

Unix-Encrypt Password System, shadow, Pluggable Authentication Module

Thomas Glanzmann sithgla@stud.uni-erlangen.de

2. Mai 2002

Ausarbeitung zum Vortrag über Unix-Passwörter, shadow und PAM

Bei Mehrbenutzersystemen ist es üblich die verschiedenen Benutzerkonten durch Passwörter zu sichern. Dies gewährleistet die Privatsphäre der einzelnen Benutzer. Unter Unix Systemen werden die Passwörter in der Datei `/etc/passwd` aufbewahrt. Diese Passwörter werden dort nicht im Klartext gespeichert, da die Datei für alle Benutzer des Systems lesbar ist. In den folgenden Sektionen werde ich auf diverse Mechanismen eingehen, welche die Authentifizierung eines Benutzers am System ermöglichen.

1 UNIX Authentifizierungskonzepte

1.1 Historie und Grundlagen

Das Passwort eines Unix Benutzers wird mit Hilfe einer Einweg - Funktion in eine Prüfsumme umgewandelt und in der Datei `/etc/passwd` neben anderen Benutzerdaten aufbewahrt. Möchte sich nun ein Benutzer authentifizieren wird das eingegebene Passwort wieder in eine Prüfsumme kodiert und mit der bereits gespeicherten verglichen. Stimmen die beiden überein ist der Benutzer erfolgreich authentifiziert.

Die wohl verbreitetste Prüfsummenfunktion basiert auf dem DES Algorithmus¹ und wird `crypt()` Algorithmus genannt. Sie wurde in den 70er Jahren für die PDP 11 entwickelt [Garfinkel].

1.2 Der DES Algorithmus als Grundlage für den `crypt()` Algorithmus

Der `crypt()` Algorithmus basiert wie schon erwähnt auf dem DES Algorithmus, welcher vom NIST² entwickelt wurde [Garfinkel]. Der DES Algorithmus benutzt einen 56 bit Schlüssel, welcher acht 7 bit ASCII Zeichen entspricht, um 64 bit Blöcke Klartext in 64 bit Blöcke Chiffretext zu kodieren.

Um den DES Algorithmus als Prüfsummenfunktion verwenden zu könnten wird ein 64 bit Block von Nullen mit dem Passwort verschlüsselt. Dieser Vorgang wird mit dem Chiffretext der vorangegangenen Runde als Klartext 25 mal wiederholt [Feldmeier]. Diese Wiederholung dient um Angriffe³ zu erschweren: Es können weniger Passwörter pro Sekunde verschlüsselt werden. Und somit braucht man auch mehr Zeit um eine Reihe von Passwörtern zu testen. Als der `crypt()` Algorithmus für die PDP 11 entwickelt wurde dauerte eine einzelne Prüfsummenbildung auf diesem Rechner eine Sekunde.

¹Data Encryption Standard

²National Institute of Technology

³vgl. Abschnitt 1.6 Attacken

1.3 Modifikationen am DES Algorithmus für den crypt() Algorithmus: Der Salt

Um Bruteforce und Tabellenangriffe⁴ zu erschweren wurde der DES Algorithmus modifiziert: Es wurde eine 12 bit Nummer (0-4096) eingeführt, welche den Chiffretext minimal variiert. Der Grund in dieser Modifikation liegt darin, dass man nun ein Klartextpasswort auf 4096 verschiedene Art und Weisen als Chiffretext darstellen kann. So wird es zum Beispiel möglich das gleiche Passwort auf verschiedenen System zu verwenden ohne das ein vermeintlicher Angreifer dies feststellen könnte. In der Praxis wird dieses System allerdings schlecht umgesetzt da der Salt nach der Tageszeit gewählt wird. Dadurch häufen sich einige Salts und andere werden überhaupt nicht verwendet.

1.4 Der crypt() Algorithmus in der Praxis

Beim Setzen des Passwortes wird dieses mit Hilfe eines Salt und dem crypt() Algorithmus in eine Prüfsumme kodiert. Der Salt wird mit der Prüfsumme als 2 (Salt) + 11 (Prüfsumme) druckbare Zeichen⁵ abgespeichert. Jedes dieser druckbaren Zeichen trägt 6 bits. Will sich der Benutzer nun authentifizieren wird das eingegebene Passwort mit Hilfe des aus der Passwort Datei entnommenen Salt kodiert und mit der bereits vorhandenen Prüfsumme verglichen. Stimmen die beiden Zeichenketten überein ist der Benutzer erfolgreich authentifiziert.

1.5 Schwachstellen des crypt() Algorithmus

Im Laufe der Jahre sind Rechen und Festplattenkapazität enorm gestiegen. Dadurch kann man mit einem handelsüblichen PC bis zu 350000 Passwörter in der Sekunde kodieren. Somit ist ein 7 Zeichen Passwort innerhalb einer Woche spätestens kompromittiert. Außerdem ist die Salt Länge auf 12 bit beschränkt [Garfinkel] und die maximale Passwortlänge beschränkt sich auf 8 Zeichen.

1.6 Angriffe

Es bieten sich hier zwei Arten von Angriffe an: Bruteforce Angriffe, welchen den gestammten Schlüsselraum systematisch und gegebenenfalls auch verteilt durchtesten, und Wörterbuchangriffe die mit Hilfe von Wörterbüchern oft verwendete Passwörter durchtesten. Außerdem gibt es Spezialrechner, welche den gesamten Schlüsselraum in 4 Stunden bewältigen [Garfinkel].

1.7 Der nächste Entwicklungsschritt: Die Shadow Passwort Datei /etc/shadow

Die Idee hinter diesem Prinzip ist es die Passwörter aus der Passwortdatenbank /etc/passwd in die Shadow Passwort Datei zu verschieben. Auf diese zweite Datei kann nur noch ein privilegierter Benutzer zugreifen. Diese Entwicklung wurde dadurch motiviert, dass die Mehrzahl der Benutzer schwache Passwörter wählen [Morris].

⁴Unter einem Tabellenangriffe versteht man eine generierte Tabelle in der sich das verschlüsselte Passwort im Klartext ablesen lässt.

⁵!, ', /, 0-9, a-z, A-Z

1.8 Eine Alternative zum crypt() Algorithmus: der md5 Algorithmus

Der md5 Algorithmus beseitigt einige Schwachstellen des crypt() Algorithmus: Die Länge des Salts steigt von 12 auf 48 bits und die Passwortlängenbegrenzung wird aufgehoben. Außerdem ist der md5 Algorithmus wesentlich langsamer als der crypt() Algorithmus. Dadurch werden Angriffe⁶ langsamer, da die Prüfsummenbildung eines einzelnen Passwortes länger dauert.

1.9 Weitere spezielle Alternativen

Es sind noch weitere Alternativen zum crypt() Algorithmus im Einsatz. Allerdings sind diese zu anderen System inkompatibel. Dazu gehören der Blowfish Algorithmus (OpenBSD) und ein erweiterter crypt() Algorithmus, welcher unter BSDI bzw. HP UX zum Einsatz kommt [Garfinkel].

2 Pluggable Authentication Module [Hernberg]

2.1 Motivation: Wofür benötigt man PAM ?

Mit Hilfe von PAM kann vermieden werden, dass ein Programm spezielle Privilegien braucht um auf die shadow Passwort Dateien zuzugreifen, welches wiederum die Sicherheit erhöht, da nicht alle passwortverifizierenden Programme privilegiert sein müssen. Möchte man zum Beispiel die Authentifizierungsstrategie ändern muss man, sofern man PAM nicht verwendet, alle involvierten Applikationen neu übersetzten. Weiterhin ist es durch PAM möglich den Zugang zum System und auch die Rechte der einzelnen Benutzer und Programme zu regulieren. Ein Passwort muss nicht mehr zwingend aus der Passwort Datei bezogen werden, es kann ebenso gut von einem Server auf der anderen Seite der Welt stammen.

2.2 Exkurs: Kereberos und Single Sign On

PAM kann auch im Zusammenspiel mit Kereberos verwendet werden. Dadurch wird ein Single Sign On möglich. Der Benutzer kann sich unter Verwendung eines einzelnen Passwortes an unterschiedlichen Systemen authentifizieren. Auf dieses Themengebiet wird im darauf folgenden Vortrag intensiv eingegangen.

2.3 Wie funktioniert PAM?

Durch Bibliotheken wird eine weitere Abstraktionsschicht eingeführt. Programme, welche Benutzer authentifizieren, lassen sich mit diesen Bibliotheken übersetzen. Daraufhin kann man sie zentral über die Konfigurationsdateien administrieren.

2.4 Ein Anwendungsbeispiel für PAM

Man möchte nur bestimmten Benutzern, welche der Gruppe root angehören, erlauben einen privilegierten Status einzunehmen. Dies ist mit PAM durch Einsetzen der folgenden Zeile `auth required pam_wheel.so group=root` in der Datei `/etc/pam.d/su` möglich.

⁶vgl. Abschnitt 1.6 Attacken

2.5 PAM Konfigurationsdateien

Die PAM Konfigurationsdateien liegen im Verzeichnis `/etc/pam.d/`. Für jedes Programm ist dort jeweils eine Konfigurationsdatei zu finden. Diese Konfigurationsdateien enthalten pro Zeile eine Anweisung bzw. einen Kommentar oder eine Leerzeile. Eine Anweisung besteht aus drei oder mehr Argumenten.

Das erste Argument vom Typ `type` zeigt PAM welche Art von Authentifikation für ein Modul vollzogen werden soll. Es können mehrere Module der gleichen Art benutzt werden. Der Benutzer muss dann alle Anforderungen der verschiedenen Module erfüllen. PAM kennt vier verschiedene Typen vom Typ `type`:

- `account` → Spezifiziert ob sich ein Benutzer einloggen darf.
- `auth` → Überprüft ob ein Benutzer auch derjenige ist für den er sich ausgibt.⁷
- `password` → Ermöglicht dem Benutzer über diese Applikation sein Passwort zu ändern.
- `session` → Setzt dem Benutzer nach der Authentifizierung seine Umgebung.⁸

Das zweite Argument vom Typ `control` zeigt an welche Aktion unternommen werden soll, wenn die Authentifizierung fehlschlägt. Für dieses zweite Argument gibt es vier Möglichkeiten:

- `requisite` → Ein Authentifizierungsfehler in diesem Modul führt zu einer direkten Abbruch der kompletten Authentifizierung.
- `required` → Ein Authentifizierungsfehler in diesem Modul führt zu einer Abbruch der Authentifizierung aber Module mit dem gleichen Typ können noch abgearbeitet werden.
- `sufficient` → Wenn die Authentifizierung mit diesem Modul erfolgreich ist, ist es die Authentifizierung auch, selbst wenn ein vorheriges Modul nicht erfolgreich war.
- `optional` → Dieses Modul ist nur signifikant, wenn es das einzige Modul für diesen Dienst ist.

Das dritte Argument zeigt PAM welches Modul es verwenden soll und optional wo dieses zu finden ist. Die meisten Konfigurationsanweisungen enthalten nur den Namen. In diesem Fall schaut PAM in dem vorgegebenen Konfigurationsverzeichnis⁹ nach. Ansonsten wird im vollqualifizierten Pfad nach dem Modul gesucht.

Das vierte und alle weiteren Argumente sind Optionen für das jeweilige individuelle Modul. Wenn man beispielsweise `pam_unix.so` die Option `nulok` übereicht, werden leere Passwörter akzeptiert.

3 Literaturverzeichnis

Literatur

[Garfinkel] Author Simson Garfinkel. Practical Unix & Internet Security. Spafford, O'Reilly, Second Edition, April 1996, ISBN 1-56592-148-8

⁷Überprüft das Passwort, die Retina oder eine Smartcard

⁸Es wird also zum Beispiel das home Verzeichnis gemountet oder ein Dienst gestartet.

⁹`/usr/lib/security` bzw. nach Linux Filesystem Standard unter `/lib/security`

- [Hernberg] Author Peter Hernberg. User Authentication HOWTO Explains how user and group information is stored and how users are authenticated on a Linux system (PAM), and how to secure you system's user authentication. 2000/05/02
- [Feldmeier] Author David C. Feldmeier and Philip R. Karn. Unix Password Security - Ten Years Later. Bellcore, 1989
- [Morris] Author Robert Morris and Ken Thompson, Password Security: A Case History, 1979