



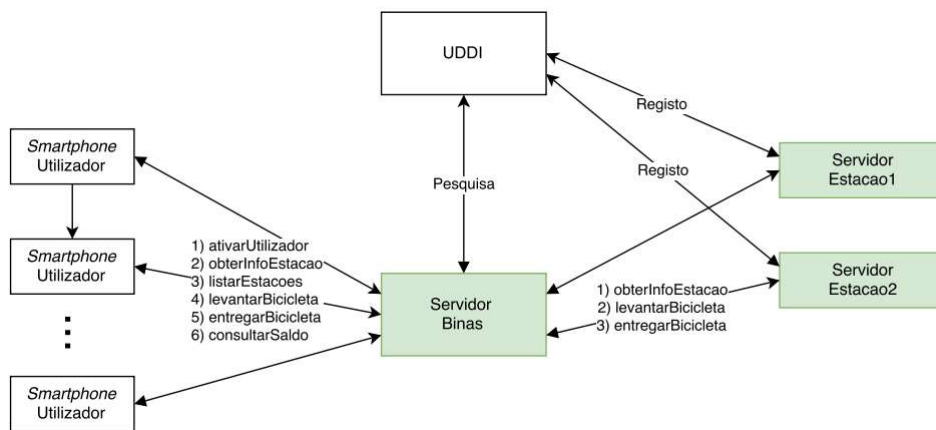
Sistemas Distribuídos

Projeto Binas, Parte 3

Segurança com Kerberos



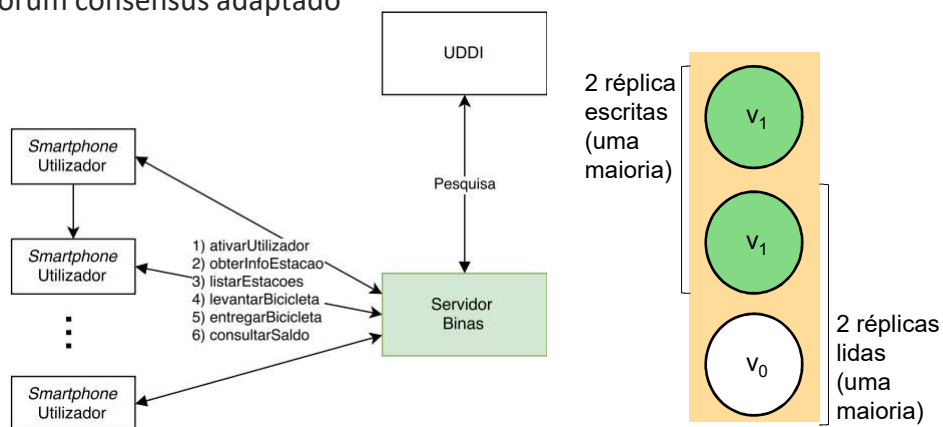
Parte 1





Parte 2

- Replicação ativa dos saldos das contas dos utilizadores
 - Quorum consensus adaptado

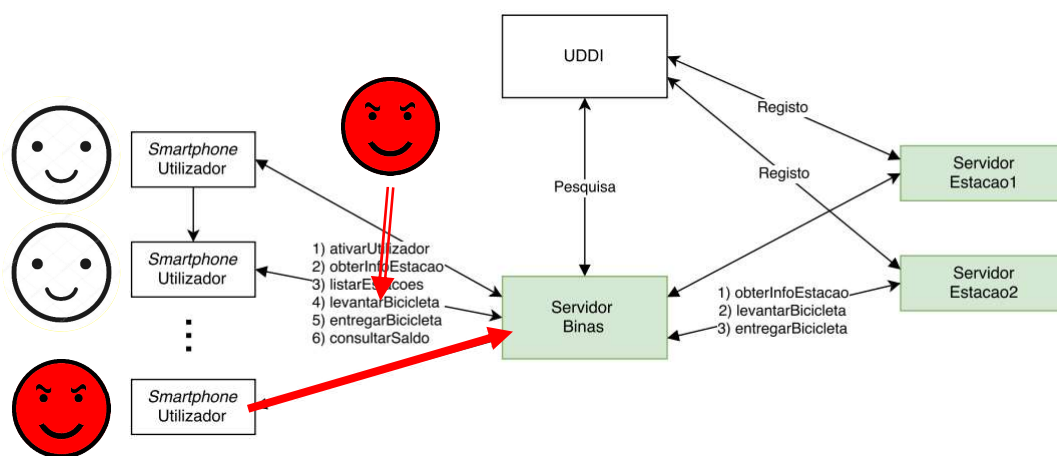


2017/18

SD




Parte 3



2017/18

SD




Departamento de Engenharia Informática

Modelo de *ameaças*

- Um utilizador malicioso pode invocar operações que afetem o saldo de um outro utilizador de forma ilegítima
- Um atacante pode adulterar as mensagens na rede

2017/18 SD



Departamento de Engenharia Informática

Política de segurança

- *Um utilizador malicioso pode invocar operações que afetem o saldo de um outro utilizador de forma ilegítima*
 - **Autenticar** as invocações feitas pelo cliente ao servidor Binas
 - Apenas **autorizar** pedidos que dizem respeito ao utilizador autenticado
 - *Está fora do âmbito a comunicação entre o Binas e as estações*
- *Um atacante pode adulterar as mensagens na rede*
 - A **integridade** das invocações – pedido e resposta – deve ser garantida
 - *O pedido e a resposta podem seguir em claro, ou seja, não se exige que pedido/resposta sejam confidenciais*

2017/18 SD

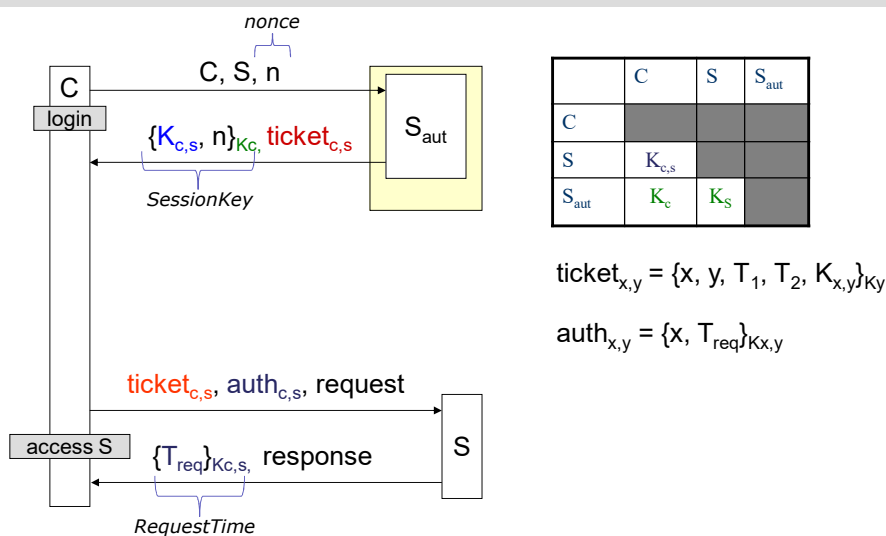


Mecanismos de segurança

- Autenticação de utilizadores
 - Cada utilizador tem uma conta e uma senha partilhada com o Kerberos
 - O Kerberos autentica os utilizadores e gera chaves de sessão
- Controlo de acessos
 - A autenticação é fresca
 - Confirmar que o email dos pedidos corresponde ao do utilizador autenticado
- Integridade dos pedidos e das respostas
 - Adicionar um MAC (*Message Authentication Code*) às mensagens SOAP
 - Usar a chave de sessão do Kerberos para o MAC




Kerby: simplified Kerberos



	C	S	S _{aut}
C			
S	K _{c,s}		
S _{aut}	K _c	K _s	

$$\text{ticket}_{x,y} = \{x, y, T_1, T_2, K_{x,y}\}_{K_y}$$

$$\text{auth}_{x,y} = \{x, T_{\text{req}}\}_{K_{x,y}}$$




Departamento de Engenharia Informática

Valorização

- A terceira parte vale 6 valores em 20, da seguinte forma:
 - Qualidade da estrutura base (20%)
 - **Autenticação de utilizadores (20%)**
 - **Controlo de acessos (20%)**
 - **Integridade de pedidos (20%)**
 - Relatório e demonstração (20%)

Sistemas Distribuídos 2018 9




Departamento de Engenharia Informática

Etapas de concretização

1. Preparar
2. Desenhar
3. Implementar
4. Demonstrar

2017/18 SD




Departamento de Engenharia Informática

Etapas de concretização: Preparar

1. Preparar
 - a) Criptografia em Java
 - Ver página dos laboratórios
 - b) Protocolo Kerberos simplificado
 - Ver *slides* das teóricas
 - c) Intercetores de mensagens SOAP
 - Ver página dos laboratórios sobre *JAX-WS Handlers*

2017/18 SD




Departamento de Engenharia Informática

Etapas de concretização: Desenhar

2. Desenhar
 - a) Para cada mecanismo de segurança, definir quem faz o quê
 - Autenticação com o Kerberos
 - Controlo de acessos
 - MAC
 - b) Escrever proposta da solução no RELATÓRIO e **validar com professor**
 - Detalhes na secção 4.7 do enunciado, página 9

2017/18 SD




Departamento de Engenharia Informática

Etapas de concretização: **Implementar**

3. Implementar

- Criar programa para testar a biblioteca e o servidor Kerberos
 - Classe de teste / classe executável - *main()*
 - Parte “cliente”: autentica-se e recebe chave de sessão e *ticket*; cria autenticador
 - Parte “servidor”: abre e valida *ticket*, valida autenticador
- Criar *handlers* Kerberos (na biblioteca ws-handlers)
 - Cliente
 - Servidor
- Criar *handlers* de segurança (na biblioteca ws-handlers)
 - Autorização – email do pedido corresponde ao utilizador autenticado?
 - MAC – protege mensagens de saída, valida mensagens de chegada

2017/18 SD



Departamento de Engenharia Informática

Etapas de concretização: **Demonstrar**

4. Demonstrar

Construir exemplos e fazer GUIÃO de DEMONSTRAÇÃO

- S1 – funcionamento normal da segurança
- S2 – resistência a um ataque

Detalhes na secção 4.8 do enunciado, página 9

2017/18 SD



Calendário de aulas de laboratório

Mês	2a	3a	4a	5a	6a	2a	3a	4a	Laboratório
fev	19	20	21	22	23				Java, Maven & Eclipse
fev/mar	26	27	28	1	2				Sockets
mar	5	6	7	8	9				Sun RPC (mE1)
mar	12	13	14	15	16				Java RMI (mE2)
mar	19	20	21	22	23				Web Services (mE3)
mar	F	F	F	F	F				Férias da Páscoa
abr	2	3	4	5	6				Web Services II: UDDI
abr	9	10	11	12	13				Apoio (P1)
abr	16	17	18	19	20				Web Services III: timeouts, one-way, asynchronous calls [atualizado]
abr/mai	23	24	F	26	27	30	F	2	Apoio (P2)
mai				3	4	7	8	9	Criptografia
mai				10	11	14	15	16	Web Services IV: handlers
mai				17	18				Apoio (P3)
mai	21	22	23	24	25				Discussões
mai/jun	28	29	30	F	1				

Devido aos feriados vamos ter **desdobramento** das semanas


e **horários** de dúvidas **extra**, junto às datas de entrega




Questões...

1. Preparar
2. Desenhar
3. Implementar
4. Demonstrar

Departamento de Engenharia Informática




Continuação de bom trabalho!



2017/18 SD 17

Departamento de Engenharia Informática



Canal de comunicação seguro
Em que nível implementar?

Sistemas Distribuídos 2018 18



Hipótese A: cifrar antes dos *stubs*

- Ou seja:
 - Cifrar valor do(s) argumentos
 - Chamar função remota passando os argumentos cifrados
- O que se perde?
 - *Stub* deixa de ser capaz de tratar a heterogeneidade do parâmetro cifrado
 - Ou seja, perdemos uma grande vantagem dos sistemas de RPC!
 - Tratar a heterogeneidade automaticamente nas funções de adaptação - *stub*
- Logo, a cifra tem de ser feita abaixo do *stub*...
 - Mas convém que seja dentro do mecanismo de RPC para garantir segurança de extremo-a-extremo (*end-to-end*)



Hipótese B: usar HTTPS como transporte

- Usar HTTPS como transporte, em vez de HTTP
- O que perco?
 - Se a mensagem SOAP tiver intermediários, estes conseguem acesso aos dados em claro
 - Todo o conteúdo da mensagem é cifrado
 - Mesmo as partes que não são confidenciais
- Logo, a cifra deve acontecer abaixo do *stub* mas acima do protocolo de transporte...

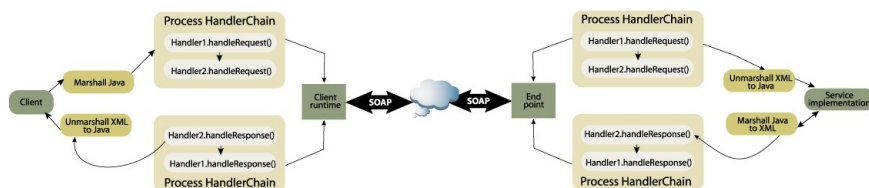


Hipótese C: cifrar num *Web Service handler*

- Abaixo do *stub*
- e
- Acima do protocolo de transporte




Relembrando: JAX-WS *Handlers*



- *Handler Chain*
 - Sequência de *handlers* executados sobre pedidos e respostas
- *Handler*
 - Estende a classe
 - `javax.xml.ws.handler.Handler`
 - Métodos relevantes
 - `handleMessage(MessageContext context)`
 - `handleFault(MessageContext context)`

Departamento de Engenharia Informática



Exemplo JAX-WS handler de segurança

```

public boolean handleRequest(MessageContext context) {
    System.out.println("handleRequest(MessageContext=" + context + ")");

    SOAPMessageContext smc = (SOAPMessageContext) context;
    SOAPMessage msg = smc.getMessage();
    SOAPPart sp = msg.getSOAPPart();
    SOAPEnvelope se = sp.getEnvelope();
    SOAPBody sb = se.getBody();
    SOAPHeader sh = se.getHeader();
    if (sh == null) { sh = se.addHeader(); }

    QName svcn = (QName) smc.get(MessageContext.WSDL_SERVICE);
    QName opn = (QName) smc.get(MessageContext.WSDL_OPERATION);

    if (!opn.getLocalPart().equals(OPERATION_NAME_TO_CIPHER)) {return;}

```


Obter conteúdo da mensagem SOAP

Obter o nome do serviço e da operação

Se a operação não contém informação sensível então não será cifrada

Sistemas Distribuídos 2018 23

Departamento de Engenharia Informática



Exemplo JAX-WS handler de segurança

```

NodeList children = sb.getFirstChild().getChildNodes();

for (int i = 0; i < children.getLength(); i++) {
    Node argument = children.item(i);
    if (argument.getNodeName().equals(NAME_OF_SECRET_ARGUMENT)) {
        String secretArgument = argument.getTextContent();

        // cipher message with symmetric key
        ByteArrayOutputStream byteOut = new ByteArrayOutputStream();
        byteOut.write(secretArgument.getBytes());
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, MyKeyManager.getSecretKey());
        byte[] cipheredArgument = cipher.doFinal(byteOut.toByteArray());

        String encodedSecretArgument = printBase64Binary(cipheredArgument);

        argument.setTextContent(encodedSecretArgument);
        msg.saveChanges();
    }
}

return true;

```

Obter os argumentos (Nodes) da mensagem


Para cada nó, verifica-se se corresponde ao argumento que é preciso cifrar

Cifra o argumento com informação sensível

Converte texto cifrado para Base 64

Atualiza-se a mensagem SOAP

Sistemas Distribuídos 2018 24




Departamento de Engenharia Informática

Texto cifrado em formato binário...

Como enviar texto cifrado em XML/SOAP?

Sistemas Distribuídos 2018 25



Departamento de Engenharia Informática

Codificação de Base 64

- Representa dados binários em texto
- Usa um subconjunto de 64 caracteres do ASCII que são os caracteres mais “universais”
 - Caracteres que são iguais em praticamente todos os códigos:
 - A-Z, a-z, 0-9, +, /
- Caracter ‘=’ usado no final para identificar quantidade de enchimento (*padding*) requerido
- Aumenta tamanho do conteúdo... Qual o sobrecusto (*overhead*)?
- Fundamental para sistemas baseados na comunicação em texto
 - Como os *Web Services*, *Email*, ...

Sistemas Distribuídos 2018 26



Exemplo de codificação em base 64

Text content	M	a	n
ASCII	77 (0x4d)	97 (0x61)	110 (0x6e)
Bit pattern	0 1 0 0 1 1 0 1	0 1 1 0 0 0 0 1	0 1 1 0 1 1 1 0
Index	19	22	46
Base64-encoded	T	W	u

Octetos transformados em grupos de 6 bits ($2^6 = 64$)

Overhead = $4/3 = +33\%$