Multi-method authentication architecture for a Linux system

Simão Reis 59575 Rafael Figueiredo 59863

23 de Dezembro de 2013

Conteúdo

Capítulo 1

Resumo

Neste primeiro capítulo resumimos as funcionalidades do sistema.

Multi-method authentication architecture, é um sistema desenvolvido que permite aos seus utilizadores, usando um Cartão do Cidadão (**CC**) registar-se e entrar via web num site alocado numa máquina Linux. Nele é permitido ao utilizador criar, renovar e editar as suas contas Linux nessa máquina, oferecendo um pin de acesso sempre que uma conta é criada ou renovada.

Este sistema usa módulos personalizados PAM para conseguir autenticar os utilizadores nas suas contas Linux, guardadas numa base de dados SQLite3, com o auxilio de um módulo NSS. O site web corre em apache2 e permite os utilizadores gerirem todas as suas tarefas. Estas devem seguir as políticas de segurança definidas por um gestor do sistema, estas escritas num ficheiro XML.

O sistema permite ainda aos seus utilizadores restringir o acesso às contas pela localização geográfica do acesso com o auxilio do sistema GeoIPLite da MaxMind.

Por último o módulo PAM destingue acessos seguros e inseguros, expirando os pins assim que a conta seja acedida por um canal inseguro.

Capítulo 2

Instalação

Neste capítulo apresentamos um tutorial de instalação do sistema. Os vários passos a seguir, desde os ficheiros a usar, em que directórios devem residir, configuração de permissões entre outros.

2.1 Apache2 Web Server

No repositório online encontram-se gerados os certificados necessários para montar o sistema. Deve-se começar por copiar o certificado da CA e o certificado do WebServer e a sua chave privada.

```
1 cp certs/SecurityCA.crt /etc/ssl/certs/
2 cp certs/WebServer.crt /etc/ssl/private/
3 cp certs/WebServer.key /etc/ssl/private/
```

Activar o módulo SSL.

```
1 a2enmod ssl
```

Copiar o ficheiro de configuração SSL do servidor.

```
1 cp conf/default-ssl /etc/apache2/sites-enabled/
```

Copiar a cadeia de certificação das entidades certificadores dos **CCs** de forma a que o servidor confie em autenticações SSL com os crtificados dos

CCs.

```
mkdir /etc/ssl/certs/PTEIDCC/
cp certs/PT.pem /etc/ssl/certs/PTEIDCC/
```

Agora podemos copiar todos os scripts php para instalar o sistema web.

```
cp -r www/ /var/www/
```

Agora queremos que o servidor redirecione as ligações HTTP para HTTPS e que bloqueie os acessos URL aos ficheiros não php e não css.

```
cp /etc/apache2/mods-enabled/rewrite.load /etc/apache2/mods-enabled
   /
cp conf/httpd.conf /etc/apache2/
```

Agora queremos instalar e activar o SQLite3 com o PHP.

```
apt-get install php5-sqlite
cp conf/sqlite3.ini /etc/php5/conf.d/
```

Instalar a nossa base de dados e ficheiro xml de configuração para o gestor do sistema.

```
mkdir /etc/www/
cp -r db/ /etc/www/db/
chown www-data:www-data /etc/www/db/users.sqlite
cp -r manage/ /etc/www/manage/
```

Por útlimo reiniciar o servidor apache.

```
service apache2 restart
```

2.2 Web Client

Para poder usar o **CC** no web browser do cliente deve carrregar a driver /usr/local/lib/libpteidpkcs11.so. Para poder usar o sistema remoto deve confiar no conf/SecurityCA.pem, abrindo o ficheiro no web browser.

2.3 PAM, NSS e GeoIP

Para instalar o GeoIP deve-se começar pro descarregar a framework do site http://geolite.maxmind.com/download/geoip/api/c/GeoIP.tar.gz. De seguida executar os seguintes comandos.

```
tar zxvf GeoIP.tar.gz
cd GeoIP-1.4.8/
apt-get install libtool # Only if it cannot install correctly.
libtoolize -f # Only if it cannot install correctly.
./configure
make
make install
```

Por fim é presiso instalar as bases de dados.

```
mkdir /usr/local/share/GeoIP/ # In case the directory is missing.
cp -r geoip/ /usr/local/share/GeoIP/
```

Os passos necessários para colocar o nosso módulo PAM e módulo NSS nos sitios correctos e efectivamente chamar estes módulos nos ficheiros de configuração resume-se ao seguinte.

Começar por instalar a biblioteca C do SQLite3.

```
apt-get install libsqlite3-dev
```

Compilar e instalar os nossos módulos pam.

```
cd pam/
make
make install
cd ../
```

Copiar os ficheiros de configuração para os directórios respectivos.

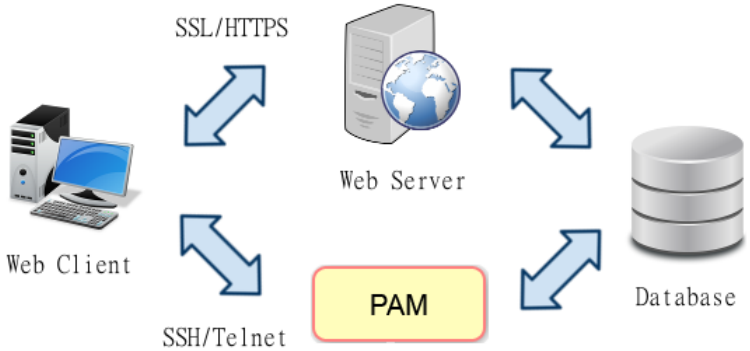
```
cp conf/nsswitch.conf /etc/
cp conf/common-auth /etc/pam.d/
cp conf/common-account /etc/pam.d/
cp conf/common-session /etc/pam.d/
```

Capítulo 3

Arquitectura

Neste capítulo descrevemos a arquitectura geral do sistema, a ligação entre os vários nós do sfotware.

O sistema é composto por um cliente na web, que deverá conter um cliente web e um cliente ssh ou telnet. No outro lado temos o servidor remoto Linux que contém um web server, módulos personalizados PAM e uma base de dados partilhada entre o Web Server e o módulo PAM.



O utilizador acede remotamente através de uma ligação segura SSL/HTTPS ao web server de forma a gerir as suas contas Linux. O Web Server usa a base de dados para fazer persistência das contas dos utilizadores do website e das contas Linux. Depois para o utilizador para aceder à sua conta usa um cliente ssh ou telnet. A ligação é autenticada através do módulo PAM que faz uso da Base de Dados.

3.1 Políticas de Segurança

O sistema apenas considera como ligaçõ segura autenticação sshd, sudo e su. Qualquer outro acesso é considerado é inseguro e leverá que a conta Linux seja expirada. A dimensão dos pins e algoritmos de hash são escolhidos e geridos pelo gestor do sistema localmente num ficheiro XML. O utilizador para navegar nas páginas seguras do webserver deverá manter inserido o **CC** em todo o momento. Para isso todas as ligações HTTP são redirecionadas para ligações HTTPS. O utilizador apenas pode aceder por URL a ficheiros php e css. Qualquer outro tipo de aceddo a ficheiros armazenado no web server é bloqueado. Na versão final do sistema não se encontram outros tipo de ficheiros no web server. Quando um utilizador entra pela primeira vez numa conta Linux gerada pelo web server, é gerada o seu directório home com confinamento apropriado para que apenas o próprio utilizador possa entrar e manipular esse directório.

Capítulo 4

Base de Dados SQLite3

Neste capítulo apresentamos a base de dados partilhada entre os módulos PAM, NSS com o servidor apache2.

4.1 Tabelas

A primeira tabela users é a dos utilizadores da web app. Possui o username do utilizador, a hash da sua password escolhida, o serial que é o número do Bilhete de Indentificação (**BI**) do **CC** e o salt da password. Tanto o username como a password eram campos não estritamente necessários mas o grupo decidiu coloca-los para que o acesso às contas não dependesse apenas da posse do **CC** e do conhecimento do seu pin.

```
1 CREATE TABLE users (
2     name VARCHAR(16) PRIMARY KEY NOT NULL,
3     pass TEXT NOT NULL,
4     salt VARCHAR(21) NOT NULL,
5     serial VARCHAR(16) NOT NULL
6 );
```

A tabela passwd representa a lista de contas Linux criadas pelos utilizadores do nosso sistema. Têm naturalmente um uid e gid associado que identifica de forma única a conta, o username que é o nome da conta, a hash

do pin de acesso e seu salt correspondente. Contém ainda o caminho absoluto do directório home da conta, o path para a shell, a data de expiração e duas flags. A expflag indica que a conta já expirou por qualquer motivo. A retrycount conta o número de acesso falhados. À terceira tentativa falhada a flag expflag é activada.

```
CREATE TABLE passwd (
    uid INT PRIMARY KEY NOT NULL,
    gid INT NOT NULL,
    username VARCHAR(16) NOT NULL UNIQUE,
    password TEXT NOT NULL,
    salt TEXT NOT NULL,
    home TEXT NOT NULL,
    bash TEXT NOT NULL,
    expdate BIGINT NOT NULL,
    expflag INT NOT NULL,
    retrycount INT NOT NULL,
    FOREIGN KEY(gid) REFERENCES groups(id)
);
```

A tabela user accounts associa as contas temporárias com os seus donos. O campo username é o nome do utilizador do web site enquanto a account é o nome da conta em si.

```
CREATE TABLE useraccounts (
    username VARCHAR(16) NOT NULL,
    account VARCHAR(16) NOT NULL,
    PRIMARY KEY(username, account),
    FOREIGN KEY(username) REFERENCES users(name),
    FOREIGN KEY(account) REFERENCES passwd(username)
);
```

A tabela groups representa os grupos do nosso sistema. São identificados unicamente por um id, têm um nome e uma password. Neste momento não

estamos a usar a password de grupo pelo motivo que não consideramos a sua utilização útil no âmbito do nosso projecto. Apenas existe no momento um grupo, no entanto, existe a possibilidade de serem criados mais grupos e serem atribuidas diferentes permissões a cada.

```
1 CREATE TABLE groups (
2     id INT PRIMARY KEY NOT NULL,
3     name VARCHAR(16) NOT NULL,
4     password TEXT NOT NULL
5 );
```

A tabela geopermissions lista um conjunto de regras de acesso geográfico às contas Linux criadas pelo nosso sistema. O campo account é o nome da conta Linux, o country é o código do pais de 2 letras e a city é a cidade, em que caso não seja expecificada é prenchida com a string 'N/A'.

```
1 CREATE TABLE geopermissions (
2     account TEXT NOT NULL,
3     country VARCHAR(2) NOT NULL,
4     city TEXT NOT NULL,
5     PRIMARY KEY(account, country, city),
6     FOREIGN KEY(account) REFERENCES passwd(username)
7 );
```

Capítulo 5

Web Application

Neste capítulo descrevemos as funcionalidades permitidas na aplicação Web.

5.1 Estrutura Geral do Site

O sistema é fundamentalmente composto por uma zona segura e uma zona menos segura. Na zona menos segura estão as páginas home, login e register. As últimas 2 como o nome indica permitem o login e registo do sistema. Apos preencher as suas credenciais o utilizador dever inserir no seu smartcard reader o seu **CC** e ai carregar no botão para realizar a operação. Será posteriormente requisitado a inserir o pin do seu **CC**. Isto porque o servidor correr os scripts de autenticação dentro da zona segura que é definida pela necessidade de apresentar credenciais SSL. Obviamente caso alguns dos campos seja errado a operação é anulada.

Na zona segura estão para além dos scripts de autenticação, as páginas que permitem gerir as contas Linux: criar, remover, renovar. Também esta presente uma página para gerir as geo-permissões. O feedback do estado das contas é mostrado sobre a forma de uma tabela que contém as colunas com o nome da conta, a data de expiração e o estado (activa ou supensa). No caso das geo-permissões é mostrado o nome da conta Linux, o país e a cidade da regra.

Para gerir a lógica do sistema estão também presentes scripts que executam as tarefas, desde criar a eliminar contas.

Na página de new account, o utilizador define o nome da conta e caso não exista ainda nessa máquina, é feito refresh da página e apresentado o pin

gerado. Estes dados são transportados através do dicionário **$_SESSION** do php e são desactivos assim que a página é refrescada. Ou seja assim que o utilizador saia da página nunca mais poderá recuperar o pin. O uid da conta será o número seguinte do maior uid existente na Base de Dados antes da conta ser criada.

Na páginas my accounts para além de ser dado o feedback das contas sobre a forma de tabela ainda é disponibilizado dropdown lists que permite ao utilizador escolher uma conta para eliminar ou renovar. Uma conta só pode ser renovada caso esteja suspensa o que reparamos que em alguns casos pode ser incómodo pois não toma em conta de facto que o utilizador pode esquecer-se do pin, sendo obrigado a expirar a conta de alguma forma para voltar a renova-la. No fim ao selecionar a conta para renovar o mecânismo é o mesmo de quando uma contra é criada e o pin é mostrado numa página temporária.

Na página my permission tem para além do feedback das regras em vigor do acesso geográfico formulários idênticos aos da página mya accounts para adicionar e remover regras. Para adicionar uma regra deve explicitar o código de duas letras do país e o nome da cidade é um campo facultativo, sendo apresentado com a string 'N/A' na tabela caso não seja preenchido.

5.1.1 Mecânismos de Segurança

Toda a informação do browser de formulários javascript são passados pelo método POST do protocolo HTTP. Os scripts executam comandos SQL semi-hardcoded para manipular a base de dados. Para prevenir ataques de SQL Injection, todos os comandos que usam o dicionário global **$_POST** do php, são pré processados de forma a que qualquer conteúdo seja tratado apenas como texto puro e não como statments SQL.

5.2 Configuração XML

Como referido no resumo, as politicas de segurança do site não são hardcoded, estão gravadas num ficheiro XML na máquina local que o gestor do sistema pode mudar a qualquer momento. Embora o sistema está preparado para a sua ausência definindo políticas por omissão.

No elemento permited podem ser listados vários elementos user. Estes listam os utilizadores permitidos a criar e renovar contas. O elemento pin

contém dois sub-elementos. O length é o tamanho dos pins gerados e oferecidos aos utilizadores quando criam ou renovam uma conta. O elemento days indica quantos dias os pins devem durar a partir do instante que são criados. O elemento max indica o número máximo de contas que cada utilizador pode criar. Por último o elemento salt contém dois sub-elmentos. O elemento length é o comprimento do salt. O campo algorithm define o código que a função crypt() do Unix deve usar para gerar a hash do pin.

Capítulo 6

Módulos PAM e NSS

Para garantir uma autenticação correcta das contas Linux criadas no nosso Sistema foi necessário construir um novo módulo PAM e um novo módulo NSSwitch. O módulo PAM criado sabe como aceder à nossa base de dados e verificar se a conta pertence ao nosso sistema, se a password está correcta e se a conta é válida. O módulo NSSwitch criado também acede à nossa base de dados para ler informação e está responsável por toda a representação de dados do utilizador durante a sua sessão. Estão incluidos serviços como o naming uid para nome e vice-versa, listagem dos grupos a que o utilizador pertence,etc, tudo o necessário a que o utilizador tenha todos os dados para se identificar disponíveis.

6.1 PAM

A autenticação de um utilizador por parte da PAM é dividida em quatro fases com funções distintas. A auth verifica se o utilizador existe, faz o prompt da password e compara com a password guardada. A fase account verifica se o utilizador está em condições de ser autenticado, aqui são colocadas restrições de acesso. A fase session monta todas as variáveis de sessão necessárias. A fase password serve para efetuar alterações na password, etc. Para facilitar o entendimento dos retornos dos métodos PAM:

PAM_SUCCESS
: Sufficient (Não corre mais nenhum módulo. Sucesso);

PAM_IGNORE
Die (Não corre mais nenhum módulo. Insucesso);

PAM_DEFAULT
Ignore (qualquer outro resultado é ignorado e o próximo módulo é carregado).

Foram implementadas as seguintes fases:

1. *Autenticação*

   Tal como referi cima nesta fase acedemos à nossa base de dados SQLite e verificamos se o utilizador a ser autenticado faz parte do nosso sistema. Caso faça, a entrada da base de dados desse utilizador é lida e a password é comparada. Se a password estiver certa verificamos se o utilizador tinha falhado a password anteriormente (retrycount $\neq 0$) e se falhou tornamos o valor de retrycount a zero e retornamos **PAM_SUCCESS**. Se a password está errada podem acontecer duas coisas. Ou o número de tentativas falhada é menor que dois e incrementa esse número ou então a conta é expirada por excesso de tentativas falhadas. Em ambos os casos o retorno é **PAM_IGNORE**. Se a conta não faz parte do nosso sistema é retornado **PAM_AUTH_ERR**.

2. *Account Management*

   Após verificarmos que a password está correcta temos que garantir que de facto a conta está activa e tem permissões para ser autenticada. Aqui verificamos se a conta está expirada. Ela pode ter expirado por passar do prazo de validade, por terem sido excedidas as três tentativas ou por ter sido acedidade por um canal não seguro. Se a conta é válida so resta verificar as regras de acesso da mesma(GeoIP). Vou explicar o GeoIP mais tarde, por agora basta saber que se passar nas regras a conta é validada. O retorno do método é semelhante à fase autenticação. **PAM_SUCCESS** num caso bem sucedido, **PAM_IGNORE** se a conta não é valida e **PAM_AUTH_ERR** se a conta não faz parte do nosso sistema.

3. *Session*

Se o utilizador foi devidamente autenticado e a conta encontra-se valida so resta preparar a sessão. A maior parte do trabalho é feito pelos módulos ja existentes, o que eu faço é, por motivos de conveniência para o utilizador, alterar o diretório para a sua home (**chdir**) e alterar as permissões default dos ficheiros e diretórios que ele irá criar nessa sessão (**umask**). No decorrer destas acções ele acaba por verificar se, na base de dados, um path para home válido existe.

6.1.1 GeoIP

O GeoIP consiste na tradução de um IP para uma localização geográfica, país, região ou cidade. Um utilizador do nosso sistema pode colocar filtros(ou regras) nas suas contas Linux e exigir certas contas apenas possam ser acedidades a partir de certos locais, sejam eles países ou cidades (não incluimos regiões).

Existem três níveis de filtragem. Os que aparecem antes sobrepõem (ou tornam obsoletos) os que se encontram a seguir. São eles:

1. Não tem regras.

   Caso um utilizador não tenha definido regras significa que a nível de filtragem por localização geográfica qualquer pessoa pode entrar no sistema.

2. Tem regras definidas. Mas existe uma entrada com a string 'N/A' no campo city.

   Uma entrada tem 3 campos. Account, country e city. Ele pode ter várias entradas para cada país a permitir cidades diferentes. Se existir uma entrada 'N/A' para um certo país significa que essa conta pode ser acedida em qualquer lugar desse país e por isso se existirem outras entradas para esse país a definirem certas cidades, essas entradas tornam-se irrelevantes.

3. Tem regras definidas, com entradas nos campos country e city.

É o caso de acesso mais restrito. Apenas são permitidas entradas no sistema apartir das localizações especificadas nas regras. A filtragem é feita pelo par (País, Cidade) para um utilizador (Account).

6.2 NSS

Como referi anteriormente o NSS é responsavél pela tradução de nomes das contas e grupos Linux. Para uma execução correta do sistema, e como ele nativamente não sabe interpretar bases de dados SQLite, tivemos que criar um módulo novo que soubesse abrir a base de dados, recolher a informação, e coloca-la disponivél nas várias estruturas de dados que o sistema usa (passwd, group, spwd).

Foram implementados métodos para as seguintes componentes:

1. *Passwd*

   (a) Método para devolver uma estrutura passwd a partir de um nome de utilizador.

   (b) Método para preencher a estrutura passwd recebida por argumento (juntamento com o respectivo buffer) a partir de um userid.

   (c) Método para preencher a estrutura passwd recebida por argumento (juntamento com o respectivo buffer) a partir de um nome de utilizador.

2. *Groups*

   (a) Método para preencher a estrutura group recebida por argumento (juntamento com o respectivo buffer) a partir de um groupid.

   (b) Método para preencher a estrutura group recebida por argumento (juntamento com o respectivo buffer) a partir de um nome de grupo.

3. *Shadow*

(a) Método para preencher a estrutura spwd dado o nome de um certo utilizador. Esta estrutura contém a password do utilizador bem como o seu nome.

Capítulo 7

Outros

7.1 Features Não Implementadas

Encontram-se em comentário no script php que elimina contas Linux, os comandos bash para eliminar o directório na home pertencente à conta Linux. Por falta de tempo o grupo não configurou o servidor para ter permissões para executar os comandos.

7.2 Bugs Conhecidos

Parece que em alguns casos uma sessão ssh prende a base de dados (mesmo desligando o serviço nscd), não conseguindo assim o servidor realizar operações de escrita sobre ela. Este bug é um pouco estranho tendo em conta que a causa de lock na base de dados tinha sido resolvido(no nss estamos a abrir e fechar a ligação à base de dados dentro de cada método).

No telnet após errarmos a password três vezes seguidas, continua a a pedir a autenticação até que se acerte a password, embora como a conta já foi invalidade este nega o acesso. Ou seja sevia negar logo o acesso à terceira tentativa.

Capítulo 8

Bibliografia

Apresentamos as referências encontradas durante a nossa pesquisa para concretizar o sistema.

Referências

[1] btk. *how to enable sqlite3 for php?* URL: http://stackoverflow.com/questions/948899/how-to-enable-sqlite3-for-php/2528302#2528302.

[2] *Force Apache2 to redirect from HTTP to HTTPS.* URL: http://blog.edwards-research.com/2010/02/force-apache2-to-redirect-from-http-to-https/.

[3] Kenneth Geisshirt. *Development with Pluggable Authentication Modules.* URL: https://www.packtpub.com/article/development-with-pluggable-authentication-modules-pam.

[4] *How PAM works.* URL: http://www.tuxradar.com/content/how-pam-works.

[5] *PHP.* URL: http://php.net/.

[6] *PHP 5 Tutorial.* URL: http://www.w3schools.com/php/.

[7] Dag-Erling Smørgrav. *Pluggable Authentication Modules.* URL: http://www.freebsd.org/doc/en/articles/pam/article.html.

[8] *Understanding PAM.* URL: http://aplawrence.com/Basics/understandingpam.html.