Flexible Benutzerauthentifizierung mit PAM

Schlüsselfertig

Die Pluggable Authentication Modules (PAM) erlauben unter Linux nicht nur die Einbindung von Directory-Services wie LDAP, sondern auch die Integration von Hardware-basierter Authentifizierung. Thorsten Scherf

Zur Authentifizierung von Benutzerkonten kommt immer öfter neuartige Hardware zum Einsatz. Damit sich solcherlei Gerät für den Benutzer transparent ins System einbinden lässt, gibt es die so genannten Pluggable Authentication Modules (PAM). Hiermit steht dem versierten Admin nicht nur eine Vielzahl unterschiedlichster Methoden zur Authentifizierung seiner Benutzer zur Verfügung, ganz nebenbei erhält er auch recht umfangreiche Möglichkeiten, um den kompletten Ablauf einer Benutzersitzung zu steuern.

Alte Schule

Auf Linux-Systemen funktioniert die Benutzeranmeldung üblicherweise über die Dateien »/etc/passwd« und »/etc/shadow«. Meldet sich ein Benutzer mit seinem Namen und Passwort beispielsweise über »login« an einem System an, erzeugt dies eine kryptographische Prüfsumme des eingegebenen Benutzerpassworts und vergleicht das Ergebnis mit der gespeicherten Prüfsumme für diesen Benutzer in der Datei »/etc/shadow«. Stimmen beide überein, ist der Benutzer korrekt authentifiziert, anderenfalls schlägt die Anmeldung fehl.

Dieses Verfahren stößt natürlich schnell an seine Grenzen. So befinden sich beispielsweise in großen Netzwerkumgebungen die Benutzerdaten üblicherweise an zentraler Stelle, beispielsweise auf einem LDAP-Server. In diesem Fall muss das Login-Programm die Passwort-Prüfsumme also nicht aus der Datei »/etc/shadow«, sondern vom Verzeichnisdienst beziehen. Diese Aufgabe lässt sich durch den Einsatz der Pluggable Authentication Modules [1] recht einfach bewerkstelligen.

Modulare Authentifizierung

Ursprünglich Mitte der 90er Jahre von Sun Microsystems entwickelt, steht PAM heutzutage auf den meisten Unix-artigen Systemen zur Verfügung. PAM verlagert dabei den kompletten Ablauf des Authentifizierungsvorgangs von der Anwendung selbst auf ein zentrales Framework. Dieses Framework besteht aus einer umfangreichen Sammlung von Modulen. Jedes dieser Module hat dabei eine bestimmte Aufgabe. Die Anwendung erhält nur noch die Information zurückgeliefert, ob die Anmeldung des Benutzers erfolgreich

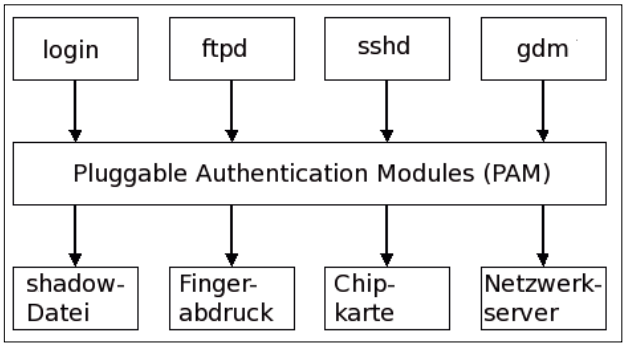


Abbildung 1: Mit PAM steht für die Anwendung ein zentrales Framework zum Benutzermanagement zur Verfügung.

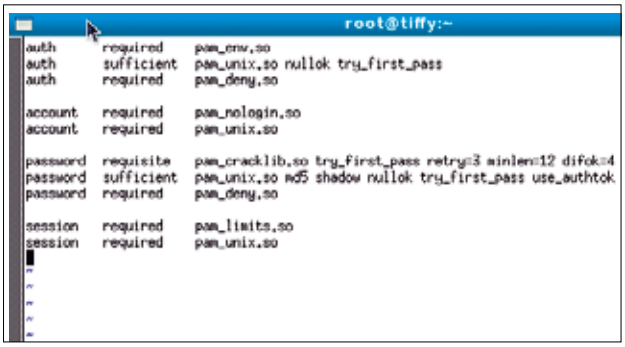


Abbildung 2: Eine klassische PAM-Konfigurationsdatei enthält Module und Bibliotheken, über die der Administrator PAM anpassen kann.

war oder nicht. Es ist nun also Aufgabe von PAM, sich darum zu kümmern, einen Benutzer über entsprechende Verfahren zu authentifizieren. Wie diese Verfahren aussehen, ist komplett im PAM-Framework hinterlegt, die Anwendung selbst besitzt hierüber überhaupt keine Informationen.

PAM ist in der Lage, auf verschiedenste Verfahren zur Authentifizierung zurückzugreifen. Neben den bekannten Netzwerk-basierten Methoden wie LDAP, NIS oder Winbind kann PAM mit Hilfe aktueller Bibliotheken auch auf diverse Hardwaregeräte zurückgreifen. So klappt die Anmeldung beispielsweise auch über eine Smartcard oder einen digitalen Fingerabdruck des Benutzers. Auch Einmalpasswörter wie S/Key oder Secure ID sind für PAM keine Fremdwörter. Es gibt sogar Verfahren, mit denen die Anmeldung nur dann klappt, wenn sich ein bestimmtes Bluetooth-Device in der Nähe des Rechners befindet.

Die Funktionsweise von PAM ist dabei recht simpel. Jede Anwendung, die mit PAM zusammenarbeitet – die Anwendung muss in diesem Fall gegen die Bibliothek »libpam« gelinkt sein –, verfügt im Ordner »/etc/pam.d/« über eine eigene Konfigurationsdatei. Sie heißt üblicherweise so wie die Anwendung selbst, also beispielsweise »login«.

In dieser Datei unterteilen Module die Aufgabenbereiche von PAM. Für jeden Aufgabenbereich steht eine Vielzahl von Bibliotheken zur Verfügung, die – dem Bereich entsprechend – die unterschiedlichsten Aufgaben haben (Abbildung 2). Über so genannte Kontroll-Flags lässt sich das Verhalten von PAM im Fehlerfall regeln, also beispielsweise dann, wenn ein Benutzer ein nicht korrektes Passwort eingegeben hat oder ein Fingerabdruck nicht verifiziert werden konnte.

Fingerabdruck

Aktuelle PAM-Bibliotheken bieten die Möglichkeit, Benutzer auch über Smartcards, USB-Tokens oder biometrische Merkmale zu authentifizieren. Aktuelle Notebooks besitzen oftmals einen Fingerabdruck-Leser, mit dessen Hilfe sich Benutzer über ihren digitalen Fingerabdruck am System anmelden können. Hierfür steht unter [2] eine PAM-Bibliothek mit dem Namen »thinkfinger« bereit. Laut Dokumentation arbeitet dieses Modul problemlos mit dem Fingerabdruck-Lesegerät UPEK/SGS Thomson Microelectronics zusammen. Dies finden Sie in den meisten aktuellen Lenovo-Notebooks und vielen externen Geräten.

Die meisten großen Linux-Distributionen bieten bereits fertige Pakete für diese PAM-Bibliothek an. Über die jeweiligen Paketmanager lässt sich die notwendige Software aus den Repositories installieren. Auf einem Fedora-System gelingt dies mittels »yum install thinkfinger«, auf einem Ubuntu Hardy bringt der Aufruf »apt-get install thinkfinger-tools libpam-thinkfinger« die notwendigen Pakete auf die Festplatte, und unter Gentoo reicht ein schlichtes »emerge sys-auth/thinkfinger«.

Open Suse benötigt die beiden Pakete »libthinkfinger« und »pam_thinkfinger«, die in einer nicht mehr ganz aktuellen Version in den Repositories liegen. Hier

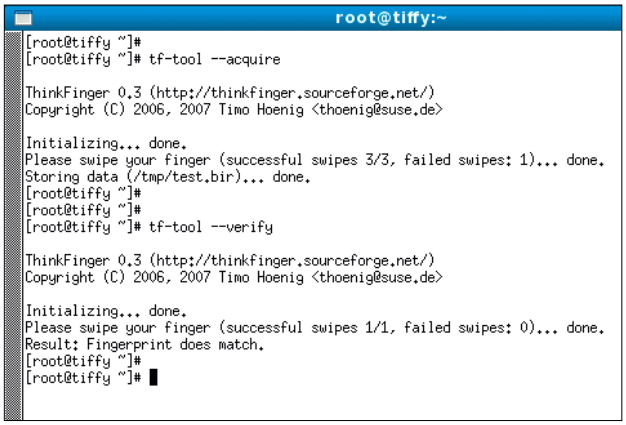


Abbildung 3: Über »tf-tool« kann der Administrator die Funktionsweise des Fingerabdruck-Lesegeräts testen ...

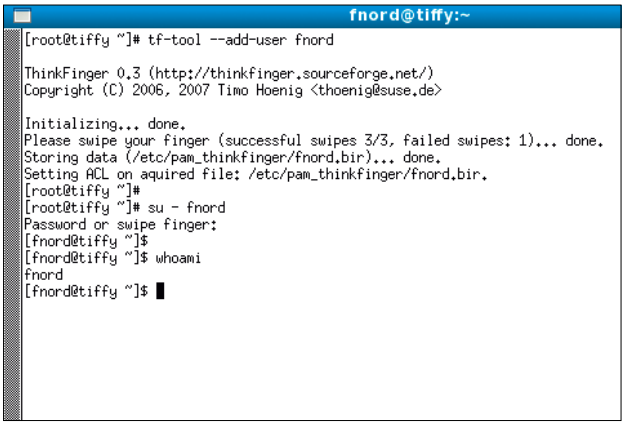


Abbildung 4: ... und anschließend für jeden Benutzer einen Fingerabdruck einlesen und abspeichern.

Listing 1: Konfigurationsdatei für »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>
```

hilft eventuell eine manuelle Installation mit dem klassischen Dreischritt (»./configure, make, make install«) aus einem aktuellen Sourcecode-Archiv heraus. Debian-Benutzer müssen auch in Lenny noch auf das Experimental-Repository zurückgreifen, dann klappt die Installation mittels »aptitude install libthinkfinger0 libpam-thinkfinger thinkfinger-tools«.

Hardware-Test

Bevor Sie die bestehende PAM-Konfiguration ändern, sollten Sie erst mal das Gerät selbst testen. Hierfür lesen Sie mit »tf-tool --acquire« den Abdruck eines Fingers. Im Anschluss können Sie diesen mit »tf-tool --verify« überprüfen (Abbildung 3). Stimmt der Fingerabdruck nicht überein, lautet das Ergebnis »Fingerprint does *not* match«. Die ersten Versuche können durchaus etwas holprig sein, da man sich erst einmal an die Funktionsweise des Geräts gewöhnen muss.

Ziehen Sie Ihren Finger zu schnell oder zu langsam über das Lesefeld, erkennt das Lesegerät den Abdruck oftmals nicht richtig und steigt mit einer Fehlermeldung aus. Funktioniert das Einlesen des Fingerabdrucks ohne Probleme, löschen Sie die temporäre Datei mit dem Testabdruck unterhalb von »/tmp« und erzeugen für jeden Benutzer eine individuelle Datei mit seinem Fingerabdruck auf dem System (Abbildung 4). Der Programmaufruf hierfür lautet »tf-tool --add-user username«. Der Benutzer muss dann dreimal seinen Finger über das Lesefeld ziehen. Wurde jeder Abdruck problemlos erkannt, speichert das Tool ihn in einer eigenen Datei unterhalb von »/etc/pam_thinkfinger/«.

Hat soweit alles funktioniert, machen Sie sich an die eigentliche PAM-Konfiguration. Abbildung 2 zeigt ein Beispiel für das Login-Programm, in dem als einziges Authentifizierungsmodul »pam_unix« aufgelistet ist. Möchten Sie sich zuerst über den Fingerabdruck-Leser authentifizieren, ist das hierfür notwendige PAM-Modul »pam_thinkfinger« vor »pam_unix« aufzurufen.

Damit PAM nach einem erfolgreichen Test des Fingerabdrucks nicht noch einmal nach dem Benutzerpasswort fragt, müssen Sie dahinter das Kontroll-Flag »sufficient« schreiben. Es bewirkt, dass PAM nach dem erfolgreichen Test keine weiteren Bibliotheken zur Authentifizierung mehr aufruft und »PAM_SUCESS« an das aufrufende Programm, hier »login«, zurückliefert. Sollte die Anmeldung über den Fingerabdruck nicht funktionieren, dann springt sozusagen als Notlösung »pam_unix« ein und fragt den Benutzer nach seinem regulären Passwort.

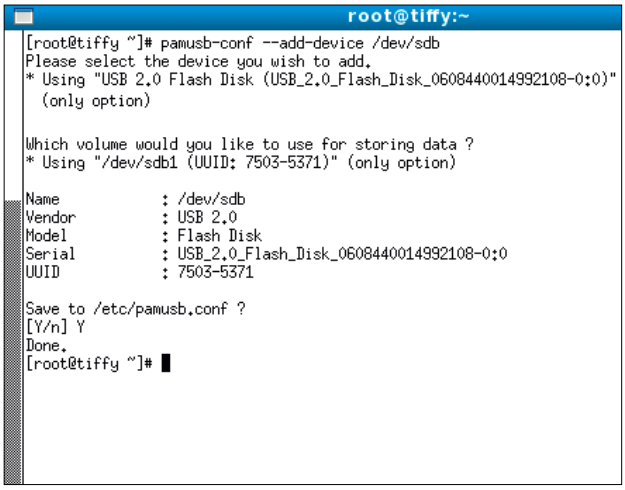
Es wäre allerdings recht umständlich, müssten Sie für alle Programme die gewünschten PAM-Bibliotheken in jeder einzelnen PAM-Konfigurationsdatei manuell eintragen. Als Lösung existiert hierfür eine Art zentrale PAM-Konfigurationsdatei. Unter Fedora oder Red Hat heißt sie »/etc/pam.d/system-auth«, einige andere Linux-Distributionen verwenden den Namen »/etc/pam/common-auth«. Hier trägt der Administrator alle gewünschten Bibliotheken zur Authentifizierung von Benutzern ein. Über das Kontroll-Flag »include« lässt sich diese Datei dann aus allen anderen PAM-Konfigurationsdateien einbinden. Somit stehen allen Programmen die PAM-Bibliotheken aus der zentralen Konfigurationsdatei zur Verfügung, ab sofort also auch »pam_thinkfinger«.

USB-Token

Eine weitere Form von Hardware-basierter Authentifizierung stellt die PAM-Bibliothek »pam_usb« zur Verfügung. Hier überprüft PAM, ob ein bestimmtes USB-Gerät am Rechner angeschlossen ist. Ist dies der Fall, gelingt die Anmeldung eines Benutzers, sonst bleibt der Zugang zum System verschlossen. Identifiziert wird das angeschlossene Gerät über seine eindeutige Seriennummer sowie den Modell- und Herstellernamen. Zusätzlich ist sowohl auf dem USB-Gerät selbst wie auch auf dem Rechner eine Zufallszahl hinterlegt, die sich nach jeder erfolgreichen Anmeldung ändert.

Bei der Benutzeranmeldung überprüft PAM neben den genannten Eigenschaften des USB-Geräts zusätzlich diese Zufallszahl. Stimmt die auf dem USB-Gerät gespeicherte Zahl nicht mit der Zahl auf dem System überein, scheitert die Anmeldung. Damit sind Sie davor geschützt, dass ein möglicher Angreifer die

◀ **Abbildung 5: Auf Fedora-Systemen steht mit »system-config-authentication« ein einfaches Tool zur PAM-Konfiguration zur Verfügung.**



▶ **Abbildung 6: Über bestimmte Eigenschaften des USB-Geräts lässt sich dieses später wieder identifizieren – nur dann klappt die Benutzeranmeldung.**

Zufallszahl Ihres USB-Geräts klaut, sie auf einem eigenen USB-Gerät hinterlegt und die Eigenschaften des eigenen Gerätes entsprechend modifiziert, um Zugang zum System zu bekommen. Da sich die Zufallszahl auf dem System bei jeder Anmeldung ändert, stimmt die gestohlene Zahl nun nicht mehr mit der auf dem System überein.

Gentoo und Debian Linux stellen für diese PAM-Bibliothek fertige Pakete zur Verfügung. Diese lassen sich über den jeweiligen Paketmanager installieren wie bereits für »pam_thinkfinger« beschrieben. Benutzer anderer Linux-Distributionen können sich unter [3] das aktuelle Sourcecode-Archiv herunterladen und mittels »make; make install« alle notwendigen Dateien übersetzen und auf dem lokalen System installieren.

Nachdem Sie dann ein beliebiges USB-Gerät angeschlossen haben – dabei kann es sich auch um ein Handy mit einer SD-Karte handeln –, werden dessen Eigenschaften in der Datei »/etc/pamusb.conf« festgehalten. Dies klappt mit dem Befehl »pamusb-conf --add-device *USB-Geräte-Name*« (Abbildung 6).

Benutzerspezifisch

Mit »pamusb-conf --add-user *Benutzer*« kann der Administrator nun jeden Benutzer zur Konfiguration hinzufügen und eine entsprechende Zufallszahl erzeugen. Diese wird für jeden Anwender sowohl auf dem USB-Gerät als auch auf dem System selbst hinterlegt.

Zusätzlich trägt das Tool jeden Benutzer in die XML-basierte Konfigurationsdatei »/etc/pamusb.conf« ein. Hierüber lassen sich später für jeden bestimmte Aktionen definieren, die immer dann ablaufen, wenn das USB-Gerät vom System entfernt oder wieder angesteckt wird. Beispielsweise sorgt der Eintrag aus Listing 1 in der Konfigurationsdatei dafür, dass automatisch der Bildschirm gesperrt wird, wenn das USB-Gerät nicht mehr am System angeschlossen ist

Listing 2: Authentifizierung mittels USB-Gerät

```
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
```

Abschließend ist die PAM-Bibliothek »pam_usb« noch in die entsprechende PAM-Konfigurationsdatei »/etc/pam.d/system-auth« oder »/etc/pam.d/common-auth« einzutragen. Verwenden Sie hierbei als Kontroll-Flag »sufficient«, klappt die Anmeldung, solange das entsprechende USB-Gerät am System vorhanden ist und die Zufallszahl für den

Benutzer auf beiden Geräten übereinstimmt (Listing 2).

Wer den Sicherheitsstandard etwas erhöhen möchte, kann das Kontroll-Flag »sufficient« durch »required« ersetzen. Nun wird zuerst nach dem passenden USB-Gerät gesucht, aber selbst wenn es korrekt erkannt wird, erhält der Benutzer im nächsten Schritt noch die Aufforderung, sein Benutzerkennwort einzugeben. Erst wenn beide Tests erfolgreich sind, klappt die Anmeldung.

Alle hier vorgestellten Verfahren zur Hardware-basierten Anmeldung sind zwar schnell eingerichtet, besitzen aber auch ihre Schwachstellen. Wie leicht sich beispielsweise Fingerabdrücke fälschen lassen, zeigt ein Video vom Chaos Computer Club [4]. Im Falle des USB-Sticks ist ebenfalls daran zu denken, dass es mit der Sicherheit schnell vorbei ist, sollte das Gerät geklaut werden.

Wer es mit der Sicherheit wirklich ernst meint, sollte auf bekannte Zwei-Faktoren-Authentifizierungsverfahren zurückgreifen. Hier kommen Sie aber um den Einsatz von Chipkarten mit entsprechenden Lesegeräten oder USB-Token mit Einmalpasswörtern und PIN nicht herum.

Yubikey

Eine kleine Firma names Yubico [5] aus Schweden vertreibt seit Kurzem die so genannten Yubikeys (Abbildung 7). Dabei handelt es sich um sehr kleine USB-Token, die eine reguläre USB-Tastatur simulieren. Der Key besitzt einen Button auf der Oberseite; drücken Sie diesen, sendet das Token ein Einmalpasswort (One Time Password, OTP) an die gerade aktive Anwendung, also etwa den Login-Prompt eines SSH-Servers oder die Anmeldemaske eines Webdienstes. Der Abgleich dieses OTP findet in Echtzeit über einen Authentifizierungsserver der Firma Yubico statt. Da die Software unter einer Open-Source-Lizenz steht, lässt sich theoretisch auch ein eigener Authentifizierungsserver im LAN aufbauen. Damit entfällt die Notwendigkeit einer ständigen Internetverbindung.

Abbildung 7: **Der Yubikey für One-Time-Passwörter übergibt dank USB-Keyboard-Emulation ohne spezielle Treiber den Pasdwort-Hash an Anwendungen.**

Die Funktionsweise des Token ist recht einfach. Anders als etwa die bekannten RSA-Tokens benötigt ein Yubikey keine Batterie, da die Generierung des OTP nicht zeitabhängig stattfindet, sondern eine gewisse Anzahl von Einmalpasswörtern direkt im Vorfeld festgelegt wird. Diese sind auf dem Token und in einer Datenbank auf dem Authentifizierungsserver gespeichert. Beim Drücken auf den Yubikey-Button sendet der Key eines dieser Einmalpasswörter an die aktive Anwendung. Diese greift dann über ein API auf den Server zurück, um das Passwort zu verifizieren.

Gelingt dies nicht (Unknown Key) oder wurde das Passwort bereits verwendet (Replayed Key), gibt es eine entsprechende Fehlermeldung und die Benutzeranmeldung scheitert. Erklärt der Server den Key als gültig, dann setzt dieser den »usage-count« auf »1« und der Benutzer gilt als authentifiziert. Eine nochmalige Anmeldung mit diesem Key ist dann nicht mehr möglich.

Dank steigender Verbreitung des Yubikey und eines einfachen API, bieten immer mehr Anwendungen die Authentifizierung über den Yubico-Server an. Beispielsweise gibt es ein Plugin für den beliebten Wordpress-Blog, sodass Benutzer eines Yubikey ihn zur Anmeldung am Blog verwenden können.

Ein Projekt aus dem Summer of Code der Firma Google hat letztes Jahr unter anderem ein PAM-Modul für eine Anmeldung an einem SSH-Server herausgebracht. Statt sein Benutzerpasswort am Login-Prompt einzutippen, drücken Sie hier einfach auf den Button des Yubikey, damit er einen 44 Zeichen langen, mit Modhex kodierten Passwort-String an den SSH-Server sendet. Der wiederum verifiziert den String über eine Abfrage des Yubico-Servers. Über die ersten zwölf Zeichen lässt sich der Benutzer auf dem Yubikey-Server eindeutig identifizieren, die anderen 32 Zeichen stellen dann das Einmalpasswort dar.

Listing 3: PAM-Konfiguration für einen Yubikey

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

PAM hilft Yubi

Auf dem SSH-Server können Sie über eine zentrale Datei festlegen, wer sich mit Hilfe eines Yubikey auf dem Server anmelden darf. Hierzu erzeugen Sie eine Datei »/etc/yubikey-users.txt«, sie enthält pro Zeile einen Benutzernamen und – mit Doppelpunkt getrennt – die zugehörige Yubikey-ID, also die ersten zwölf Zeichen eines jeden OTP für diesen Benutzer. Alternativ kann sich jeder Benutzer in seinem Homeverzeichnis eine Datei mit

Abbildung 8: Über den Security-Client lässt sich die Chipkarte sehr einfach verwalten. Hier zeigt es die vorhandenen Zertifikate an.

diesen Informationen anlegen, zum Beispiel »~/.yubico/authorized_yubikeys«. Damit ein Abgleich des OTP mit dem Yubico-Server stattfindet, ist die entsprechende PAM-Konfiguration notwendig. Hierfür erweitern Sie einfach die Datei »/etc/pam.d/sshd« um eine Zeile für den Yubikey (Listing 3).

Bei einer Konfiguration wie in diesem Listing erfolgt die Authentifizierung zusätzlich zum regulären, über die Datei »system-auth« eingebundenen Authentifizierungsverfahren. Ersetzen Sie das Kontroll-Flag »required« nun wieder durch »sufficient«, erfolgt eine Anmeldung des Benutzers bereits nach erfolgreicher Validierung des Yubikey-OTP.

Da dieses jedoch nicht mit einer zusätzlichen PIN geschützt ist, besteht wieder eine Gefahr beim Diebstahl des Token. Nicht autorisierte Benutzer könnten allein durch den Besitz des Token eine fremde Benutzeridentität spoofen. Eine Erweiterung der OTP durch eine PIN steht schon auf der Roadmap der Entwickler, entsprechende inoffizielle Patches sind bereits verfügbar.

X.509-Zertifikate und PAM

Die klassische Zwei-Faktoren-Authentifizierung findet meist immer noch mit Hilfe von Chipkarten statt. Die enthalten dann üblicherweise ein mit einer PIN geschütztes Zertifikat. Mit der PAM-Bibliothek »pam_pkcs11« können Benutzer sich dann über ein solches X.509-Zertifikaten am System anmelden.

Das Zertifikat enthält einen privaten und einen öffentlichen Schlüssel. Beide lassen sich auf einer geeigneten Chipkarte speichern, wobei der private Schlüssel durch eine zusätzliche PIN gesichert ist. Somit sind Sie davor geschützt, dass andere Benutzer sich unter falscher Identität anmelden können, selbst wenn diese im Besitz einer fremden Chipkarte sind. Zur Anmeldung ist neben der Chipkarte selbst nämlich auch noch die PIN der Karte erforderlich. Ist diese nicht bekannt, schlägt das Login fehl.

Der Anmeldevorgang läuft im Detail wie folgt ab: Der Benutzer schiebt seine Chipkarte in das Lesegerät ein und gibt die passende PIN ein, das Zertifikat mit dem öffentlichen und dem privaten Schlüssel wird auf der Karte gesucht. Ist das Zertifikat gültig, findet ein Mapping für den entsprechenden Benutzer auf dem System statt. Für dieses Mapping lassen sich verschiedene Informationen aus dem Zertifikat heranziehen, üblicherweise verwendet man hierfür den »Common Name« oder die »uid«, die im Zertifikat gespeichert sind.

Um zu verifizieren, dass es sich wirklich um den richtigen Benutzer handelt, generiert das System eine zufällige, 128 Bit lange Zahl. Eine spezielle Funktion der Chipkarte verschlüsselt diese dann mit Hilfe des privaten Schlüssels, der sich ja ebenfalls auf der Karte befindet. Um Zugriff auf den privaten Schlüssel zu erhalten, ist im Vorfeld die korrekte PIN einzugeben. Danach nutzt das System den frei verfügbaren öffentlichen Schlüssel, um das soeben erzeugte Chiffrat wieder zu entschlüsseln. Stimmt das Ergebnis mit der erzeugten Zufallszahl überein, ist der Benutzer korrekt authentifiziert, da beide Schlüssel zusammenpassen.

Als Hardware benötigen Sie eine Chipkarte mit entsprechendem Lesegerät. Hier bieten sich beispielsweise die Gemalto E-Gate- oder die SCR-USB-Geräte von SCM an. Als Token können Sie getrost auf alle zu Java Card 2.1.1 oder Global Platform 2.0.1 kompatiblen Karten zurückgreifen. Bekannte Token hierfür sind die Karten aus der Gemalto-Cyberflex-Reihe. Auf Software-Seite stehen verschiedene Lösungen zur Verfügung. Die hier beschriebene Variante setzt die Pakete »pcsc-lite« und »pcsc-lite-libs« für den Zugriff auf das Lesegerät voraus.

Public Key Infrastructure

Sinnvoll ist der Einsatz von X.509-Zertifikaten allerdings nur dann, wenn eine komplette Public-Key-Infrastruktur (PKI) zur Verfügung steht. In diesem Beispiel kommt als PKI-Lösung Dogtag [6] aus dem Fedora-Projekt zum Einsatz. Ein kleines Tutorial zu Dogtag findet sich unter [7]. Für die Benutzer anderer Distributionen bietet sich die Open-SC-Software [8] an. Die eingesetzte PAM-Bibliothek ist bei beiden Varianten jeweils die »pam_pkcs11«.

Dogtag besteht aus verschiedenen Komponenten. Zum Erzeugen der eigentlichen X.509-Zertifikate kommt die Certificate Authority (CA) zum Einsatz. Zum Verifizieren eines Zertifikats auf einer Chipkarte lässt sich für Online-Validierungen der OCSP-Dienst (Online Certificate Status Protocol) einsetzen. Für

Listing 4: Konfigurationsdatei für »pam_pkcs11«

```
01 pkcs11_module coolkey {
02     module = libcoolkeypk11.so;
03     description = "Cool Key"
04     slot_num = 0;
05     ca_dir = /etc/pam_pkcs11/cacerts;
06     nss_dir = /etc/pki/nssdb;
07     crl_dir = /etc/pam_pkcs11/crls;
08     crl_policy = auto;
09 }
```

Offline-Validierungen reicht es aus, die jeweils aktuelle Version der Certificate Revocation List (CRL) auf dem Clientsystem verfügbar zu haben.
Natürlich muss nun auch irgendwie das Benutzerzertifikat von der Certificate Authority auf die Chipkarte kommen. Hierfür baut der Enterprise Security Client (ESC) eine Verbindung zu einer weiteren PKI-Komponente auf, dem so genannten Token Processing System (TPS). Nach korrekter Authentifizierung wird dann das Benutzerzertifikat auf die Chipkarte übertragen (Enrollment). Über das ESC-Tool kann der Anwender die Karte anschließend auch bequem verwalten. Möchte er beispielsweise ein neues Zertifikat von der CA anfordern oder eine neue PIN für den privaten Schlüssel auf der Karte setzen – alles kein Problem.

Chipkarte und Zertifikat

Wenn Sie zum Verwalten Ihrer Chipkarte die Open-SC-Software verwenden, können Sie eine bereits vorhande PKCS#12-Datei [9] mittels »pkcs15-init --store-private-key tscherf.p12 --format pkcs12 --auth-id 01« auf die Karte übertragen. Die PKCS#12-Datei enthält dabei sowohl den öffentlichen als auch den privaten Schlüssel.
Besitzen Sie ein Benutzerzertifikat von einer öffentlichen Zertifizierungsstelle, zum Beispiel von der CACert [10], haben Sie die Möglichkeit, es über die Zertifikatsverwaltung Ihres Browsers in eine Datei zu exportieren und diese dann wie beschrieben auf die Chipkarte zu bringen. Ist noch kein Zertifikat vorhanden, können Sie eine entsprechende Anfrage erzeugen und diese dann an die zuständige Zertifizierungsstelle senden. Hat diese die Authentizität der Anfrage verifiziert, gibt sie ein entsprechendes Zertifikat zurück.
Bei beiden Varianten lässt sich über die Datei »/etc/pam_pkcs11/pam_pkcs11.conf« festlegen, mit welchem Treiber auf die Chipkarte zugegriffen werden soll. Der Treiber lässt sich in der Konfigurationsdatei dann wie in Listing 4 anpassen. Wichtig ist hierbei, die passenden Pfade zur lokalen CRL- und CA-Zertifikatsdatenbank richtig anzugeben. Die CRL-Datenbank ist nötig, um sicherzustellen, dass das Zertifikat auf der Chipkarte eines Benutzers immer noch gültig ist und nicht bereits von der Zertifizierungsstelle zurückgerufen wurde. Aus der CA-Zertifikatsdatenbank ist das Zertifikat der Zertifizierungsstelle erforderlich, die die Benutzerzertifikate ausgestellt hat. Hiermit lässt sich die Authentizität der Benutzer verifizieren.

Zertifikate für Thunderbird & Co.

Anwendungen, die auf die Network Security Services (NSS) zurückgreifen, also etwa Thunderbird zum Signieren oder Verschlüsseln von E-Mails mittels S/MIME, verwenden als CA-Zertifikatsdatenbank die Datei im Verzeichnis »nss_dir«. Anwendungen die auf den Open-SSL-Bibliotheken basieren, verwenden die Datenbank im Verzeichnis »ca_dir«. Das Tool »certutil« kann das CA-Zertifkat in die NSS-Datenbank importieren, Open-SSL-basierte Zertifikate sind einfach an die bestehende Datei anzuhängen. Schließlich lässt sich in der »pam_pkcs11«-Konfigurationsdatei noch festlegen, wie die Zuordnung zwischen einem Benutzerzertifikat und dem zugehörigen Linux-Benutzer vorzunehmen ist. Hierfür stehen diverse Mapper zur Verfügung, aus denen Sie mittels »use_mappers = cn, uid« wählen.
Abschließend ist noch die PAM-Bibliothek »pam_pkcs11« in die entsprechende PAM-Konfigurationsdatei einzutragen, also etwa in die Datei »/etc/pam.d/login« oder »/etc/pam.d/gdm«. Diese passen Sie entweder manuell oder mit dem bereits erwähnten Tool »system-config-authentication« an.
Wenn Sie nun die Chipkarte in das Lesegerät stecken und das ESC-Tool starten, sollten Sie das Zertifikat der Chipkarte sehen (Abbildung 8). Versuchen Sie sich über die Konsole oder eine neue GDM-Sitzung am System anzumelden, sollte der Authentifizierungsvorgang, wie oben beschrieben, komplett transparent ablaufen. Das E-Mail-Programm Thunderbird kann die so beschriebene Karte auch zum Signieren und Verschlüsseln von E-Mails verwenden, Firefox das Zertifikat für eine Client-seitige Authentifizierung auf einem Webserver verwenden.
Der relativ große Konfigurationsaufwand entschädigt also mit einem umfangreichen Angebot verschiedenster Einsatzzwecke. Eine recht ausführliche Beschreibung zum ESC-Tool und dessen Konfiguration liefert [11].

Fazit

Abschließend lässt sich festhalten, dass mit PAM für alle Linux-Administratoren ein sehr leistungsstarkes Framework für die Authentifizierung verfügbar ist. Wie die Beschreibung einer PAM-Bibliotheken in diesem Artikel zeigt, beschränkt sich der Funktionsumfang dabei nicht nur auf die Authentifizierung von Benutzern, sondern umfasst auch Bereiche wie Autorisierung sowie Passwort- und Session-Management.
Jeder Administrator, der sich näher mit der nicht immer ganz einfachen PAM-Konfiguration beschäftigt, wird durch die reichhaltigen und flexiblen Möglichkeiten für die Authentifizierung und Autorisierung über Passwörter und Hardware belohnt. *(ofr)* ■

Infos

[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] Video über das Nachbilden eines Fingerabdrucks: [http://chaosradio.ccc.de/ctv001.html]
[5] Yubico-Webseite: [http://www.yubico.com/products/yubikey/]
[6] Dogtag PKI: [http://pki-svn.fedora.redhat.com/wiki/PKI_Main_Page]
[7] Thorsten Scherf, „Public-Key-Infrastruktur mit Dogtag“: ADMIN 01/2009
[8] Open SC: [http://www.opensc-project.org]
[9] PKCS-Spezifikationen: [http://de.wikipedia.org/wiki/PKCS]
[10] CACert-Zertifizierungsstelle: [http://cacert.com]
[11] ESC-Guide: [http://directory.fedoraproject.org/wiki/ESC_Guide]

Der Autor

Thorsten Scherf arbeitet als Senior Consultant für Red Hat EMEA. Er ist oft als Vortragender auf Konferenzen anzutreffen. Wenn ihm noch Zeit bleibt, nimmt er gerne an Marathonläufen teil.