Zufallszahlen

Robert Huber *

November 2008

Zufallszahlen sind in der mordenen Welt von enormer Bedeutung. Sie sind in vielen wissenschaftlichen und praktischen Bereichen, ein fundamentaler Bestandteil der täglichen Arbeit.

Ein Problem bei der Erzeugung von Zufallszahlen ist, dass echte Zufallszahlen nur mit großen Aufwand oder teuer erzeugt werden können. Abhilfe dabei schaffen mathamatische Algorithmen, welche aber nur Pseudozufallszahlen hervorbringen. Diese Pseudozufallszahlen haben erhebliche Nachteile, wie die Reproduzierbarkeit oder die endliche Periode. Dennoch sind Pseudozufallszahlen der Ausgangspunkt für die meißten modernen Anwendungen wie bei Algorithmen zur Verschlüsselung von Daten oder Erzeugung von Schlüsseln. Um eine möglichst gute Zufälligkeit zu gewährleisten verwenden Computer zufällige Signale der Hardware, wie Benutzereingaben, Netzwerkverkehr oder Zugriffszeiten, um ihre Zufallszahlen zu eryeugen und damit yu verbessern. Trotzdem kann keine absolute Zufälligkeit gewährleistet werden und so beleibt immer eine kleine Schwachstelle in jedem Algorithmus der auf Pseugozufallszahlen beruht.

Schlagworte: Zufallszahlen, Pseudozufallszahlen, Linux Psydozufallszahlengenerator, Linux-PRNG.

1 Einleitung

Jüngste Ereignisse, mit der Sicherheitslücke in openSSH, die eine Brute-Force Attacke auf die Schlüssel zuließen sind auf einen Fehler im Code des Zufallsgenerators zurückzuführen. Dabei war die Zufallszahl die zum erzeugen des Schlüssels verwendet wird, duch einen Fehler in einen kleineren Bereich eingeschränkt. Man sieht an diesem Vorfall wie wichtig ein gut durchdachter Zufallsgenerator für die Kryptographie und alle andere Anwendungen ist.

Selbst in Vorgängen die nichts mit Zufall zu tun haben, spielen Zufallszahlen ein große Rolle. Klassische Beispiele für Zufallszahlen sind Glücksspiele wie Roulette, ein

*E-mail: e0826943@student.tuwien.ac.at

einfacher Münzwurf oder das im Moment so beliebte Pokern. An dem letzten Beispiel sieht man auch das der Zufall sehr eng mit der Wahrscheinlichkeit verknüpft ist. Und so ist es nicht verwunderlich das Zufallszahlen bei der Berechnung des Wetters oder der Mindestbetriebsdauer von Geräten, um nur einige Beispiele zu nennen, eine wesentliche Rolle spielen. Dabei werden Berechnung in sekundenschnelle hunderte und tausende Male wiederholt und so ein enormer Vorrat von Zufallszahlen verbaucht.

Dabei tritt das erste Problem auf, um echte Zufallszahlen zu erzeugen, müssen echte Zufallsexperimente (Münzwurf, Roulett, ..) durchgeführt oder echte zufällige Ereignisse in der Natur (atomarer Zerfall, thermisches Rauschen, Hintergrundstrahlung, ..) gemessen werden. Diese Vorgänge oder Messungen sind aber Zeitaufwendig und meißt kompliziert und teuer. Der Bedarf an Zufallszahlen brachte die Idee auf, Zufallszahlen aus mathematischen Algorithmen zu erzeugen.

Diese Algorithmen lassen sich einfach am Computer implementieren und schnell durchführen. Jedoch können diese Pseudozufallszahlen nicht zufällig sein, da sie einem mathematischen Gesetz folgen. Jedoch ist in praktischen Anwendung meißt keine absolute Zufälligkeit von Nöten und je nach Anwendung kann die eine oder andere Beschränkung vernachlässigt werden.

Nur einige Anwendungesbereiche von Zufallszahlen seihen hier erwähnt:

- Simulationen
- künstliche Intelligenzen
- Kryptographie
- Computerspiele
- uvm

Ein wichter Begriff bei der Erzeugung von Pseudozufallszahlen ist die Entropie. Im Zusammanhang mit Zufallszahlen beschreibt die Entropie die Anzahl der Bits an Zufälligkeit.

Da dieser Begriff etwas schwerer verständlich ist, hier ein kleines Beispiel: Wie haben eine Zahl aus 12 Bit, die ersten 4 Bit sind immer 1 und dies ist bekannt, daher ist die Entropie der Zahl 8 Bit.

Diese Arbeit behandelt kurz den Begriff der SZufallszahlünd geht anschließend auf die Erzeugung von Zufallszahlen sowie Pseudozufallszahlen und deren Anwendung, besonders im Bereich der Kryptographie ein. Außerdem wird die Bedeutung und das Problem der Erzeugung von Pseudozufallszahlen besondern auf den heutigen Computern behandelt. Dabei wird als Beispiel der Pseudozufallszahlengenerator (PRNG) des Linux Kernels heran gezogen und auf dessen Aufbau, Sicherheit und Schwächen eingegangen.

2 Hauptteil

2.1 Was ist eine Zufallszahl

Was ist Zufall? - Diese eher etwas philosophische Frage lässt sich nicht so einfach beantworten. Ein Ansatz wäre zu sagen, dass eine Menge von Zufallszahlen durch Komprimierung nicht wesentlich verkleinert werden kann, da keine Zusammhänge zwischen den Zahlen gefunden werden können. Aber für praktische Anwendung ist keine vollige Zufälligkeit von nöten, was es auch erst möglich macht Zufallszahlen effektiv zu erzeugen und zu verwenden.

2.1.1 Definition von Zufallszahlen

Auch wenn die mathematische Deffinition einer Zufallszahl eindeutiger und Deffinition aus dem Duden komplizierter zu lesen ist, soll uns diese Deffinition reichen:

> Zahlen, die aus Zufallsexperimenten entstanden sind, werden als echte Zufallszahlen bezeichnet. Zufallszahlen, die mit Hilfe mathematische Algorithmen erzeugt wurden, werden als Pseudozufallszahlen bezeichnet.[1]

2.1.2 Vergleich zwischen Echten und Pseudo-Zufallszahlen

Im verlgeich zu Pseudozufallszahlen sind echte Zufallszahlen nicht generierbar, sie lassen sich nur durch Zufallsexperimente erzeugen und aufzeichnen oder durch Vorgänge in der Natur messen. Dabei können alle Arten von Quellen herangezogen werden. Zufallsquellen aus der Natur, wie der Atomare Zerfall oder thermisches Rauschen, oder von Zufallsexperimenten, wie Roulette oder anderen Glückspielen, benötigen eine relativ lange Zeit um zu erzeugt und gemessen zu werden. Für alle Quellen gilt zusätzlich, um so länge zwei Messungen zeitlich auseinander liegen um so unabhängiger sind die Zahlen voneinander.

Aber es gibt auch einfach zu messende Quellen, die nicht sofort ersichtlich sind und auch dem Zufall unterliegen wie die Tastenanschläge auf einer Tastatur oder den Traffic auf einer Netzwerkkarte. Diese Quellen sind besonders für dein Einsatz am Computer geeignet und es wird auf sie später noch genauer eingegangen.

Echten Zufallszahlen haben aber signifikante Vorteile gegenüber Pseudozufallszahlen, so hängen ihre Ergebnisse praktisch nicht vom vorherigen Ereignissen ab und es wird kein Startwert benötigt, der die Ergebnisse reproduzierbar macht, wie es bei Pseudozufallszahlen der Fall ist. Um möglichst gute, aber auch eine ausreichende Menge an Zufallszahlen errechnen zu können, werden echte Zufallszahlen als Ausgangspunkt für die Berechnung von Pseudozufallszahlen heran gezogen. Diese Methode findet sich in jedem Computer wieder aber auch in allen anderen Arten von Geräten die aus Hard- und Software bestehen.

[1] [1] 2.1 Definition einer Zufallszahl

2.2 Pseudozufallszahlen

Pseudozufallszahlen werden mittels eine mathematschen Algorithmus, ausgehend von einem echten zufälligen Startwert gebildet. Dabei gibt es mehrere Arten solche Pseudozufallszahlen zu erzeugen.

Viele Algorithmen die früher verwendet wurden sind heute nicht mehr stand der Technik, da sie eine zu kleine Periode haben. Jeder Algorithmus beginnt sich nach einer gewissen Anzahl von Ausgaben zu wiederhohlen dies lieght im begrenzten Zahlenbereich der zur Verfügung steht. Da jeder Pseudozufallsgenerator periodisch ist, mussten Algorithmen mit einer möglichst großen Periode gefunden werden. Andere Kriterien für Pseudozufallsgeneratoren sind, dass die Ausgabe nicht vorhersagbar ist (wie bei 1,2,3...) und das die Ausgabe möglichst gleich über den gesammten Zahlenbereich verteilt ist. Wenn zum Beispiel Zahlen von 0 bis 1 erzeugt werden sollen 25% der Zahlen zwischen 0 und 0,25 liegen. Auch soll nicht von einem Zufallswert auf den alle anderen Werte oder dem Startwert geschlossen werden können.

Einige dieser veralteten Algorithmen sind der Middle Square Generatoröder der Ëumel Generator". Der heute meißt verwendetste Pseudozufallsgenerator (PRNG) ist der Kongruenzgenerator.

$$x_{i+1} \equiv a_0 * x_i + ... + a_j * x_{i-j} + b \quad (mod\ m) \tag{1}$$
$$z_i = x_i/m\ z_i \in [0,1] \qquad z...\text{Zufallszahl}$$

Wir wollen nicht weiter auf die Details eingehen, die sich doch sehr ins mathematische ziehen. Hierbei sei auf [1] verwiesen.

2.2.1 Güte von Pseudozufallszahlen

Wenn man nun Pseudozufallszahlen hat stellt sich die Fragen woher man weiß ob diese auch Zufällig sind. Um die Güte einer Zufallsfolge zu bestimmen,ist zum Beispiel der Mensch kein geeignetes Messinstrument. Da der Mensch Zufälligkeit nicht objektiv bewerten kann. Zum Beispiel würde man meinen wenn bei einem Würfelwurf öfers hintereinader die gleiche Zahtl entsteht, sei dies kein Zufall, jedoch ist die Wahrscheinlichket für eine Folge von 2 oder 3 gleichen Zahlen relativ groß. Hierbei sind Statistiken schon ein wesentlich geeigneteres Mittel.

Man kann die Tests für Pseudozufallszahlen in drei grundlegende Arten unterteilen:

- empirische Tests
- theoretische Tests
- spektrale Tests

Ingesammt gibt es duzende an Testverfahren die angewant werden können, dabei ergeben einige, an der selben Methode, sehr gute Resultate andere eher weniger Gute. Es kommt nun auf den Anwendungsbereich an für die die Pseuozufallszahlen bestimmt sind. Um so mehr Test ein Generator besteht um so universteller ist einsetzbar und um so besser ist er.

Ein Zufallsgenator wird auf die Gleichverteilung der Ausgabe überprüft. Bei den Ausgaben 0 und 1 sollten die Hälfte der Zahlen 0 und die andere Hälfte 1 sein. Auch sollte alle Permatuatione von zweier, dreier, .. Paaren mit gleicher Wahrscheinlichlkeit vorkommen. Außderem dürfen keine Sequenzen von Zahlenfolgen vorkommen, sondern die Zahlen beliebig im Wertebereich hin- und her springen.

Um hier noch einige Test zu erwähnen: Chi-Quadrat-Test, Kolmogorow-Smirnow-Test, Run-Test, Permutations-Test, Korrelations-Test, Poker-Test. Zu weiteren Studien sei hier auf [1] verwiesen.

2.3 Pseudozufallszahlengenerator am Computer

Der Linux Zufallsgenerator wurde im Jahr 2004 entwickelt und seitdem stetig weiter verbessert. Der Zufallsgenerator ist Teil des Kernels. Dies hat den Grund, dass der Kernel für seine Arbeit Zufallszahlen benötigt und der Zufallsgenerator Interrupts der Hardware als Quellen benutzt. Der Genenerator kann grob in drei Teile gegliedert werden:

- Das Umwandeln von Ereignissen in Bits
- Das Mixen des Pools mit den gewonnenen Bits
- Das Ausgeben der Zufallszahlen mit Feedback zum Pool

Als Anwender bzw. Entwickler kann man auf die Zufallszahlen über `/dev/random` sowie `/dev/urandom` zugreifen. Ersteres liefert Zufallszahlen mit einer bestimmten Entropie (Grad der Zufälligkeit) zurück. Werden nun öfters schnell hintereinander Zufallszahlen abgefragt sinkt mit jeder Abfrage die Entropie. Ist die geforderte Entropie nicht vorhanden blockiert das Device. Zweiteres liefert immer Zufallszahlen ohne auf die Entropie zu achten. Dies kann zu vorhersagbaren Zufallszahlen führen und ist für die Zwecke der Kryptographie nicht zu empfehlen, lieftert jedoch bei nicht kritischen Anwendungen schnelle und außreichende Zufallszahlen.

2.3.1 Quellen

Da ein Pseudozufallsgenerator auf einem besimmten Algorithmus beruht, wird mit zufälligen Quellen als Ausgangswert, die Zufälligkeit der Ausgabe gewährleistet. Voraussetzungen für solche Quellen sind, dass sie nicht vohersagbar sind und nicht beeinflusst werden können. Unter Linux gibt es fünf quellen die als Eingang für den Pseudozufallsgenerator dienen:

- Bewegung der Maus
- Tastaturanschlag
- Block Device Operationen
- Interrupts
- Feedback des PRNG selbst

Die ersten beiden Quellen sind auf Server-Systemen meißtens nicht vorhanden. Andere Quellen können durch Angreifer in gewissem Maße beeinflusst werden, worauf wir noch später eingehen werden. Jedoch sind diese Quellen die zufälligsten Events, die auf einem Computersystem (einfach und billig) zu finden sind.

Diese Events werden als einfache Zahlen interpretiert. Je nach Quelle werden unterschiedliche Eigenschaften sowie die Zeit des Auftretens in die Zahl eingerechnet. Diese Zahl wird zum Pool des PRNG (Pseudo-random-number-generators) hinzugefügt und dabei die aktuelle Entropie berechnet. Um so länger der Abstand zwischen zwei Events ist und je zuverläser die Quelle ist, umso mehr wird die Entropie erhöht.

2.3.2 Pools

Es gibt drei verschieden Pools die die Werte des Zufallsgenerators enthalten. Der Primary Pool, mit einer größe von 512Byte bekommt seine Werte direkt aus den Events des Computers. Er dient anschließend als Quelle für den Urandom Pool sowie dem Secondary Pool (beide mit einer Größe von 128Byte). Jede Entnahme aus einem der Pools verändert den Inhalt des Pools selbst (Feedback). Der Secondary Pool ist die Quelle von `/dev/random` und der Urandom Pool ist die Quelle von `/dev/urandom`. Siehe Bild 1

Jeder der Pools hat einen eigenen Zähler für die Entropie. Bei jeder Entnahme wird der Wert um die Anzahl der entnommenen Bit verringert. Bei einer Entnahme vom Secondary Pool wird überprüft ob genug Entropie vorhanden ist, andernfalls wird die Ausgabe verweigert. Beim Urandom Pool spielt änder eine zu niedrige Entropie nichts an der Ausgabe.

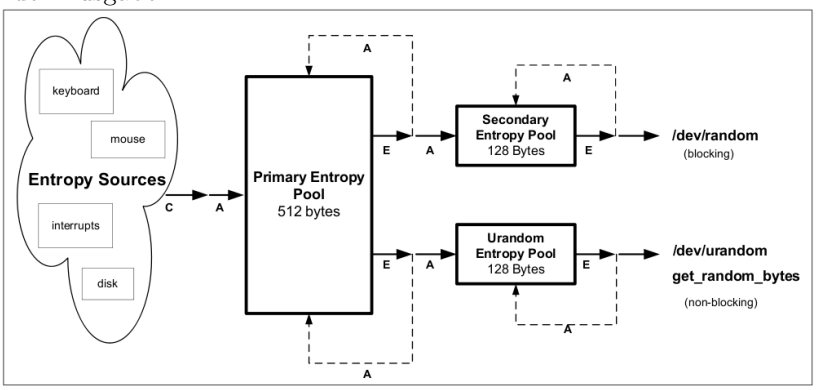

Bild 1[2]: Der Aufbau des Linux-PRNG. Entropie wird den Pools entzogen (E) Dabei wird der Inhalt dises Pools veränder (Feedback - gestichelte Linie) und die Entropie dem nächten Pool hinzugefügt (A)

2.3.3 Abruf von Zufallszahlen

Die Ausgabe einer Zufallszahl über eine der Geräte kann in vier Schritte gegliedert werden:

[2]Analysis of the Linux Random Number Generator `http://eprint.iacr.org/2006/086.pdf`

- Update des Poolinhalts (Feedback), bei Bedarf Nachfüllen durch den Primary Pool
- Überprüfen ab genug Entropier für die Abfrage vorhanden ist
- Hashwert der Ausgabe des Pools berechnen
- Entropiezähler des Pools verringern

Wie in der obigen Liste zu sehen ist werden die Zahlen der Pools nicht direkt an den User übergen, sondern zuert mit Hilfer der SHA-1 Funktion der Hash-Wert gebildet. Dies dient einerseits dazu durch den Ausgabewert keine Rückschlüsse auf den Pool zuzulassen, andererseits wird die Zufälligkeit der Ausgabe erhöht durch die Hashfunktion erhöht.

Der Algorithmus zur Ausgabe der Zufallszahl funktioniert folgendermaßen (siehe Bild 2):

Für die Ausgabe wird der Hashwert der ersten 64Byte gebildet, dieser Wert kommt als Feedback wieder in den Pool zurück. Dannach wird der Hashwert der zweiten 64Byte brechnet und als Initialiserungvektor das Ergebniss des ersten Hashwertes genommen. Nun wurd der Hashwert der außeren 64Byte (bis zur Stelle i-2) gebildet in die die vorherigen Hashwerte schon einfließen. Anschließend wird das Ergebniss auf 80Bit verkürzt und durchgemischt (Folding). Das Ergebniss ist anschließend der Ausgabewert des PRNG.

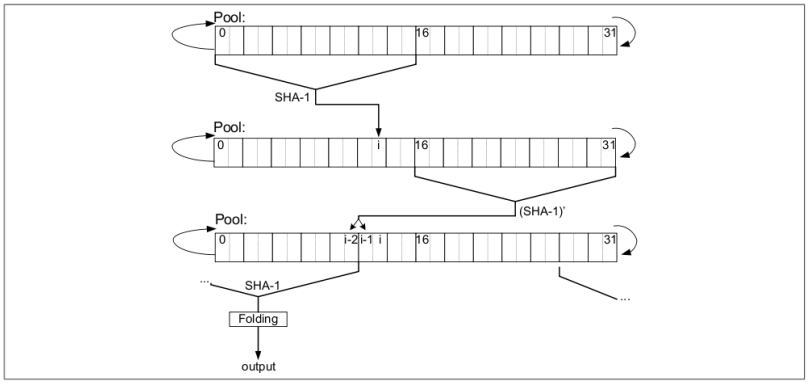

Bild 2[3]:

2.3.4 Sicherheit

Da sich ein System beim booten immer annähernd gleich verhält (Festplattenzugriffe, Interrupts, ..) ist hierbei der Zustand des Zufallsgenerator leichter vorherzusagen, als wenn das System eine gewissen zeit läuft. Deshalb wird für jeden Bootvorgang ein Seedfile angelegt mit dem der Zufallsgenerator initialisiert wird. Das Seedfile wird bei jedem herunterfahren des System mit `/dev/urandom` erzeugt und auf der Festplatte abgelegt. Um denn Fall auszuschließen, dass bei einem Systemabsturz das selbe

[3] Analysis of the Linux Random Number Generator `http://eprint.iacr.org/2006/086.pdf`

Seedfile zweimal verwendet wird, wird gleich nach dem initialisieren des Seedfile neu geschrieben. Gleich nach dem Bootvorgang ist der Zufallswert zwar nicht so stark jedoch lässt eine Initialisierung mit den selben Werten viel schwerwiegender.

Der selbe Schluss kann bei Aufsetzen eines neuen Systems gezogen werden. Daher sollten vielleicht nicht gleich alle Schlüssel bei der installation erzeugt werden.

2.3.5 Angriffsmöglichkeiten

Da der Pseudozufallsgenerator seine Zufälligkeit aus den verschiedenen Quellen auf dem Computersystem bezieht, ist es naheliegt dort einen Angriff zu versuchen. Man kann dabei zwischen zwei arten von Quellen unterscheiden: Schwache Quellen und Bösertige Quellen.

Schwache Quellen haben eigentlich keine bösartige Absicht können jedoch von einem Angreifer missbraucht werden um den Zufallsgenerator eine höhere Entropie vorzutäuschen als er eigentlich hat. Ein Beispiel dazu ist die Netzwerkschnittstelle die von außen mit geziehlten Anfragen beeinflusst werden kann.

Bösartige Quellen werden von einem Angreifer ins System eingebaut um den Wert des Zufallsgenerators zu verändern. Ein Angreifer braucht dazu aber in der Regel weite Rechte auf dem Zielsystem, um zum Beispiel Kernel Module zu laden oder Code zu kompilieren.

Ein einfacher und effektiver Anfriff, um das System zu blockieren, besteht in der häufigen Abfage von Werten von `/dev/random`. Damit werden andere Prozesse die eine Zufallzahl anfordern blockiert. Um dies zu verhinder kann eine Quota-Regelung eingebaut werden. Damit kann ein Benutzer nie den gesammten Pool leeren und so andere Benutzer blockieren.

Auch bietet eine schwache Hash Funktion eine Angriffsmöglichkeit. Sollte man durch den Hashwert auf den Zustand des Poosl Rückschlüsse ziehen können, kann eine Anfreifer weitere Werte mit einer gewissen Genauigkeit vorherberechnen. Die eingesetzte SHA-1 Funktion erzeugt einen Hashwert mit 160Bits aus einem beliebigen Input und wurde bis heute noch nicht gebrochen und ist daher sicher genug.

3 Zusammenfassung

Zufallszahlen werden für alle möglichen Anwendungen in der heutigen Zeit gebraucht. Jedoch ist das erstellen von Zufallszahlen aufwending und teuer. Deshalb griff man mathemathischen Algorithmen um durch sie Pseudozufallszahlen zu erzeugen. Zwar unterliegen Psydozufallszahlen nicht dem echten Zufall jedoch kommten moderne Psydozufallszahlengeneratoren dem schon sehr nahe.

Durch Psydozufallszahlen wird es erst möglich Zufallszahlen effektiv zu nutzen und in Anwendungen einzusetzen. Dies zieht zwar eine Nachteile mit sich, die aber je nach Anwendung vernachlässigt werden können. Eine der wichtisten Anwendung von guten Zufallszahlen bzw. Psydozufallszahlen ist die Kryptographie und dort die Erzeugung von Schlüsseln.

Der Linux PRNG ist ein sehr gutes Beispiel wie solch ein Zufallsgenerator am Computer funktioniert und woher er seine Zufälligkeit bezieht. Eingabegeräte wie die Maus und Tastatur oder die Netzwerkkarte sind sehr gute quellen von zufälligen Zahlen mit denen der PRNG arbeiten kann. Jedoch haben auch diese Quellen ihre Schwachstellen und sind in manchen Fällen nicht so zufällig wie man es gerne hätte. Beim Systemstart zum Beispiel unterscheiden sich die Werte meißt nur um die Uhrzeit wodurch die geliferten Zufallszahlen Entropie (Zufälligkeit) einbüßen. Abhilfe schaffen hierbei Page-Files welche noch zufällige Werte vom letzen Betrieb enhalten.

Der Linux Zufallsgenerator ist Software die schon sehr viele Verbesserungen und Patches erfahren hat und kann daher als sichere Quelle für Zufallszahlen für alle Arten von Anwendungen herangezogen werden.

4 Literatur

Literatur

[1] Stefan Hagmann, **Erzeugung, Test und Anwendung von Pseudo-Zufallszahlen**, Diplomarbeit, Technische Universtität Wien, 2000

[2] Daniel Würsch, **Test von Pseudo-Zufallszahlen**, Maturaarbeit, Kantonsschule Zug, 2002/2003, `http://www.smalllinks.com/6NF`

[3] **Linux-Zufallsgenerator**, hakin9 Nr.1/2008, Software-Wydawnictwo Sp. z o.o, `http://www.hakin9.org/de`

[4] Zvi Gutterman,**Analysis of the Linux Random Number Generator**, The Hebrew University of Jerusalem, 2006, `http://eprint.iacr.org/2006/086.pdf`