão Kcs. Por fim, **S** abre e valida o ticket recebido com a sua chave Ks, e verificar se o nome do cliente no Ticket coincide com o nome do cliente no Auth, rejeitando pedidos caso contrário.

Controlo de acessos

O controlo de acessos consiste em rejeitar pedidos de utilizadores que pretendam invocar operações binas em nome de outro utilizador, isto é, rejeitar um pedido de um utilizador que pretende alterar o saldo de outro utilizador. O controlo de acessos é feito quando o BinasAuthorizationHandler interceta a mensagem SOAP, comparando o nome do utilizador do ticket com o do autenticador, e se existir, com o nome do utilizador presente no corpo da mensagem. Caso 2 dos nomes sejam diferentes, o pedido é rejeitado.

Replay Attacks

Os replay attacks são ataques que consistem em obter uma mensagem enquanto passa pela rede, e a reenviar com o objetivo de executar o pedido mais do que uma vez. Para combater isto, é efectuado um registo da data de envio das mensagem SOAP. Se a diferença entre a data de envio e a data atual for acima de 10 segundos, a mensagem não é processada. Também é necessário combater o caso em que o atacante volte a re-enviar a mensagem dentro dos 10 segundo. Tratamos deste caso através da geração um nounce, um número aleatório suficientemente grande, registado numa coleção do Servidor. Caso haja um pedido com um nounce já pertencente a esta coleção, o pedido é rejeitado. É possível um pedido ser rejeitado devido a nounce repetido, mesmo se foi gerado pelo cliente antes de enviar, mas a probabilidade de 2 nounce serem iguais é extremamente pequena.

Handlers

Ao invocar um método remoto, os handlers intercetam o pedido antes de este sair para a rede. A ordem dos handlers que intercetam é KerberosClientHandler e MacHandler na saída. Na entrada a ordem é MacHandler, KerberosServerHandler e BinasAuthorizationHandler.

KerberosClientHandler

No BinasClient, existe uma TicketCollection que entradas constituidas por SessionKeyAndTicketView e um tempo limite de validade, permitindo verificar a validade do pedido quando invocado o método getTicket() da TicketCollection. Caso a TicketCollection esteja vazia, o KerberosClientHandler ao intercetar instancia um KerbyClient para obter um Ticket e uma nova chave de sessão do **Saut** e guarda na TicketCollection. O Ticket e a chave de sessão são adicionados ao cabeçalho da mensagem SOAP antes do envio como duas CipheredView.

MACHandler

Este handler permite garantir a integridade das mensagens. Gera-se um MAC (Message Authentication Code) com o algoritmo SHA-256. Este algoritmo permite “resumir” uma mensagem na forma de um código, de forma a que o número de colisões seja mínima, e não seja invertível. Na saída de um pedido do **C** ao **S**, gera-se o resumo M do corpo da mensagem e adiciona-se ao cabeçalho da mensagem SOAP. Na entrada de um pedido, é gerado um novo resumo M’ do corpo da mensagem. Caso M e M’ sejam diferentes, sabe-se que a mensagem foi alterada entre **C** e **S** e o pedido é rejeitado. Caso contrário garantiu-se a integridade da mensagem.

FreshnessHandler e TimeHandler

Estes handlers permite a frescura na comunicação. Esta é garantida através do TimeHandler presente na handler-chain **S** que regista a data de envio no Header das mensagens SOAP. Se a diferença entre esta data e a data atual for acima dos 10 segundos, a mensagem é descartada. No entanto, durante o intervalo de 10 segundos pode existir ataques de repeticao. Esta situação é resolvida com a introdução de um número aleatório **nounce** através do FreshnessHandler guardado no lado do Servidor. A chegada de uma nova mensagem, confirma-se se o nouce ja foi anteriormente lido. Os números aleatório são guardados num ficheiro tsv, de modo a que os dados possam ser preservadas.

Exemplo de uma mensagem SOAP para a operação activateUser, do cliente binas para o servidor Binas.

Observa-se que a mensagem possui no cabeçalho um ticket e um auth cifrados e possui um digest gerado a partir do corpo da mensagem