Gestão de identidades centradas no utente

Rafael José Santos Simões
rafaeljsimoes@ua.pt

26/06/2021

Contents

1 Introdução

Este projeto baseia-se numa mudança protocolar relativamente ao primeiro projeto desta cadeira. Neste caso, o objetivo é o utilizador ter controlo ativo sobre os atributos de entidade usados durante os processos de autenticação, isto é, este tem plena noção que atributos de identidade estão a ser requisitados (por um *Service provider* (SP)) e o resultado da sua resolução (providenciado por um *identity provider* (IDP)). Complementarmente à amostrem explicita dos atributos de entidades, o utilizador consegue tomar de decisões de consentimento (ou não) relativamente ao conjunto de atributos solicitados e resolvidos pelas entidades de autenticação. Outra caraterística vantajosa deste protocolo é a impossibilidade dos *IDPs* conseguirem fazer *track* de quais *SPs* o utilizador interage.

2 Formato usado para os pedidos e respostas de atributos de entidade

Sabendo que no primeiro projeto foi usado o padrão de troca de dados de autenticação e autorização entre entidades - ***SAML***, devido à natureza ativa do utilizador nos atributos de entidade e da flexibilidade na definição dos atributos de entidade a usar por parte do *SP*, o padrão *SAML* deixou de ser utilizável, uma vez que o *SP* não consegue especificar os atributos de identidade (no pedido de autenticação) que este necessita, pois isto é estabelecido pelo intercâmbio de meta-dados entre o *SP* e o *IDP* onde estes negoceiam a forma como vão interagir.[1] Além disso, uma vez que as interações de *SAML* eram apenas realizadas entre o *IDP* e o *SP*, os atributos de entidade usados poderiam nunca ser apresentados ao utilizador caso o *IDP* não partilhasse essa informação.

Como é expectável foi criado um novo formato de pedido e resposta para partilha de atributos de entidade.

2.1 Formato do pedido de atributos de entidade

- **sp_info**: Informação relativa ao *SP*
  - **id**: Valor que armazena o identificador do *SP*
  - **location**: *URL* do *SP* para receber os atributos pedidos e a respetiva assinatura
- **identity_attributes**: Lista de atributos requisitados pelo *SP*
- **idp_info**: Informação relativa ao *IDP*
  - **id**: Valor que armazena o identificador do *IDP*
  - **location**: *URL* do *IDP* que permite a obtenção dos atributos recebidos

```
{
    "sp_info": {
      "id": "http://localhost:8081",
      "location": "http://localhost:8081/receive_identity_attributes"
    },
    "identity_attributes": [
      "username",
      "email"
    ],
    "idp_info": {
      "id": "http://localhost:8082",
      "location": "http://localhost:8082/handle_identity_request"
    }
  }

```

Listing 1: Exemplo de um pedido de atributos de entidade

[1] Apesar de não ser possível de utilizador o padrão SAML na totalidade, ainda é possível usar este padrão para o manuseamento de respostas de autenticação. Contudo, o padrão SAML foi completamente removido deste projeto, pois para possibilitar o uso deste padrão era necessário a *helper application* interpretar pedidos usando o SAML (que não foi implementado no projeto 1)

2.2 Formato da resposta a um pedido de atributos de entidade

- **attributes**: Dicionário que armazena o mapeamento entre os atributos solicitados e o respetivo valor
- **signature**: Atributo que armazena a assinatura (em base64) dos atributos de entidade. Esta assinatura é realizada pelo *IDP* usando a sua chave privada.

```
1       {
2           "attributes": {
3             "username": "rafael",
4             "email": "rafael@ua.pt"
5           },
6           "signature": "4UgFUEe+
    ↪ u3uauUnC2NzIbbzhe32p7rOdZCnTVaeynvvSIHEibasiozVv9nk0EsKo9Em4bqp0ms3WHrpC9C6gej
7           SYvATUAO0IhVgVKX1mQnN3vgzJ1bCULa7rr7uVXxGL0N1domUPDuUdhjN8c32TB1CGBtI11G8qUQGfrn8APDc="
8         }
9
```

Listing 2: Exemplo da resposta ao pedido 1 de atributos de entidade

3 Mudanças na arquitetura relativamente ao primeiro projeto

Como já foi referido, o utilizador tem poder ativo no que toca ao consentimento (ou não) no uso de atributos de entidade que lhe pertencem, por isso, o *workflow* moldado à arquitetura do primeiro projeto terá que sofrer alterações para permitir que *helper application* possa intercetar os pedidos e respostas de identificação para esta conseguir:

- Apresentar ao utilizador que atributos de identidade foram requisitados pelo *SP*
- Recolher o consentimento do utilizador e em caso positivo encaminhar o pedido de identidade para o *IDP*
- Receber a resposta ao pedido de identidade, e apresentar ao utilizador que valores que foram devolvidos pelo *IDP*
- Recolher o consentimento do utilizador sobre os valores recebidos e em caso positivo encaminhar a resposta ao pedido de identidade para o *SP*.

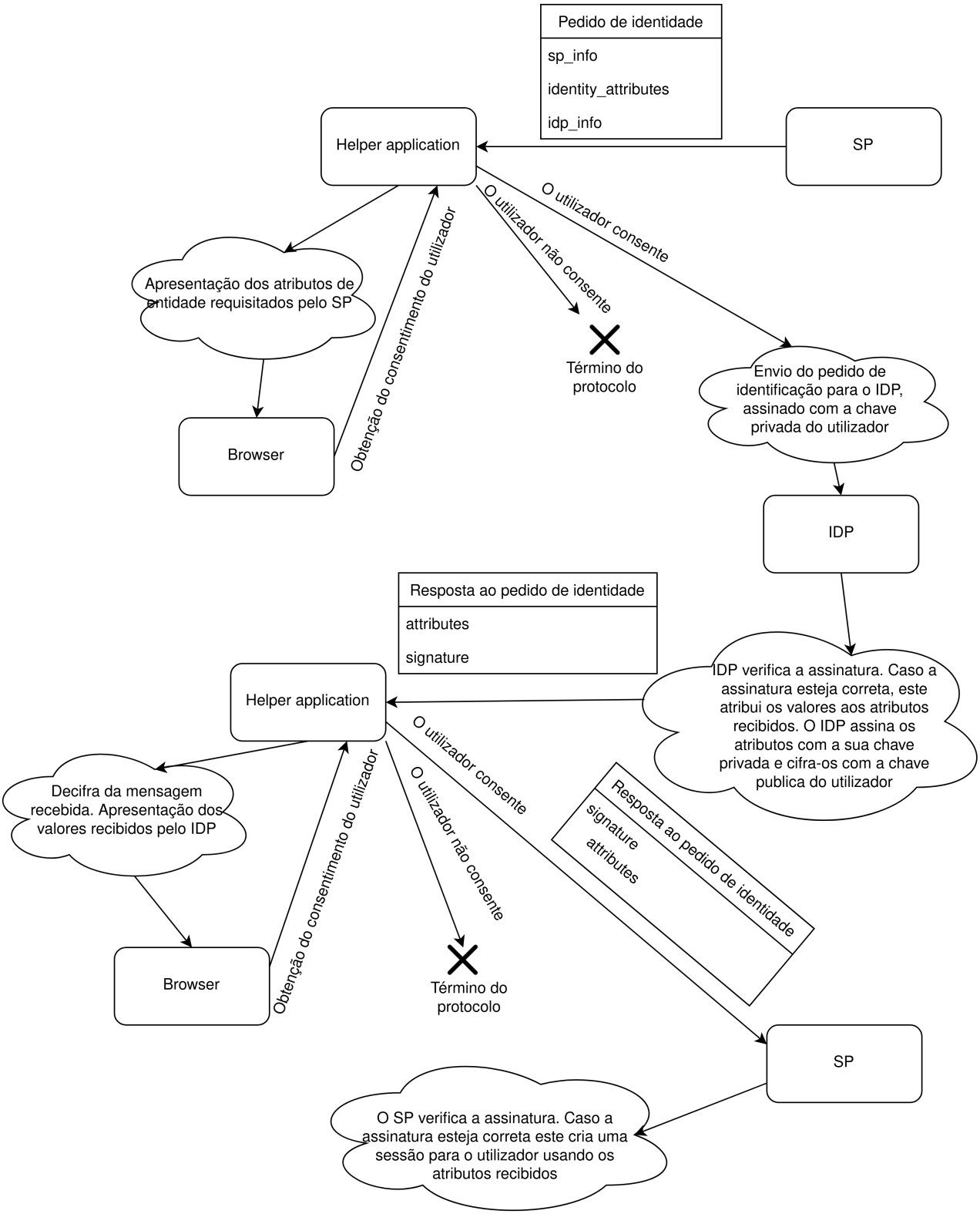
Como é fácil de perceber, a *helper application* neste sistema, funciona como entidade central por onde todos os pedidos e respostas passam e a qualquer momento, devido a uma ação de não consentimento do utilizador, esta pode cancelar o *workflow* deste protocolo cancelando quaisquer processos de identificação ativos.

Uma das caraterísticas desta nova arquitetura é a necessidade de existir uma par de chaves assimétricas entre o utilizador (através da *helper application*) e o *IDP* para proceder ao protocolo de identificação. No projeto anterior, existiam dois métodos de autenticação, um através do protocolo *ZKP* e outro usando um par de chaves assimétricas (geradas após um processo correto de autenticação usando o *ZKP*), naquele caso as credenciais assimétricas serviram para melhorar o processo de autenticação em termos temporais uma vez que o processo de *ZKP* é um processo consideravelmente lento. Neste projeto, o processo de autenticação com *ZKP* vai continuar a existir, quer para autenticar o utilizador quer para gerar um par de chaves assimétricas. Após a criação do material criptográfico assimétrico é possível realizar o *protocol collapsing* que num processo só permite realizar a autenticação do cliente e o fornecimento autenticado dos atributos de identidade. Portanto, qualquer explicação referente ao *protocol collapsing* será inferido que já existe um par de chaves assimétricas após o *ZKP*.

3.1 *Protocol collapsing*

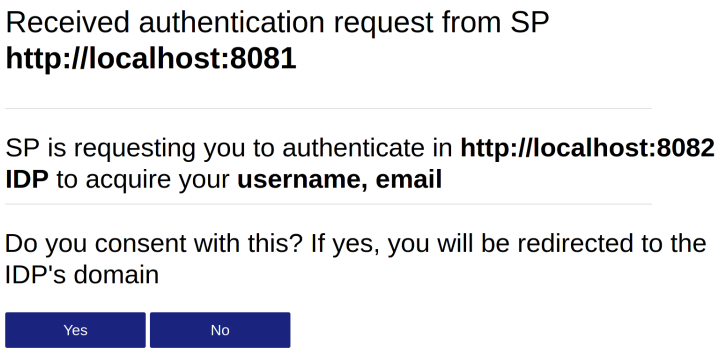
Neste secção vai ser explicado o *workflow* de operações necessários para realizar o processo de identificação de maneira segura neste serviço.

Figure 1: Protocolo de identificação usando o conceito de *Protocol collapsing*



1. O utilizador tenta aceder a um serviço disponibilizado por um *SP* ao qual não tem uma sessão ativa, portanto, o *SP* redireciona o utilizador para a *helper application* enviando uma mensagem com os atributos de identidade necessários para criar uma sessão.
2. A *helper application* apresenta os atributos requisitados pelo *SP* ao utilizador e obtém o consentimento do mesmo para conseguir enviar o pedido de identificação (com os atributos) para o *IDP*. Obviamente, caso o utilizador não permita o uso dos atributos especificados a *helper application* interrompe o processo e não envia o pedido para o *IDP* [2]

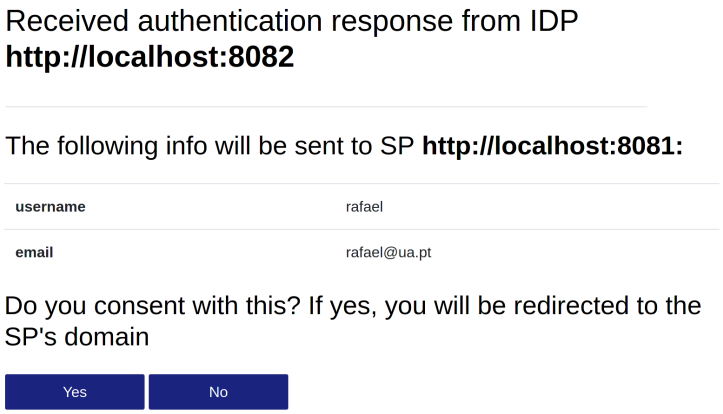
Figure 2: Página mostrada pela *helper application* para recolher o consentimento do utilizador antes do envio do pedido de identificação para o *IDP*



3. Após receber o consentimento do utilizador a *helper application* assina o pedido (usando a chave privada do utilizador, criada após o processo do *ZKP*) de identificação (enviado pelo *SP*) e envia o pedido e assinatura para o *IDP*
4. O *IDP* após receber a mensagem anterior, este verifica a assinatura com o conteúdo recebido. Caso a assinatura seja válida este sabe que a mensagem veio do utilizador pretendido e que o conteúdo da mensagem não foi adulterado (daí o termo *protocolo collapsing* pois permite numa verificação autenticar o utilizador e garantir que os dados não foram alterados no canal de comunicação). Depois da verificação da assinatura, o *IDP* obtém os atributos necessários e encapsula-os numa nova mensagem com o mapeamento entre os atributos recebidos e os valores obtidos. Antes de enviar a resposta para a *helper application* este faz uma assinatura sobre os conteúdos mapeados (com a chave privada associado ao *IDP*) e cifra o conteúdo da mensagem com a chave publica do utilizador.
5. A *helper application* faz o processo de decifra (usando a chave privada do utilizador) da resposta recebida e apresenta ao utilizador os valores retornados pelo *IDP*. Mais uma vez, obtém o consentimentos (ou não) do utilizador e caso este permita a *helper application* envia o conteúdo da resposta e a respetiva assinatura para o *SP*. Mais uma vez, caso não existe consentimento por parte do utilizador o processo é interrompido.

[2] Neste fase é considerado que já existe um par de chaves assimétricas, caso estas não existam o utilizador deve ser autenticado com o protocolo *ZKP* para de seguida serem criadas as credenciais assimétricas

Figure 3: Página mostrada pela *helper application* para recolher o consentimento do utilizador antes do envio da resposta para o *SP*



6. Por fim, o *SP* verifica a assinatura recebida com a chave publica do IDP e caso esta assinatura seja valida este consegue criar uma sessão para o utilizador usando os valores presentes na resposta recebida.

4 Extras

Nesta secção será relatado alguns mecanismos implementados para melhorar este caso de uso.

4.1 Túnel seguro

Para garantir a proteção no intercâmbio de mensagens foi usado a mesma estratégia de criação de um túnel seguro implementado no primeiro projeto. No primeiro projeto isto foi possível pois o *IDP* na primeira interação com o *helper application* enviava um segredo partilhado para que ambas as entidades conseguissem cifrar as mensagens enviadas usando algoritmos de criptografia simétrica, contudo, neste novo serviço isto já não é possível pois já não existe esta primeira interação do *idp* para a *helper application*, portanto, para que este segredo seja criado e partilhado é necessário a *helper application* fazer um operação explicita de obtenção deste segredo do *IDP*. Portanto, antes de iniciar qualquer comunicação de autenticação e/ou processo de identificação a *helper application* faz um *redirect* para um *endpoint* do *IDP* que por sua vez redireciona novamente para a *helper application* com o segredo partilhado enviado por argumento. Uma vez que estas operações de redirecionamento são seguras (devido ao uso de *https*) a partilha do segredo é segura.