Entropie des systemes d'exploitation

Rene Amirkhanian

(email: amirkh_r@epita.fr)

May 13, 2002

Abstract

Les logiciels de cryptographie forte, ou certains services reseaux necessitent la plupart du temps la presence sur le systeme d'exploitation d'un generateur de nombres pseudo-aleatoires fiable afin de garantir une entropie suffisante pour assurer la securite du chiffrement, ou fournir un niveau de securite plus eleve lors de l'attribution de ports par certains services.

Ce document presente le fonctionnement, les besoins et l'importance d'une entropie elevee au sein du systeme d'exploitation afin de fournir un niveau de securite plus eleve au sein meme du systeme.

Contents

1 Introduction 2

2 Nombres aleatoires 2

2.1 Etat indevinable 3

2.2 Generateurs de nombres pseudo-aleatoires 4

3 Les attaques sur les PRNG 6

3.1 Sequence TCP 6

3.2 Attaques cryptographiques 7

4 Conclusion 9

1 Introduction

La generation de nombre aleatoires au sein de tout systeme d'exploitation est utile lors de l'attribution de ports reseaux, creation d'un numero de sequence TCP ou encore a une part essentielle dans la plupart des methodes de chiffrement utilisees a ce jour. Ces operations dependantes du hasard ne doivent pas pouvoir etre predites, bien souvent la securite du systeme lui-meme ou la confidentialite des echanges en dependent. Les generateurs de nombres aleatoires sont un veritable sesame sur bien des portes et fonctions offertes par un systeme.

2 Nombres aleatoires

Au sein d'un systeme d'exploitation, les nombres aleatoires sont souvent utilises. Leur utilisation va de la creation d'identifiant uniques a des entrees d'algoritme cryptographique, tel que:

- Allocation de sin_port dynamique dans bind
- PID des processus
- IDs de datagrams IP,
- IDs de transactions RPC
- IDs de requetes DNS
- Generation des numeros d'inode sur un filesystem tel q'ext2
- Generation de noms temporaires

- Generation de numeros de sequences TCP
- Creation de cles de chiffrement

Certains ordinateurs sont munis d'un vrai generateur de nombres aleatoires, ce n'est cependant pas toujours le cas. La plupart des machines n'ont pas de generateur dedie et ont besoin d'une maniere de generer des nombres aleatoirement qui est suffisament hasardeuse pour ne pouvoir etre predite. En general, cela revient a 3 choses:

- Un etat indevinable; generalement ceci est fait en mesurant des variances dans les timings de peripheriques bas niveau (frappes de touches, tetes de lectures) d'une maniere qu'un adversaire ne puisse controler.
- Un fort generateur de nombres pseudo-aleatoires qui utilise l'etat pour generer des nombres "aleatoires".
- Un grand nombre de bits dans l'etat initial et en sortie de PRNG, un nombre de bits reduits rend le PRNG vulnerable a des attaques "brute force".

2.1 Etat indevinable

Certains systemes d'exploitation utilisent les temps d'interruption de la souris, les informations d'entree/sortie des disques, l'intervalle de temps entre les frappes de touches ou encore les delais et debit reseaux pour initialiser le generateur. Les nombres aleatoires sont disponibles au routines du noyau et sont alors exportes dans notre cas en "userland" par le biais de devices, c'est le cas sous linux ou *BSD sous /dev/random

2.2 Generateurs de nombres pseudo-aleatoires

Un generateur de nombres pseudo-aleatoires (PRNG de l'anglais pseudo-random number generator) fournit aux applications un flux de nombre qui possede d'importantes proprietes pou la securite du systeme:

- Il devrait etre impossible pour un intervenant de predire la sortie du generateur, meme avec la connaissance de la sortie precedente
- Les nombres generes ne doivent pas avoir de cycle qui se repete, ce qui signifie qu'un PRNG doit avoir un cycle tres long.

Un PRNG est normalement juste un algorithme ou la meme sequence initiale de valeur fournira la meme sequence de sorties. Sur un systeme d'exploitation multi-utilisateur il y a de nombreuses sources qui permettent d'initialiser le generateur avec des nombres aleatoirs.

Un PRNG se resumant a un simple algorithme, il va de soi que quiconque connait tout ou partie de l'etat initial de l'algoritme pourrait etre en mesure de predire la prochaine sortie. C'est pour cette raison que des solutions afin d'initialiser les PRNG aleatoirement on ete etudiees. Solutions telles que EGD (Entropy Gathering Daemon) qui surveille l'activite systeme et par le biais de fonctions de hachage renvoit des valeurs aleatoires. Il est eventuellement conseille d'utiliser des fonctions de hachage cryptographique en sortie de PRNG afin d'en ameliorer l'entropie: avec l'utilisation d'une fonction de hachage, meme si le PRNG se revele previsible, l'attaquant doit egalement savoir casser la fonction de hachage.

Les dangers d'un PRNG

Le vrai danger d'un generateur de nombre aleatoire est que la plupart des librairies offrent un grand choix de PRNGs qui sont appropries pour des raisons de securite. Les PRNGs de librairies sont concus pour des simulations, jeux et autres utilisation et ne sont pas suffisament aleatoires pour une utilisation en securite tel que la generation de cles. La plupart des PRNGS de librairies non cryptographique sont une variation de generateurs lineaires ou la prochaine valeur est calculee en fonction de la sortie precedente tel que (aX + b) mod m (ou X est la derniere valeur generee). De tels generateurs sont rapides et appropries pour leurs utilisations de simulation/statistique. Le probleme est que les valeurs futures (bien qu'elles puissent paraitre hasardeuses) peuvent facilement etre deduites par un attaquant. D'autres algoritmes rapides de generation de nombres aleatoires tels que generateurs quadratiques ou cubiques ont aussi ete casses.

L'echec de la creation de nombres aleatoires a ete la source de nombreux problemes dont des trous de securite dans Kerberos, le systeme X Windows et NFS.

Les differents type de hasard

On peut brievement categoriser les generateurs de nombres aleatoires en 5 categories.

- Complexite woooaw-gee - C'est la complexite qui a l'air d'etre hasardeuse pour un utilisateur naif. Par exemple une equation du chaos produisant une image complexe. Les equations du chaos sont des exemple de

PRBGs (pseudorandom bits generators) non securise et qui n'ont pas ete designe comme statistiquement aleatoires.

- fonction random() - Ceci represente une source de valeurs generees algoritmiquements declarees aleatoires par des tests statistiques. Ideal pour Las Vegas ou Monte Carlo.
- Complexite obscure - C'est l'entropie qui n'est encore ni secrete, ni quantique mais dont le designer ne comprend pas le processus qui la genere.
- Entropie secrete - Entropie non aleatoire, mais dont il est, esperons, impossible a l'enemi de capturer ou predire. (chemins d'acces, /dev/random ...)
- Hasard quantique - Aux connaissance d'aujourd'hui, du "vrai" alatoire. Il est considere a ce jour que les evenements quantiques sont guides par un processus completement aleatoire.

3 Les attaques sur les PRNG

Afin de mieux illustrer l'importance d'une entropie importante au sein du systeme d'exploitation, considerons quelques attaques pouvant etre appliquees a leur encontre.

3.1 Sequence TCP

Lors d'une connection TCP/IP vers un hote, l'hote genere un numero de sequence initial (ISN). Ce numero de sequence est utilise entre les deux hotes

afin de suivre plus aisement l'orde des paquets et de s'assurer de la qualite de service.

Des 1985 de nombreuses speculations on ete etablie sur la prediction du prochain ISN, un attaquant pourrait forger une connection spoofee sur l'adresse source d'une machine de confiance, et recuperer ainsi le numero de sequence. Afin de rendre les connections TCP/IP plus integres, il a alors ete defini qu'il devrait pour chaque connection etre assigne un numero de sequence unique et aleatoire. Afin de compromettre une connexion existante en envoyant des donnees malicieuses un attaquant devrait connaitre le numero de sequence de la connection.

Dans son excellent article "Strange Attractors and TCP/IP Sequence Number Analysis", M. Zalewski dcrit l'utilisation d'espace de phase afin de dterminer le prochain ISN: Un espace de phases est un espace de dimension n qui dcrit completement l'etat de n variables. Un attracteur est alors une forme qui est specifique au PRNG employe et revele la complexite de l'algoritme et les dependences des resultats generes. Les illustrations de l'article illustrent parfaitement les differentes complexites des differents PRNG utilises sur divers systemes d'exploitation et expose la difficulte de generer entierement aleatoirement des nombres sur un ordinateur.

3.2 Attaques cryptographiques

La generation de nombre aleatoires est un facteur important dans certains algoritmes de cryptographie, pas seulement au niveau de la generation de cles. Une connaissance des sorties peut severement compromettre les donnes cryptees.

Attaque directe cryptanalitique

Lorsqu'un attanquant est directement capable de faire la distinction entre sortie de PRNG et sortie aleatoire, il s'agit d'une attaque cryptanalitique. Cette sirte d'attaque est applicable a la plupart des PRNGs.

Attaque bases sur les entree

Une telle attaque a lieu lorsque l'attaquant est capable d'utiliser la connaissance des entrees, ou de les controler. Les attaques basees sur les entrees peuvent etre divisees en principalement 2 categories:

- Entrees connues
- Entrees choisies

Ces deux methodes peuvent permettre d'identifier le PRNG voir, de controler/predire la sortie de celui-ci.

Extension d'un etat compromis

Cette attaque considre qu'un etat interne S du PRNG est connu, soit par une fuite accidentelle, une analyse cyptanalytique fructueuse, etc. Ce genre d'attaque est plus a meme d'etre etabli lorsque le PRNG est demarre dans un etat devinable (initialisation pas assez aleatoire). Il est cependant recommande meme si le PRNG est convenablement initialise de considerer qu'occasionnellement la compromission de l'etat S pourrait survenir. Cette situation compromise permet en autre:

- Backtracking Attack - Utilisation de l'etat connu S au temps t pour connaitre les sorties precedentes du PRNG

- Permanent Compromise Attack - Se produit lorsque la connaissance de S au temps t si elle permet de connaitre toutes les sorties passees et futures.

- Iterative Guessing Attacks - Utilise la connaissance de S au temps t et des sorties du PRNG pour connaitre S a $t + e$ lorsque les entrees collectees sur ce laps de temps peuvent etre devinees/connues.

- Meet-in-the-middle Attacks - C'est essentiellement une combinaison d'attaque iterative et de backtracking: La connaissance de S au temps t et $t + 2e$ permet de retrouver S au temps $t + e$.

4 Conclusion

Une forte entropie au sein d'un system d'exploitation est comme nous l'avons vu importante afin de pereniser certains rouages tant au niveau de la securite que d'eventuel cryptages de donnes.

On ne peut donc que suggerer de renforcer l'entropie du systeme graces a notament:

- L'utilisation d'un generateur d'entropie secrete tel que EGD (Entropy Gathering Daemon) ou encore l'utilisation de /dev/random afin d'initialiser le PRNG

- Le choix d'un PRNG efficace

- L'utilisation d'une fonction de hachage cryptographique en sortie

Bibliography

[1] Carl M. Edison, *Cryptographic Randomness*, (2001)

[2] M. Zalewski, *Strange Attractors and TCP/IP Sequence Number Analysis*, (2001) http://razor.bindview.com/publish/papers/tcpseq.html

[3] n/a, *Secure Programming for Linux and Unix HOWTO*, (2002)

[4] M. Santha and U.V. Vazirani, *Generating Quasi-Random Sequences from Slightly Random Sources*, Journal of Computer and System Sciences, v33, (1986), pp 75-87

[5] J. Kelsey, *Cryptanalytic Attacks on Pseudorandom Number Generators*, (2001)

[6] n/a, *Cryptography in OpenBSD*, (2002)