УДК 004.056

РЕКОМЕНДАЦИИ ПО ПРИМЕНЕНИЮ СЛУЧАЙНЫХ ЧИСЕЛ КРИПТОГРАФИЧЕСКОГО КАЧЕСТВА ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Червоная Ольга Владимировна
ГВУЗ «Национальный горный университет», http://bit.nmu.org.ua, chervonaolga@gmail.com

Рассмотрены существующие механизмы генерации случайных и псевдослучайных чисел, их достоинства и недостатки, разработаны рекомендации по работе с ними.

Ключевые слова - генератор случайных чисел; генератор псевдослучайных чисел; разработка программного обеспечения; рекомендации.

ВСТУПЛЕНИЕ

При разработке программного обеспечения случайные числа применяются для решения различных задач, однако наиболее часто используются для генерирования криптографических ключей и идентификаторов сеансов. Случайные числа – это одна из основ криптографии, поэтому методы их использования встречаются в самых разных языках программирования. И ошибки при их использовании тоже присущи любому языку.

Существуют три разных механизма:

- некриптографические генераторы псевдослучайных чисел (некриптографические PRNG);
- генераторы псевдослучайных чисел криптографического качества (CRNG);
- генераторы «истинно» случайных чисел (TRNG), известные также под названием энтропийные генераторы.

НЕКРИПТОГРАФИЧЕСКИЕ PRNG

До появления сети Интернет случайные числа не использовались в критических с точки зрения безопасности приложениях. Они находили применение лишь в статистическом моделировании. Для испытаний методом Монте Карло [1] нужны были числа, прошедшие все статистические тесты на случайность. Такие испытания по самому замыслу должны были быть повторяемыми. Поэтому API строился так, чтобы, получив на входе одно число, можно было порождать из него длинную серию чисел, выглядящих случайными. В подобных генераторах использовалась довольно простая математическая формула, порождающая последовательность элементов из начального значения (затравки).

Таблица 1. Некриптографические генераторы псевдослучайных чисел в популярных языках [2]

Язык	API
C и C++	rand(), random(), seed(), initstate(), setstate(), drand48(), erand48(), jrand48(), lrand48(), nrand48(), nrand48(), lcong48() и seed48()
Windows	UuidCreateSequential
C# и VB.NET	Класс Random
Java	Весь пакет java.util.Random
JavaScript	Math.Random()
VBScript	Rnd
Python	Целиком модули random и whrandom
Perl	rand() и srand()
PHP	rand(), srand(), mt_rand() и mt_srand()

Когда стали задумываться о безопасности, требования к случайным числам ужесточились. Нужно было, чтобы они не только успешно проходили статистические тесты, но еще чтобы злоумышленник не мог предсказать, какое число будет следующим, даже если видел несколько предыдущих.

Задача ставится следующим образом: если противник не знает затравку, то не может угадать число, которое еще не видел, сколько бы чисел ни наблюдал перед этим.

ГЕНЕРАТОРЫ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ КРИПТОГРАФИЧЕСКОГО КАЧЕСТВА CRNG

Простейшие криптографические генераторы псевдослучайных чисел (CRNG) работают примерно так же, как традиционные генераторы: вырабатывают из затравки длинную последовательность чисел. Если затравки одинаковы, то и последовательность будет той же самой. Проблема в том, что злоумышленник не должен знать затравку. Чтобы CRNG-генератор был безопасным, затравка должна быть неугадываемой.

Но, говоря о CRNG-генераторах, также необходимо включить в рассмотрение инфраструктуру изменения затравки, а не только сам алгоритм генерации псевдослучайных чисел. Поэтому современные CRNG-генераторы не так полезны, как шифры, ибо они стремятся как можно чаще подмешивать новые истинно случайные данные (энтропию).

Следует также отметить, что стойкость криптографического генератора не может быть выше стойкости ключа. Поэтому считается правильным выделять истинную энтропию, устраняя статистически определимые закономерности. Один из способов добиться этого – подать исходное значение на вход CRNG-генератора в качестве затравки и воспользоваться выработанным результатом.

ЭНТРОПИЙНЫЕ ГЕНЕРАТОРЫ TRNG

TRNG являются наиболее безопасными генераторами, однако на практике их трудно найти, ведь компьютеры по своей природе детерминированы. И такие процессы как измерение

времени между нажатиями клавиш или перемещениями мыши, состояния ядра или процессов не являются достаточно случайными и не могут быть использованы напрямую [3]. Поэтому считается правильным «выделять» истинную энтропию, устраняя статистически определимые закономерности. Один из способов добиться этого – подать исходное значение на вход CRNG-генератора в качестве затравки и воспользоваться выработанным результатом.

ВЫВОДЫ

1. При разработке программного обеспечения необходимо проанализировать код и выявить места, в которых следовало бы пользоваться случайными числами, но это не делается;
2. Выявить места, где применяются PRNG-генераторы;
3. Убедиться, что для всех применяемых CRNG-генераторов используется затравка надлежащего качества.

Рекомендации:

1. По возможности пользоваться криптографическим генератором псевдослучайных чисел (CRNG).
2. Убедиться, что затравка любого криптографического генератора содержит по меньшей мере 64, а лучше 128 битов энтропии.
3. Не пользоваться некриптографическими генераторами псевдослучайных чисел (некриптографические PRNG).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Статья «Моделируя жизнь», автор Андрей Тепляков, опубликована в №7 2001 Hard'n'Soft
2. Ховард М., Лебланк Д., Виега Д. Х68 "19 смертных грехов, угрожающих безопасности программ. Как не допустить типичных ошибок". – М.: ДМК Пресс, 2006. – 288 с.: ил. ISBN 5$9706$0027$X
3. RFC 1750: рекомендации по обеспечению случайности в целях безопасности: www.ietf.org/rfc/rfc1750.txt