Segurança de autenticação

Autenticação flexível de usuários com PAM

Aprenda a flexibilizar a autenticação de usuários através do PAM, ferramenta de autenticação baseada em hardware e software.

por Thorsten Scherf

Ana Vasileva – Fotolia.com

O PAM é um conjunto de módulos poderosos para trabalhar com autenticação de usuários baseada em software e hardware, dando ao administrador melhor poder de escolhe sobre os métodos de implementação.

Há inovações de hardware diárias para a autenticação de contas de usuários. Os *Pluggable Authentication Modules* (PAM) ajudam a integrar o sistema – com transparência –, esses novos dispositivos. Isso propicia aos administradores experientes a opção de oferecer uma variedade de métodos de autenticação a seus usuários enquanto oferecem possibilidades de controlar todo o fluxo de trabalho da sessão do usuário.

Antigamente

Os logins de usuário em sistemas Linux ficam tradicionalmente por conta dos arquivos `/etc/passwd` e `/etc/shadow`. Quando um usuário executa o comando `login` para entrar no sistema com um nome e senha, o programa cria um `checksum` criptográfico da senha e compara os resultados com o `checksum` armazenado no arquivo `/etc/shadow`. Caso os `checksums` coincidam, o usuário é autenticado; caso contrário, o login irá falhar.

Essa abordagem não acompanha bem a escala. Em ambientes maiores, as credenciais do usuário ficam, normalmente, armazenadas centralmente em um servidor LDAP, por exemplo. Nesse caso, o programa de login não recupera o checksum da senha do arquivo `/etc/shadow` mas, sim, de um serviço de diretório. Essa tarefa pode ser simplificada com o uso do PAM [1].

Autenticação modular

Originalmente desenvolvido em meados dos anos 90 pela Sun Microsystems, o PAM hoje está disponível na maioria dos sistemas baseados em Unix. O PAM se encarrega de todo o processo de autenticação desde o aplicativo em si até um conjunto central que compreende uma extensa coleção de módulos (**figura 1**). Cada um desses módulos cuida de uma tarefa específica; no entanto, o aplicativo só fica sabendo se o usuário conseguiu ou não se conectar. Em outras palavras, é tarefa do PAM encontrar um método adequado para autenticar o usuário. O PAM define a aparência do módulo e o aplicativo e nem percebe isso.

O PAM pode usar vários métodos de autenticação. Além dos tradicionais métodos de rede como o LDAP, o NIS ou o Winbind, o PAM pode usar bibliotecas mais recentes para acessar uma grande variedade de dispositivos de hardware suportando, assim, logins baseados em smartcards ou na impressão digital do usuário. Sistemas com senhas de única utilização, tais como o *S/Key* ou o *SecurID*, também são suportados pelo PAM, e alguns métodos até exigem um dispositivo Bluetooth específico para conectar o usuário.

O PAM funciona de modo simplificado. Cada aplicativo que utiliza o PAM (o aplicativo precisa estar ligado à biblioteca `libpam`) possui um arquivo de configuração separado na pasta `/etc/pam.d/`. Normalmente, o arquivo receberá o nome do próprio aplicativo `-login`, por exemplo. No arquivo, os módulos distribuem tarefas PAM entre si. Numerosas bibliotecas estão disponíveis em cada grupo, e cuidam de uma variedade de tarefas dentro do grupo (**figura 2**). Flags de controle permitem gerenciar o comportamento do PAM em caso de erro – por exemplo, se o usuário não fornece a senha correta ou se o sistema não pode verificar a impressão digital.

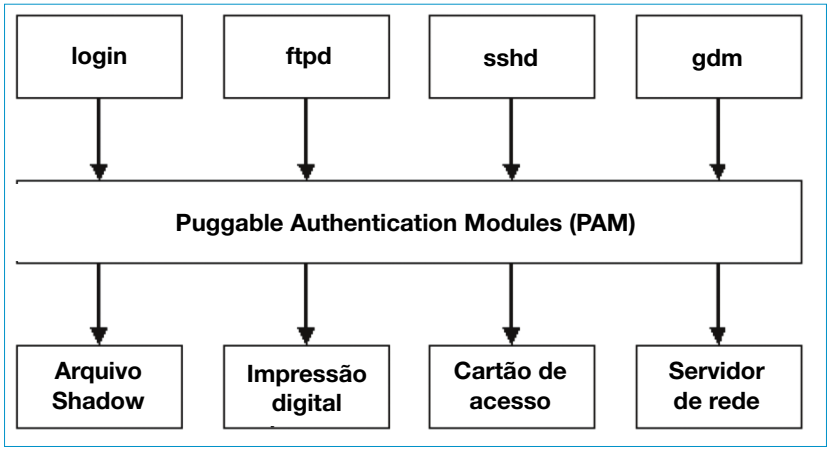

Figura 1 O PAM oferece um gerenciamento de usuários centralizado para o aplicativo.

Impressões digitais

Bibliotecas PAM mais recentes permitem que os administradores autentiquem os usuários através de smartcards, tokens USB ou recursos biométricos. Notebooks de ponta geralmente incluem um leitor de impressão digital que permite que o usuário utilize esse recurso ao se conectar ao sistema. A biblioteca PAM ThinkFinger [2] oferece o suporte necessário. De acordo com a documentação, o módulo irá suportar o leitor de impressão digital da UPEK/SGS Thomson Microelectronics, usado pela maioria dos notebooks Lenovo recentes e muitos dispositivos externos.

A maior parte das grandes distribuições Linux oferece pacotes para as bibliotecas PAM. É possível utilizar o gerenciador de pacotes da sua distribuição para instalar o software dos repositórios. Para instalar os pacotes necessários no seu disco rígido, use o comando `yum install thinkfinger` no sistema Fedora e os comandos:

```
apt-get install thinkfinger-tools
libpam-thinkfinger
```

no Ubuntu Hardy. Administradores do Gentoo podem usar o compacto comando `emerge sys-auth/thinkfinger`.

Se você estiver utilizando o openSUSE, precisará dos pacotes `libthinkfinger` e `pam_thinkfinger`, cujas versões presentes no repositório não estão atualizadas.

Pode ser preferível a instalação manual com os típicos passos `./configure`, `make`, `make install` e os arquivos da coleção de códigos fontes corrente. Usuários do Debian no Lenny precisarão acessar o repositório experimental e depois digitar para instalação:

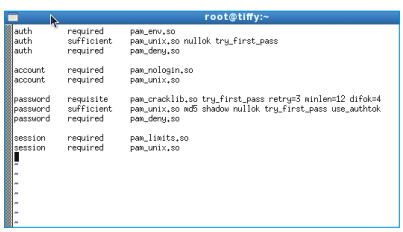

Figura 2 Um arquivo de configuração PAM clássico contém módulos e bibliotecas que o administrador pode usar para customizar o PAM.

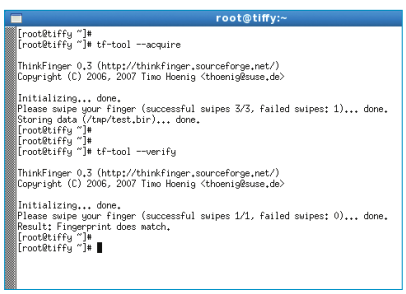

Figura 3 A ferramenta tf-tool oferece a opção de testar seu escaner de digitais.

```
aptitude install libthinkfinger0
libpam-thinkfinger thinkfinger
➥-tools
```

Antes de modificar a configuração do PAM, seria bom testar o próprio dispositivo. Para isso, escaneie sua impressão digital usando o comando `tf-tool --acquire` (**figura 3**).

Depois digite: `tf-tool --verify`.

Para verificar os resultados. É possível seja exibida a mensagem *Fingerprint does *not* match* (Impressão digital não combina). As primeiras tentativas podem ser pouco precisas, pois será preciso se familiarizar com o dispositivo.

Se arrastar seu dedo muito rapidamente ou muito lentamente pelo escaner, o dispositivo pode falhar na identificação da impressão digital. Nesse caso, haverá uma mensagem de erro e o fechamento. Ao conseguir resultados confiáveis no scaner de digitais, será possível excluir o arquivo temporário com o teste de escaneamento em `/tmp` e criar um arquivo individual para cada usuário no sistema que guardará a digital do usuário. O comando é `tf-tool --add-user username` (**figura 4**).

Os usuários precisam escanear suas digitais três vezes para que isso funcione. Caso a digital seja identificada corretamente a cada vez, a ferramenta irá armazená-la em um arquivo separado, em `/etc/pam_thinkfinger/`. Quando tudo estiver funcionando, é possível começar a configuração do PAM. Caso você queira uma autenticação utilizando o escaner de impressão digital primeiro, será preciso chamar o módulo `pam_thinkfinger` antes do `pam_unix`.

Para evitar que o PAM solicite uma senha após passar pelo teste da impressão digital, será preciso adicionar uma flag de controle chamada `sufficient`. Isso impede que o PAM chame qualquer outra biblioteca após um teste de autenticação bem sucedido (ou seja, considera a autenticação suficiente) e o faz devolver `PAM_SUCESS` ao programa `-login` neste exemplo. Se o login que utiliza impressão digital falhar, o `pam_unix` é chamado como um último recurso e irá solicitar a senha regular do usuário.

Entrar manualmente em todas as bibliotecas PAM de todos seus aplicativos que o utilizam, em cada arquivo de configuração, seria bem maçante. Um arquivo de configuração centralizada do PAM é uma alternativa. No Fedora ou no Red Hat, esse arquivo chama-se `/etc/pam.d/system-auth`, apesar de outras distribuições Linux o chamarem através de `/etc/pam/common-auth`. Nesse arquivo, é possível entrar em todas as bibliotecas que serão usadas para autenticar seus usuários, conforme disposto na **figura 5**.

A flag de controle `include` irá incluir o arquivo em todos os seus arquivos de configuração PAM. De agora em diante, isso faz com que todos os programas nas bibliotecas PAM listados pelo arquivo de configuração central permaneçam disponíveis nos arquivos individuais PAM, incluindo o módulo `pam_thinkfinger`.

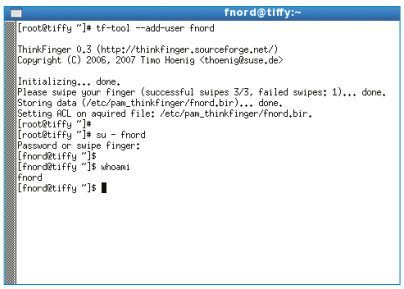

Figura 4 Criação de uma impressão digital para cada usuário.

Figura 5 No Fedora, o system-config-authentication é uma ferramenta de configuração PAM básica.

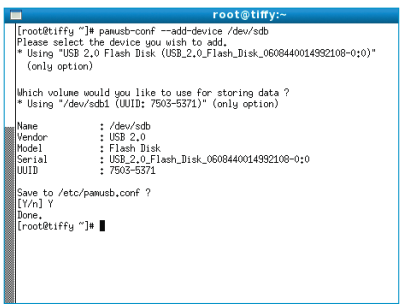

Figura 6 O dispositivo USB é identificado por suas propriedades. Se o usuário tentar se conectar sem o dispositivo, não irá conseguir.

Tokens USB

A biblioteca `pam_usb` suporta outra abordagem baseada em hardware, na qual o PAM verifica se um dispositivo USB específico está conectado à máquina. Em caso positivo, o usuário se conecta; caso contrário, o acesso ao sistema é negado. O dispositivo conectado é identificado por seu número serial único, modelo e nome do fabricante. Além disso, um número aleatório é armazenado no dispositivo USB e no computador; o número muda após cada login bem sucedido.

Quando um usuário se conecta, o PAM confere as propriedades do dispositivo USB específico e o número aleatório. Caso o número armazenado no USB não combine com o número no disco, o login falha. Isso evita que invasores roubem o número, coloquem-no em seu próprio dispositivo USB e depois modifiquem as propriedades de seus dispositivos para acessar o sistema. O número aleatório no sistema muda a cada

login, por isso o número roubado não irá combinar com o número no sistema.

Os Linux Gentoo e o Debian oferecem pacotes prontos dessa biblioteca PAM. Em ambos os casos, é possível usar o gerenciador de pacotes para instalar, como descrito para o `pam_thinkfinger`. Usuários de outras distribuições podem baixar o código fonte atual [3] e executar `make` e `make install` para compilar os arquivos necessários e instalá-los no sistema local. Então, será preciso conectar qualquer dispositivo USB – que pode ser um telefone celular com um cartão SD, por exemplo – e armazenar suas propriedades no arquivo `/etc/pamusb.conf`. O comando para isso seria `pamusb-conf --add-device USB-device-name` conforme ilustra a **figura 6**.

O comando `pamusb-conf --add-user user` permite que se adicione mais usuários à configuração e gera o número aleatório correspondente. O número de cada usuário é armazenado no dispositivo USB e no sistema. Além disso, a ferramenta adiciona cada usuário ao arquivo de configuração XML `/etc/pamusb.conf`. É possível usar o arquivo para definir ações para cada usuário; essas ações serão executadas quando o USB for conectado ou desconectado. Por exemplo, as linhas presentes no arquivo de configuração da **listagem 1** bloqueiam automaticamente a tela se o dispositivo USB for desconectado: então, é preciso adicionar a biblioteca PAM `pam_usb` ao arquivo de configuração PAM correspondente em `/etc/pam.d/system-auth` ou `/etc/pam.d/common-auth`. Caso a flag de controle `sufficient` seja usada, os usuários podem se conectar ao sistema conectando o dispositivo USB, assumindo que o número aleatório do usuário combina em ambos os dispositivos (**listagem 2**).

Para melhorar a segurança, é possível substituir a flag de controle `sufficient` por `required`. Essa alteração irá procurar primeiro o dispositivo USB, mas, mesmo que o dispositivo seja corretamente identificado, o PAM ainda irá pedir a senha ao usuário no próximo estágio do processo de login. Ambos os testes precisam ser bem sucedidos para que o usuário se conecte.

Listagem 1: Arquivo de configuração para pam_usb

```
01  <user id="tscherf">
02 <device>
03 /dev/sdb1
04  </device>
05  <agent event="lock">gnome-screensaver-command --lock</agent>
06  <agent event="unlock">gnome-screensaver-command --deactivate</agent>
07  </user>
```

Listagem 2: Autenticação baseada em dispositivo USB

```
01 tscherf@tiffy$ id -u
02 500
03 tscherf@tiffy$ su -
04 * pam_usb v0.4.2
05 * Authentication request for user "root" (su-l)
06 * Device "/dev/sdb1" is connected (good).
07 * Performing one time pad verification...
08 * Verification match, updating one time pads...
09 * Access granted.
10 root@tiffy# id -u
11 0
```

Todo os métodos de login baseados em hardware examinados até agora são de fácil configuração, mas todos têm pontos de vulnerabilidade, e é fácil falsificar uma impressão digital. Além disso, tokens USB podem ser roubados, colocando um fim a qualquer segurança que

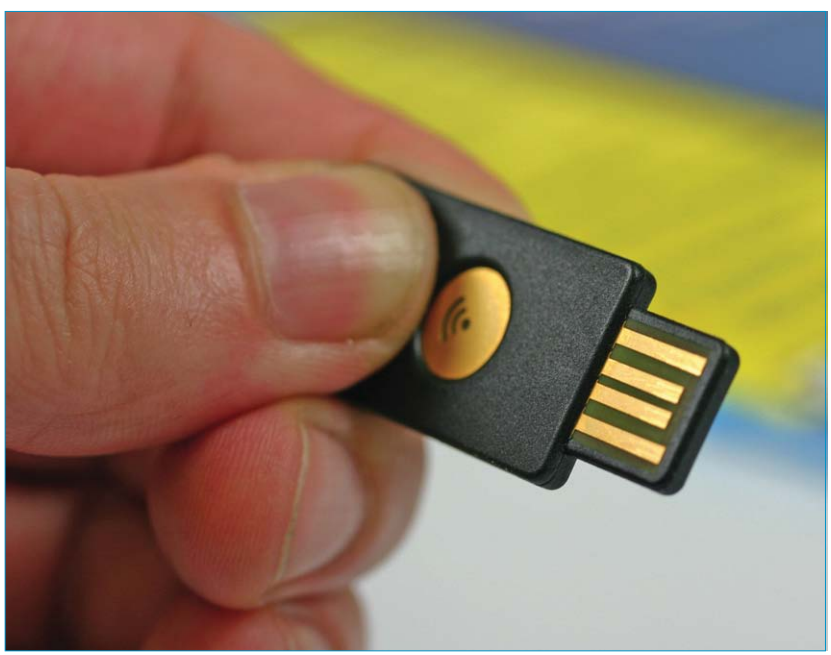

Figura 7 Emulação de teclado USB significa que o Yubikey contém senhas de uma única utilização e não precisa de drivers especiais. O token funciona com um simples apertar de botão.

ofereçam. Caso você leve segurança a sério, provavelmente irá querer usar uma autenticação de dois fatores. Esse método invariavelmente envolve o uso de *chip cards* com leitores ou tokens USB com senhas e PINs que são usados uma única vez.

Yubkey

Uma pequena empresa sueca, a Yubico [4], começou recentemente a vender Yubkeys (**figura 7**), que são pequenos tokens USB que emulam um teclado USB normal. O dispositivo possui um pequeno botão que, quando pressionado, faz com que o token envie uma senha usada apenas uma vez (OTP – *one-time password*) ao aplicativo ativo, tais como um prompt de login em um servidor SSH ou uma janela de login de um serviço web. A OTP é verificada em tempo real por um servidor de autenticação Yubico. O software foi lançado com licença de código aberto, por isso, teoricamente, é possível configurar seu próprio servidor de autenticação na sua LAN. Isso removeria a necessidade de uma conexão com a Internet.

O modo de funcionamento do token é bem simples. Em contraste com o popular token RSA, o Yubikey não necessita de bateria, pois a OTP não é gerada dinamicamente; em vez disso, as senhas são definidas com antecedência. As senhas são armazenadas no token e em um banco de dados no servidor de autenticação.

Ao apertar o botão do Yubikey, uma dessas OTP é enviada ao aplicativo ativo, que então usará uma API para acessar o servidor e verificar a senha. Se isso falhar (`Unknown Key`) ou se a senha já tiver sido usada (`Replayed Key`), uma mensagem de erro é enviada e o login falha. Se o servidor identificar a chave como válida, ele atribui o valor `1` à `usage-count` e o usuário é autenticado. O usuário não irá se conectar mais com essa chave.

Por causa da API simples, mais e mais aplicativos usam autenticação do servidor Yubico. Um exemplo é o plugin para o popular blog WordPress, que permite que usuários com Yubikey se conectem ao blog. Um projeto do *Summer of Code* do Google produziu um módulo PAM que suporta conexões com servidores SSH [5].

Em vez de digitar sua senha de usuário na tela de login, basta apertar o botão no Yubikey para enviar uma senha de 44 caracteres, com codificação *modhex* ao servidor SSH. O servidor, então, verifica a *string* com uma busca no servidor Yubico; os 32 caracteres restantes representam a senha de uma única utilização.

É possível definir um arquivo central no servidor SSH para especificar usuários com permissão de login produzindo uma Yubikey. Para isso, crie primeiramente um arquivo `/etc/yubikey-users.txt` com um nome de usuário, dois pontos (:) como separador e o ID Yubikey (os primeiros 12 caracteres da OTP do usuário) para cada usuário. Como alternativa, os usuários podem criar um arquivo (`~/.yubico/authorized_yubikeys`) com a mesma informação de seu diretório `home`.

Será preciso configurar o PAM para que este verifique a OTP no servidor Yubico. Para isso, adicione uma linha para o Yubikey em seu arquivo `/etc/pam.d/sshd`. A configuração mostrada na **listagem 3** executa essa autenticação, além do método de autenticação normal, com `system-auth`. Mas, se a flag `required` for substituída por `sufficient`, não há necessidade de o usuário fazer login depois que a OTP do Yubikey foi validada. Infelizmente, o Yubikey não está protegido por um PIN adicional, e o sistema fica vulnerável caso o token seja roubado. Um usuário não autorizado com um token poderia falsificar a identidade de terceiros. Os desenvolvedores estão trabalhando para acrescentar uma proteção PIN para as OTPs, e um patch não oficial já está disponível e em testes.

Listagem 3: Configuração do PAM para Yubikey

```
01 auth required pam_yubico.so authfile=/etc/yubikey-users.
02 txt
03 auth include system-auth
04 account required pam_nologin.so
05 account include system-auth
06 password include system-auth
07 session required pam_selinux.so close
08 session required pam_loginuid.so
09 session required pam_selinux.so open env_params
10 session optional pam_keyinit.so force revoke
11 session include password-auth
```

Listagem 4: Arquivo de configuração para pam_pkcs11

```
01 kcs11_module coolkey {
02 module = libcoolkeypk11.so;
03description = "Cool Key"
04slot_num = 0;
05ca_dir = /etc/pam_pkcs11/cacerts;
06nss_dir = /etc/pki/nssdb;
07crl_dir = /etc/pam_pkcs11/crls;
08crl_policy = auto;
09 }
```

Certificados 509 e o PAM

A clássica autenticação de dois fatores normalmente se baseia em cartões com chip. Os cartões tipicamente contêm um certificado protegido por um PIN. A biblioteca PAM `pam_PKCS11` permite que os usuários efetuem login no sistema através de um certificado X.509. O certificado contém um par de chaves privada/pública. Ambas podem ser armazenadas em um cartão com chip adequado, com a chave privada protegida por um PIN para evitar falsificação de identidade, simplesmente devido ao roubo do cartão. Para se conectar, é preciso ter o cartão e o PIN correspondente. Se o PIN for desconhecido, o login falha.

Os detalhes do processo de login são os seguintes: o usuário insere o cartão com chip no leitor e insere o PIN. O sistema busca o certificado com as chaves pública e privada do cartão. Se o certificado for válido, o usuário é mapeado no sistema. O processo de mapeamento pode recuperar uma variedade de informações do certificado, tipicamente o nome ou a UID armazenada no certificado. Para ter certeza de que o usuário é realmente quem diz ser, o sistema gera um número aleatório de 128 bits. Uma função no cartão com chip, em seguida, criptografa o número usando a chave privada, que também está armazenada no cartão. O usuário precisa digitar o PIN correto para acessar a chave privada. O sistema, em seguida, usa a chave pública disponível para quebrar o número criptografado. Se os resultados corresponderem ao número aleatório, o usuário é autenticado corretamente, pois as duas chaves combinam.

O hardware necessário para essa configuração é um cartão com chip e um leitor correspondente – por exemplo, o Gemalto e-Gate ou o dispositivo SCR USB da SCM. É possível usar qualquer token compatível com Java Card 2.1.1 ou Global Platform 2.0.1: os tokens Cyberflex da Gemalto estão amplamente disponíveis. Várias soluções de software também estão disponíveis: a abordagem descrita neste artigo baseia-se nos pacotes `pcsc-lite` e `pcsc-lite-libs` para acessar o leitor.

Infraestrutura de chave pública

Faz sentido usar certificados X.509, mas somente se houver uma configuração completa de *Infraestrutura de Chave Pública* (PKI). Neste exemplo, usarei Dogtag [6] do projeto Fedora como solução PKI. Usuários com outras distribuições podem preferir OpenSC [7]. A biblioteca PAM é a mesma para ambas, `pam_pkcs11`.

O Dogtag consiste de vários componentes. Para esta configuração, você também vai precisar de uma *Autoridade Certificadora* (CA) para criar os certificados X.509. O *Certificado Online de Protocolo de Status* (OCSP – *Online Certificate Status Protocol*) é usado para validação dos certificados online nos cartões com chip. Para a validação offline, basta a última versão da *Lista de Certificados Revogados* (CRL – *Certificate Revocation List*) no sistema cliente. Claro, também é preciso encontrar uma maneira de mover o certificado do usuário da autoridade certificadora para o cartão. É possível usar o *Enterprise Security Client* (ESC) para abrir uma conexão para outro componente da PKI, o *Token Processing System*, no caso.

Assumindo uma autenticação correta, o certificado de usuário é, então, copiado para o cartão no processo de inscrição. A ferramenta ESC, em seguida, dá ao usuário uma abordagem prática para gerenciar o cartão. Se o usuário precisar solicitar um novo certificado à CA ou precisar de um novo PIN para a chave privada no cartão, isso não é um problema com ESC. Caso use OpenSC para gerenciar seu cartão, é possível transferir um arquivo PKCS#12 [8] para ele usando:

```
pkcs15-init
--store-private-key tscherf.p12
--format pkcs12
--auth-id 01
```

O arquivo PKCS#12 contém as chaves públicas e privadas. Caso possua um certificado de usuário de uma autoridade pública de certificação como a CACert [9], será possível usar o gerenciador do certificado do seu

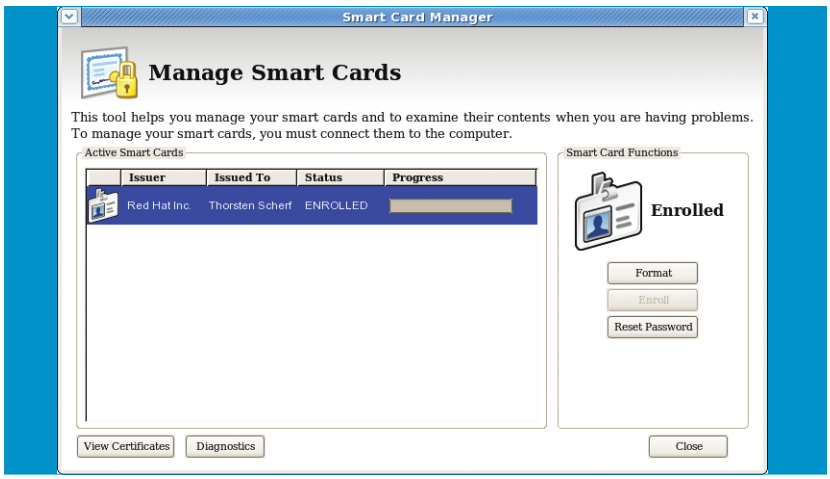

Figura 8 O Security Client oferece uma opção simples para gerenciar cartões com chip.

navegador para exportar o certificado para um arquivo e depois transferi-lo para o cartão como descrito.

Caso não possua um certificado, é possível fazer uma solicitação e enviá-la à autoridade de certificação apropriada. Quando a autoridade verificar sua solicitação, ela lhe entregará um certificado.

Nesses dois métodos, é possível usar o arquivo `/etc/pam_pkcs11/pam_pkcs11.conf` para definir o driver que irá acessar o cartão. O driver pode ser modificado no arquivo de configuração, como mostrado na **listagem 4**.

Aqui, o caminho correto precisa ser especificado para os repositórios de certificados locais CRL e CA. O banco de dados CRL é necessário para checar se o certificado no cartão do usuário ainda é válido e não foi revogado pela autoridade de certificação. Você precisa do certificado para a autoridade que emitiu a licença do usuário a partir do repositório de certificado CA. Isso possibilita a validação da autenticidade do usuário.

Certificados para Thunderbird e outros

Aplicativos que dependem de *Network Security Services* (NSS) para assinar ou criptografar e-mails com S/MIME, como o Thunderbird, usam um arquivo em `nss_dir` como banco de dados da CA; aplicativos baseados em bibliotecas OpenSSL usam o banco de dados no diretório `ca_dir`. A ferramenta `certutil` pode importar o certificado CA para o banco de dados NSS; certificados baseados em OpenSSL podem ser anexados ao arquivo existente. Por fim, é possível definir o mapeamento entre certificados de usuários e usuários Linux no arquivo de configuração `pam_pkcs11`. Várias ferramentas de mapeamento estão disponíveis para isso, especificadas da seguinte maneira: `use_mappers = cn, uid`.

Depois, ainda será necessário adicionar a biblioteca PAM `pam_pkcs11` ao arquivo de configuração PAM correto – isto é, `/etc/pam.d/login` ou `/etc/pam.d/gdm`. É possível editar o arquivo manualmente ou usar a ferramenta `system-config-authentication` mencionada antes.

Ao inserir o cartão no leitor e inicializar a ferramenta ESC, será possível ver o certificado (**figura 8**). Se tentar agora se conectar através de um console ou uma nova sessão GDM, o processo de autenticação deve ser completamente transparente. O programa de e-mail Thunderbird pode usar o cartão para assinar e criptografar e-mails; o Firefox pode usar o certificado para autenticação do cliente em um servidor web. A recompensa disso tudo, a configuração, oferece vários cenários de utilização. O Guia ESC [10] tem uma descrição mais detalhada da ferramenta e de sua configuração.

Conclusão

O PAM é um conjunto muito poderoso para lidar com autenticação. Como vimos nas bibliotecas PAM introduzidas nesse artigo, a funcionalidade não se restringe a autenticar usuários, mas também cobre tarefas como autorização, gerenciamento de senhas e gerenciamento de sessões. Os administradores que se familiarizarem com a configuração do PAM, que não é trivial, serão recompensados com uma riqueza de recursos e opções flexíveis para autenticação e autorização baseadas em senhas e em hardware. ■

Mais informações

[1] Linux PAM: http://www.kernel.org/pub/linux/libs/pam/

[2] PAM ThinkFinger: http://thinkfinger.sourceforge.net/

[3] pam_usb: http://downloads.sourceforge.net/pamusb/pam_usb-0. 4.2.tar.gz?download

[4] Site da Yubico: http://www.yubico.com/products/yubikey/

[5] Servidor SSH para Yubikey: http://code.google.com/p/yubico-pam/downloads/

[6] Dogtag PKI: http://pki-svn.fedora.redhat.com/wiki/PKI_Main_Page/

[7] OpenSC: http://www.opensc-project.org/

[8] Especificações PKCS: http://en.wikipedia.org/wiki/PKCS/

[9] Autoridade certificadora CACert: http://cacert.com/

[10] Guia ESC: http://directory.fedoraproject.org/wiki/ESC_Guide/

Gostou do artigo?

Queremos ouvir sua opinião. Fale conosco em cartas@linuxmagazine.com.br

Este artigo no nosso site: http://lnm.com.br/article/4261