Regelbasierte Passwortgeneratoren für Skripte

Merkwürdig

Tam Hanna, Susanne Nolte

Automatisch Passwörter zu erstellen, die nicht zu einfach zu knacken und für Benutzer nicht zu schwer zu merken sind, gehört nicht zu den einfachsten Aufgaben.

Ideale Passwörter sind keine echten Wörter, diesen vom Aufbau her aber so ähnlich, dass sie im Gehirn des Nutzers an bekannten Ausdrücken „andocken“ und so nicht vergessen werden. Erstmals befasste sich die 1975 von der US-Regierung beauftragte und von Morrie Gasser durchgeführte Studie „A Random Word Generator for Pronounceable Passwords“ mit dem automatischen Generieren aussprechbarer Wörter [a] (siehe „Alle Links“).

Inzwischen gibt es viele mit Regelsystemen arbeitende Passwortgeneratoren. Da sich weder die im Web zu findenden Generatoren wie pwgen.us, olspwgen.php oder die Erweiterung des *pwgen*-JavaScript-Ports von FM4DD noch grafische Anwendungen wie PWGen für Windows für das Einbinden in Skripte eignen, seien hier nur die CLI-Werkzeuge behandelt.

Der Klassiker ist *pwgen* von Theodore Ts'o. Er ist seit Jahr und Tag Bestandteil etlicher Unix- und Linux-Distributionen. Wer *pwgen* ohne Optionen ausführt, bekommt einen Block in Spalten und Reihen angeordneter zufällig generierter Passwörter mit acht Zeichen inklusive Großbuchstaben und Zahlen zurückgeliefert.

Geht die Ausgabe nicht an *stdout*, gibt das Programm nur ein Passwort zurück. Auch die Optionen verändern das Verhalten des Generators. Hinter ihnen lässt sich erst die Passwortlänge und dann die Zahl gewünschter Kennwörter in Form allein stehender Zahlen angeben. *pwgen –B 10 1* etwa generiert ein einzelnes zehnstelliges Passwort, das keine „Konfliktzeichen“ enthält, in dem also keine verwechslungsanfälligen Zeichen wie das kleine l und die Ziffern 1 oder 0 und ein großes O vorkommen. Da das Reduzieren der Menge erlaubter Zeichen die Zahl möglicher Passwörter und ihre Qualität einschränkt, sollte man diese Option nur für Anwender mit Sehschwäche wählen.

Großbuchstaben kann man mit der Option *–A* oder *--no-capitalize* ausschließen, Zahlen mit *–0* oder *--no-numerals*. Passwörter mit Sonderzeichen erzeugt man mit der Option *–y* oder *--symbols*, regellose, also schwer merkbare Maschinenpasswörter mit *–s* oder *--secure*. Reproduzierbare Ergebnisse erhält man mit dem Argument *–H* oder *--sha1=/<pfad>/<datei>*. Dann verwendet *pwgen* den SHA1-Hash der Datei und liefert beim nochmaligen Aufruf mit denselben Argumenten dasselbe Ergebnis. Benutzt man die Option *–H* direkt auf der Kommandozeile, sollte man daran denken, dass Ort und Name der Datei in der *bash*-History gespeichert bleiben, und den Eintrag gegebenenfalls entfernen.

Gänzlich anders zu handhaben ist *apg*. Da es auf den Arbeiten von Gasser fußt, sind die damit generierten Passwörter für englischsprachige Personen aussprechbar. Eine Hilfe zur Aussprache liefert *apg* gleich mit, etwa:

```
Naug1memU (Naug-ONE-mem-U)
manMadd5 (man-Madd-FIVE)
```

Auf die Regeln aussprechbarer Wörter verzichtet *apg*, wenn man es mit *–a 1* in den Random-Modus schaltet. Insgesamt geht es flexibler mit Vorgaben um als *pwgen*. Mit den Optionen *–m* und *–x* lassen sich die minimale und die maximale Passwortlänge festlegen, mit *–n* die Zahl der vorgeschlagenen Kennwörter.

Variantenreich

Für die Art der verwendeten Zeichen ist *–M* zuständig. Die zugehörigen Argumente definieren die jeweiligen Modi, in denen Klein- und Großbuchstaben, Zahlen und Sonderzeichen vorkommen müssen (*–M LCNS*) oder sollen (*–M lcns*). Will man bestimmte Arten von Zeichen nicht haben, etwa Zahlen und Sonderzeichen, muss man den Default-Modus *–M lcns* mit eigenen Angaben überschreiben, zum Beispiel *–M c* – Kleinbuchstaben lassen sich im regelbasierten Modus nicht abwählen. Einzelne Zeichen, beispielsweise O, 0, l, I und 1, schließt man mit *–E O0lI1* aus. Dadurch lässt sich *apg* auch für andere Zwecke einsetzen: Zehn zufällige, hexadezimal anmutende, 32 Zeichen lange Strings etwa lassen sich mit *apg –a 1 –n 10 –m 32 –M nl –E ghijklmnopqrstuvwxyz* erzeugen.

apg schöpft seine Entropie aus mehreren Quellen: Wer auf den Parameter *–c <string>* verzichtet, den fordert *apg* zur Eingabe von Entropiedaten auf. Da es sich zusätzliche Daten aus */dev/random* beschafft, führt das mehrfache Aufrufen von *apg –c <derselbe_string>* trotzdem zu unterschiedlichen Ergebnissen.

Mit korrekten Grammatiken arbeitende Passwortgeneratoren können mitunter Wörter erzeugen, die in der realen Sprache vorkommen. Das lässt sich durch den Rückgriff auf ein Wörterbuch vermeiden, das alle bekannten Wörter auflistet. Neben klassischen Wörterbüchern mit dem Schema ein Wort pro Zeile nimmt das Programm Bloom Filter Files entgegen [e].

Der aus den Arbeiten von Gasser hervorgegangene Generator war zum Zeitpunkt seines Erscheinens revolutionär. Die verwendeten Strukturen können aber prinzipbedingt nicht mit den Resultaten einer aus einem Wörterbuch der Zielsprache generierten Logik mithalten. Dies zeigt sich an den generierten Passwörtern, die nur leidlich englisch wirken.

Bessere Ergebnisse liefert *gpw*. Sein Algorithmus lehnt sich an das Konzept des Bi- oder Trigrammkatalogs an, das durch Analyse eines bestehenden Wörterbuchs entsteht. Diese Abfolgeinformationen beschreiben einen Baum, den dann ein Zufallsgenerator zwecks Generierung von Wörtern „durchforstet“ [f]. Heraus kommen Pseudowörter wie *plusemau*, *latorede*, *bratinge* oder *stornarf*. Da Wörter nur höchst selten Zahlen und Sonderzeichen enthalten, beschränkt sich *gpw* auf das Generieren alphabetischer Passwörter. Das vereinfacht auch die Aufrufstruktur: *gpw <pw-zahl> <pw-laenge>*. (sun)

Tam Hanna

ist Gründer der Tamoggemon Holding.

Alle Links: www.ix.de/ix1510151