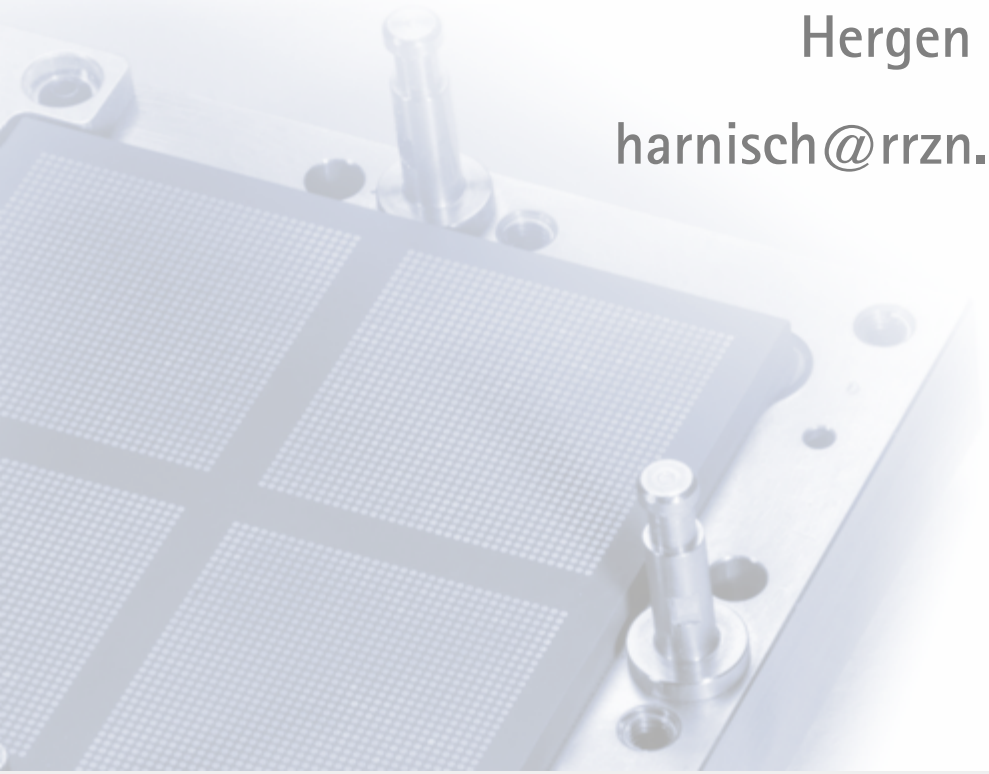
Passwörter



Hergen Harnisch

harnisch@rrzn.uni-hannover.de

Einleitung

Auf dem 26C3 wurde eine Flirtplattform der Rechten Szene gehackt, Passwort-Hitliste der Nutzer:

1. 123456 (13×)
2. landser (10×)
3. landser88 (8×)
4. siegheil (8×)
5. 14881488 (6×)
6. 888888 (6×)
7. skinhead (5×)
8. deutsch (5×)
9. 123456789 (5×)
10. siegheil88 (4×)

... leider wählen nicht nur Rechte schlechte Passwörter

Passwort als Authentifizierungsmethode

Jeder weiß, dass Passwörter eine schlechte Authentifizierungsmethode sind:

- komplexe Passwörter kann sich keiner merken
- Passwörter sind leicht kopier- & entwendbar

Bessere Methoden sind TAN-Listen, Out-of-Band, (PKI-) Chipkarten, OTP-Tokens:

- deutlich teurer
- schwieriger im Umgang
- bei sich zu tragen

... und werden die Passwörter noch lange nicht ablösen.

Einleitung

1. Grundlagen:
   - Komplexität
   - Hashes
   - Challenge-Response
2. Gefährdung & Maßnahmen:
   - Server
   - Transport
   - Client
3. Policy & Organisatorisches
4. Funktions-Accounts

R|R|Z|N

Erratbarkeit

Die Schwierigkeit, ein Passwort zu erraten, ergibt sich aus 2 Faktoren:

- Menge der möglichen Passwörter:
  - Länge
  - Zeichenvorrat (Groß-/Kleinschreibung, Zahlen, Sonderzeichen?)

8 Kleinbuchstaben $26^8 = 21 \cdot 10^{10}$	6 bel. Buchstaben oder Ziffern $62^6 = 5,7 \cdot 10^{10}$
10 Kleinbuchstaben $26^{10} = 14.117 \cdot 10^{10}$	8 bel. Buchstaben oder Ziffern $62^8 = 21.834 \cdot 10^{10}$

- Auswahl durch Menschen (zu einfach)
  - Wörter (aus dem Wörterbuch oder gängige Abkürzungen)
  - persönliche Angaben (z.B. Geburtsname der Mutter)

Stichwörter im Duden 135.000	openwall-Liste deutsch / komplett 118.000 / 3,9 Mio

$\longrightarrow$ Länge recht egal gegenüber menschlicher Schwäche, Komplexitätszwang!

Grundlagen: Hashes

Eine kryptologische Hash-Funktion ist eine nicht-umkehrbare Einwegfunktion, die eine Art „Quersumme“ oder „Fingerprint“ berechnet:

- aus Ergebnis ist Eingabe(-Passwort) nicht berechenbar, auch keine andere Eingabe mit gleichem Ergebnis
- Nebeneffekt: Ergebnis hat feste Länge und ist Standard-Ascii / Hex

Bei der Eingabe eines Passwortes durch einen sich authentisierenden Nutzer muss der authentifizierende Rechner das Passwort prüfen können. Dazu muss der Rechner

- entweder das Klartext-Passwort abgespeichert haben und vergleichen
- oder eine festgelegte Hash-Funktion auf das Passwort anwenden und das Ergebnis mit dem abgespeicherten Hash-Wert vergleichen.

Übliche Hash-Verfahren

veraltet, unsicher		noch okay		zu empfehlen	
crypt	altes Unix	NT/md4	Windows	sha-2	Unix, LDAP
LM	Windows-Kompatibiltät	md5	Unix, LDAP	sha-3	(Zukunft)
		sha-1	Unix, LDAP		

(sha-2 bezeichnet eine ganze Familie von Funktionen: sha-???, z.B. sha-256)

```
echo -n 'otto'|openssl md5
e5645dd85deb100fd1d71d0e8d671091

echo -n 'otto'|openssl sha -sha256
4991c47202960c5755d6886af261b0d8f6ccc929551d867a6f557a9353d4f1c5
```

Probleme

Input-Länge

Einige Hash-Fkt. beschränken die Eingabe im Zeichenumfang oder Länge:

- LM-Hash verwendet Großbuchstaben, max. 14 Zeichen
- Standard-Ascii, nur erste 8 Zeichen für crypt

Kollision bei md5

- derzeit nicht zu vorgegebenem Hash-Wert Eingabe berechenbar
- zwei verschiedene Eingabewerte mit gleichem Hash-Wert können bei teilweise vorgegebener Eingabe gefunden werden,
  kein Problem für Passwörter (aber für PKI)

Angriffe

Ausgangspunkt ist Annahme einer maximalen Passwort-Länge.

- alle möglichen Passwörter werden druchprobiert (Bruteforce)
- zu allen möglichen Passwörtern werden Hashes vorberechnet (z.B. in DB)
- wegen der Größe der entstehenden Hash-Passwort-Tabelle Verwendung von Rainbow-Tables (s.u.)

Sowohl für das Durchprobieren als auch für die Vorberechnung der Datenbank bzw. der Rainbow-Tables:

$\longrightarrow$ Länge & Zeichenvorrat des $\underbrace{\text{Eingabewertes}}_{\text{u.A.(?) Passwort}}$ der Hashfunktion entscheidend

Rainbow-Tables (vereinfacht)

1. Es werden Ketten von Passwort, Hash, Hash vom Hash, ... berechnet
2. Letzter Hash ($= \mathrm{hash}^n(\mathrm{pw})$) und Passwort werden gespeichert:

- Tabelle mit allen möglichen Passwörtern in den Ketten und
- möglichst überschneidungsfreien Ketten.

Beispiel:

- Password ist eine Ziffer außer 4, 5
- „Hash"-Funktion: $H(m) \equiv m + 6 \mod 10$
- Rainbow:

Value								Key
0	$\xrightarrow{H}$	6	$\xrightarrow{H}$	2	$\xrightarrow{H}$	8	$\xrightarrow{H}$	4
1	$\xrightarrow{H}$	7	$\xrightarrow{H}$	3	$\xrightarrow{H}$	9	$\xrightarrow{H}$	5

Rainbow-Tables (vereinfacht)

4. zum zu knackenden Hash wird ebenfalls Kette berechnet,

   Bsp. 2: $2 \mid \xrightarrow{H} 8 \xrightarrow{H} 4 \xrightarrow{H} \ldots \xrightarrow{H} \ldots$

   bis Wert als Key des Rainbow-Tables vorkommt. Dieses ist bei einer Kettenlänge n und vollständiger Abdeckung spätestens nach $n - 1$ Hash-Berechnungen der Fall.

5. Zu dem gefundenen Key werden per Rainbow-Table-Lookup der Eingabe-Value ermittelt und danach Hash-Berechnungen bis zur Kettenlänge durchgeführt:

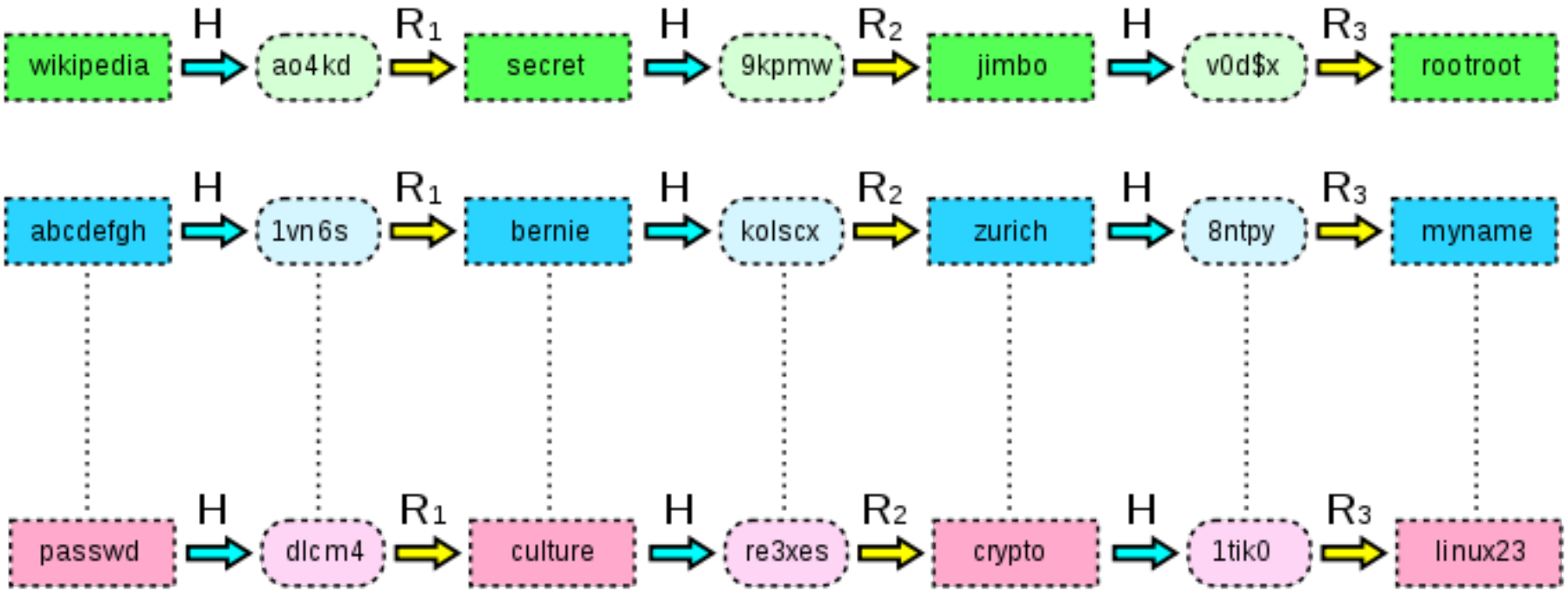
   Bsp. 2:

Hash								PW
2	$\xrightarrow{H}$	8	$\xrightarrow{H}$	4	$\xrightarrow{\text{Lookup}}$	0	$\xrightarrow{H}$	6

Tabelle und Verfahren firmieren unter dem Stichwort *rainbow table*.

Rainbow-Table

Um Abdeckungsziel der Ketten und Passwort-Format des Hash-Wertes zu erreichen, werden zusätzlich Reduktions-Funktionen genutzt.



Quelle: http://de.wikipedia.org/wiki/Datei:Rainbow_table1.svg

Salt

Als Gegenmaßnahme gegen Rainbow-Tables hilft ein *salt*:
Dem Passwort wird ein Zufallswert zugefügt, der mit dem Hash zusammen hinterlegt wird.

In `/etc/shadow`: $\underbrace{\$1\$}_{\text{md5}}\underbrace{\text{ZQQ46fZr}}_{\text{salt}}\$\underbrace{\text{qLbDI9TcwICXMyp1Tbsxw1}}_{\text{hash}}$

```
perl -e 'use Crypt::PasswdMD5; print unix_md5_crypt("otto","ZQQ46fZr");'
```

- Zufallswert verlängert nicht das Passwort,
- wohl aber die Ausgangsmenge für die Erstellung der Rainbow-Tables.

⟶ Hashes nicht unnötig lesbar machen

⟶ Hashes nur mit Salt verwenden

⟶ Salt nicht zu kurz wählen

Grundlagen: Challenge-Response

Ablauf (bei vorher festgelegtem Nutzernamen):

1. Server generiert Zufallstext Z (ggf. inkl. Zeitstempel)
2. und sendet den an Client

clienseitig:

3. Eingabe des Passwortes P_C
4. Verrechnung mit Zufallstext: $E_C = \text{hash}(Z + P_C)$

serverseitig:

3. gespeichertes Passwort P_S
4. Verrechnung mit Zufallstext: $E_S = \text{hash}(Z + P_S)$

5. Client sendet Ergebnis E_C an Server
6. Server vergleicht mit eigenem Ergebnis: $E_C == E_S$?

$\longrightarrow$ Passwort muss dafür auf Server im Klartext *(shared secret)* vorliegen

Anwendung: z.B. SASL-Methode CRAM-MD5 in smtp oder imap

Challenge-Response

Vorteil Passwort kann auch ohne Verschlüsselung nicht abgehört werden

Nachteil Passwort muss dafür auf Server im Klartext vorliegen

Challenge-Response mit Hash

Bei Windows wird ein Challenge-Response nicht mit dem Passwort sondern mit dem MD4-Hash verwendet:

+ Klartext-Passwort muss nicht auf Server liegen
- MD4-Hash hat aber gegenüber Server die Rolle eines Klartext-Passworts
+ Der MD4-Hash als Passwort lässt sich nur an einem manipulierten Client nutzen, nicht an einem unveränderten Windows-Client

Bruteforce

Bruteforce-Angriffe probieren, Passwörter zu erraten:

- meist aus einer Wörterbuch-Liste
- Gefährlichkeit entsteht durch hohe Versuchsanzahl

Schutzmaßnahmen:

- Versuchsbeschränkung
  - Reduzierung Account-Anzahl (u.A. Sperrung inaktiver)
  - Reduzierung Versuchsanzahl je Client oder Account
  - Einschränkung der erlaubten Zugriffs-Quellen
  - Reduzierung über die Zeit: regelmäßiger Passwortwechsel
- Passwort-Komplexität (s.u.)
- Logging & Log-Auswertung bei lokalen Zugängen

Auslesen

Problematisch ist es, wenn Klartext-Passwörter oder Hashes unbefugt ausgelesen werden können.

Z.B. kann das Tool ophcrack von einem Boot-Medium aus die Windows-SAM-Datei (Security Account Manager) mit den Hashes lesen und insbesondere LM-Hashes sehr schnell knacken.

Schutzmaßnahmen:

- restriktive Dateirechte (wie z.B. bei `/etc/shadow`)
- physischer Schutz des Rechners & Absicherung des Bootvorgangs, verhindert Boot von separatem Medium und Umgehung der Beschränkungen
- Auslagerung auf einen Authentifizierungsserver (LDAP, Kerberos)
- Lokale Admin-Passwörter je Rechner individuell und anders als den Domänen-Admin (analog root)

Einfachheit der Passwörter

Passwörter, die nicht durch (wirksam sichergestellte) Versuchsbeschränkung vor Bruteforce-Angriffen geschützt sind, müssen zwingend komplex sein:

- Mindestlänge
- Sonderzeichen (eher nicht wegen des Zeichenvorrats sondern als „Abwehr" einfacher Wörter)
- nicht in Wörterbuch
- nicht Daten des Nutzers

⟶ Aufschreiben von Passwörtern auf Zetteln oder in speziell verschlüsselnden Passwort-Managern ist evt. das kleinere Übel.

R|R|Z|N

Umsetzungshinweise

Windows

Komplexität	über die Kennwortrichtlinien in den GPOs
PW-Wechsel	Account-Einstellungen

Linux

Bruteforce	vgl. ssh-Vortrag nachher
Komplexität	per PAM-Modul (`pam_cracklib`)
PW-Wechsel	per `passwd -e`, gespeichert in `/etc/shadow`
Wörterbuch	Crack-Tool z.B. John-the-Ripper (PWs aber personenbez. Daten)

... aber viel auch organisatorische Maßnahmen und Nutzeraufklärung

Klartext

Klartext-Passwörter sind angreifbar durch:

Man in the Middle

- In Wlan-Hotspots, die mit Captive-Portals arbeitet, ist die Netzwerkverbindung i.Allg. unverschlüsselt und leicht abzuhören.
- Im Netzwerk sind Angriffe im LAN am wahrscheinlichsten & einfach.

Fehlende Serverauthentifizierung

Z.B. ein manipulierter DNS-Eintrag oder eine veränderte hosts-Datei kann Nutzer auf fremden Server umlenken, an den dann das Passwort übermittelt wird.

R|R|Z|N

Transportsicherung

Challenge-Response

Challenge-Response verhindert nicht eine MitM-Attacke gegen die Datenkommunikation oder die Übernahme der authentifizierten Verbindung, wohl aber das Passwort-Abgreifen.
Zufälligkeit & Timeouts sind wichtig zur Verhinderung von Replay-Attacken.

TLS

TLS ist als der goldende Weg anzusehen, weil

- die Kommunikation gesichert erfolgt und
- der Server sich authentisiert.

⟶ Mail-Klienten auf „zwangsweise TLS“ bei imap & smtp konfigurieren

Gefährdung & Maßnahmen: Client

Die Client-Sicherheit ist immer ein unsicherer Faktor

- Ist auf dem Client ein Keylogger installiert, ist das Passwort kompromitiert.
- Malware kann ssh-Keys oder private Schlüssel entwenden.

Bedingt sind Gegenmaßnahmen möglich

- Grafische Passwort-Eingaben (durch Mausklick auf Bilder), am Besten mit sich ändernder Zeichenanordnung, kann helfen, allerdings übermitteln einige Malware-Tools die angeklickten Bilder an Angreifer.
- Hinterlegte ssh-Keys oder TLS-Schlüssel müssen durch Passphrase geschützt werden, gegen diese ist dann aber ein Offline-Bruteforce-Angriff möglich und nicht feststellbar.

Physischer Zugriff

Ist das Booten von fremden Medien (CD-Rom, USB-Stick) möglich, sind lokale Accounts gefährdet:

- durch Passwort-Reset
- durch Knacken abgelegter Passwort-Hashes

Art des Clients

Entscheidend ist natürlich auch die Art des Clients:

- Büro-Rechner bzw. dienstliches Notebook
- Privat-Rechner
- Internet-Cafe

Schutzwürdige Zugänge benutzt man nur an vertrauensvollen Rechnern, kurz: kein Online-Banking im Internet-Cafe ...

Mehrfachnutzung

Passwort-Mehrfachnutzung ist bis auf wenige Ausnahmen schädlich.

Ausnahmen

- gleichartige Systeme in einheitlicher Verwaltung
- echte SSO-Lösungen wie Shibboleth sind keine Mehrfachnutzung

individuelle Passwörter

- lieber verschiedene, komplexe Passwörter aber diese dann durch Master-Passwort geschützt abspeichern (nicht Banking o.Ä.)
- Häufig werden Passwort-Synchronisationstools als SSO-Lösungen vermarktet, nicht ratsam.

Komplexität

Es gibt ein paar Strategie zur Erzeugung *und* Merkbarkeit von Passwörtern, z.B. von einem Satz die Anfangsbuchstaben der Wörter + Satzzeichen nehmen:

`v9b10:Rv` = von 9 bis 10: Rauchen verboten

(mehr/Details: `http://www.rrzn.uni-hannover.de/pw_used.html`)

notwendiger Komplexitätsgrad

Je schlechter automatisiert probiert werden kann, je restriktiver nach einer gewissen Zahl von Fehlversuchen gesperrt wird, desto geringer ist die Komplexitätsanforderung.

Aufschreiben

Die meisten Passwort-Policies verbieten das Aufschreiben von Passwörtern.

aber: Aufschreiben oder Abspeichern von Passwörtern ist häufig im Vergleich zu mehrfachverwendeten oder zu leichten Passwörtern das kleinere Übel.

Wenn aufschreiben, dann mit Vorsicht:

- Karteikasten im verschlossenen Büroschrank
- Passwort-Store, der mit Master-Passwort verschlüsselt (z.B. Firefox)
- Tool PwdHash: Hash(Master-Passwort + URL) als Passwort
- ggf. nur halb aufschreiben, halb merken; Merkteil für mehrere gleich
- ggf. „Out-of-Band"-Gerät (z.B. KeePass auf Java-Handy)
- nicht bei hohem Schutzbedarf (Bank-PIN)

Die eine Passwort-Policy?

Wird immer wieder vorgeschlagen oder verlangt, z.B.:

- Der Benutzer hat sein Passwort geheim zu halten. Idealerweise sollte das Passwort nicht notiert werden.
- Das Passwort muss mindestens 8 Stellen lang sein.
- Das Passwort muss mindestens einen Buchstaben und mindestens eine Ziffer oder ein Sonderzeichen enthalten.
- Das Passwort ist regelmäßig, spätestens nach 360 Tagen, zu wechseln …
- Neue Passwörter müssen sich vom alten Passwort, über mehrere Wechselzyklen hinweg, signifikant unterscheiden.

(IT-Sicherheitsrichtlinie der FU-Berlin 2008, M1.11)

Die eine Passwort-Policy?

Kann aber in Wahrheit nur Richtwert / Hinweis sein.[1]

Eine Passwort-Policy muss von der Anwendung, dem angestrebten Sicherheitsniveau und den möglichen Angriffen abhängen.

WLAN WLAN-Passwörter sind heutzutage schon fast maschinenbezogen, werden einmal eingegeben und auf dem Gerät gespeichert
⟶ lieber komplex, evt. vorgeneriert & nicht wählbar

Web http(s)-Zugänge (Groupware, Forum etc.) hat man viele
⟶ Abspeichern je nach Schutzwürdigkeit fast der einzige Weg

Windows Häufig kann der Login nur lokal an administrierten AD-Clients mit Versuchsbeschränkung und Logging genutzt werden
⟶ kürzeres, nicht ganz so komplexes Passwort evt. ausreichend

[1] Was auch in der FU-Richtlinie an den eher vagen Vorgaben abzulesen ist.

Augenmaß

Zu restriktive Forderungen kann der Nutzer nicht erfüllen. Der Nutzer wird sie versuchen zu umgehen, was meist deutlich schlechter ist als eine schwächere Policy.

Beispiele

Admin sehr priviligierte Accounts für

- Betriebssystem (Administrator, root)
- Datenbank (DB-Admin), Mail (Rollen)
- AD-Administrator, Novell-Supervisor, LDAP-Admin

Appl. von Anwendungen nicht Menschen genutzte

- ssh-Key für Dateiübertragung, Dienste-Accounts (z.B. www-data, Sophos)
- DB-Account einer LAMP-Anwendung, Mail-Account eines TTS

Client dezentral genutzte Passwörter

- Bios-Passwort, Passwort im Booloader (z.B. bei PXE, grub)
- lokaler Administrator

Grundsätze

Admin-Accounts:

- Nachvollziehbarkeit durch Personenbeziehbarkeit der Nutzung
  z.B. kein Remote-Root-Login, erst ssh auf persönliches Account & su
- Passwort-Hinterlegung und geordnete Änderung
  Vertretung gewährleisten, alle Nutzer müssen es kennen, Personalfluktuation
- Nutzung gering halten
  z.B. häufige Kommandos per sudo aufrufbar machen,
  gemeinsame Mail-Accounts mit Shared-Folders realisieren

Application-Accounts:

- in Programmen hinterlegte PWs lang & komplex wählen
  evt. sogar auf Keys (ssh, TLS mit Client-Authentifizierung) gehen

für alle Arten von Funktions-Accounts:

- Überwachung / Logging besonders wichtig

Passwort-Management

Admins haben viele Systeme, eigentlich gilt das Übliche:

- verschiedene Passwörter, gleich höchstens bei gleichen Systemen
  (gleich: gleicher Systembetreiber, ähnliche Daten, ähnlich Schutzwürdigkeit)
- Passwort-Hinterlegung (Vergessen, Vertretung)
- regelmäßige Änderung, komplex etc.

Aber Admins haben besondere Situation: Merken völlig unmöglich!

- Passwörter aufschreiben ist notwendig & legitim
  - im Karteikasten / Büchlein im verschlossenen Büroschrank
  - in einem Passwort-Safe wie KeePass
- Nutzung von ssh-Keys und ssh-Agent
- Zentrale Instanz wie LDAP, Kerberos, AD — dann aber häufigerer Passwort-Wechsel und trotzdem sinnvolle Einheiten; was bei Netzausfall?

Literatur

- DuD 10/2009, S.620ff.: „Mindeslängen von Passwörtern und kryptographischen Schlüsseln“
  http://www.springerlink.com/content/rkx4415304666073/fulltext.pdf
- Dud 7/2009, S.425ff.: „Passwörter – fünf Mythen und fünf Versäumnisse“
  http://www.secorvo.de/publikationen/passwortsicherheit-fox-schaefer-2009.pdf
- <kes> Nr. 6, 2009: „Die Schlüssel zum Königreich“
  http://www.kes.info/archiv/heft/abonnent/09-6/09-6-068.htm