TESTY A ZDROJE NEURČITOSTI V POČÍTAČI

Dokonalá náhoda

Kvalitní zdroj náhodných čísel je v počítačové bezpečnosti stejně cenný jako přístupová hesla, klíče nebo ostatní bezpečnostní prvky systému. V lednovém čísle Chipu jsme vás seznámili s novým objevem v měření entropie zdroje náhodných čísel. Nyní si ukážeme, jak se takové zdroje dají najít v každém počítači a jak je správně využít.

Nejprve si uvědomme, co od kvalitního zdroje náhodných čísel *(RNG, random number generator)* vlastně očekáváme. Možná si vzpomenete, že jsme v číslech 4/98 až 6/98 o náhodných generátorech psali. Tehdy však šlo o tzv. pseudonáhodné generátory *(PRNG, pseudorandom number generators)*, které pracovaly zcela deterministickým matematickým postupem. Jsou výborné pro nejrůznější účely a simulační metody, kde je potřeba velmi rychle generovat data, která mají – ze statistického hlediska – všechny příznaky náhodných dat. V těchto případech nevadí, že kdyby se na tato data blíže podívali hackeři nebo profesionální luštitelé, mohli by ze znalosti vzorce pro jejich tvorbu nebo z předchozích dat určit i následující produkci generátoru.

Od **kvalitních** náhodných čísel ovšem chceme mnohem více, protože mohou mít bezpečnostní význam. Náhodná čísla se totiž přímo používají jako šifrovací klíče prakticky ve všech bezpečnostních protokolech (např. SSL) nebo pro tvorbu velmi významných dlouhodobých klíčů, klíčů pro digitální podpis apod. Při takovémto použití je tedy nezbytně nutné kvalitu náhodného zdroje **zaručit**.

Požadavky na RNG

Budeme proto požadovat, aby znalost jakéhokoliv množství předchozích generovaných dat nedávala analytikovi (útočníkovi) žádnou šanci pro predikci následujících bitů. K tomu navíc předpokládáme, že dotyčný útočník má k dispozici:

- potřebný výpočetní výkon
- plnou znalost procesu generování
- přístup ke stejným zdrojům, z nichž je konstruován generátor (součástky, software)
- možnost testovat a používat generátor

Pokud i za těchto předpokladů náš generátor při útocích obstojí, můžeme ho považovat za kvalitní. Připomeňme, že v kontextu použití RNG je potřeba vyřešit i obranu proti útokům systémovým, fyzickým apod., kterými se ovšem v tomto článku nebudeme zabývat. Zde nám půjde jen o „kvalitu náhodnosti“ generovaných čísel.

True-random generátory

Čísla, o nichž je řeč, se nazývají ***skutečně náhodná čísla*** a produkují je tzv. ***true-random generátory***. Není jich mnoho, ani v přírodě, ani v osobním počítači. Mezi nejkvalitnější patří generátory na bázi radioaktivního rozpadu a na bázi napěťově-proudových změn způsobených tepelným šumem a kvantovými jevy v polovodičových strukturách. Radioaktivní materiál se ke generování náhodných bitů skutečně používal (i u nás). Pokud počet zachycených částic vznikajících jeho rozpadem za určitou časovou jednotku byl lichý, zařízení vygenerovalo jedničku, jinak nulu. Trochu drahé – ale jiné nezpochybnitelné generování náhodných bitů před několika desítkami let neexistovalo. Takové generátory proto používaly i všechny světové velmoci při tvorbě jednorázového hesla *(one-time pad)* pro šifrování diplomatických spojů.

Z tohoto příkladu je vidět, jaké asi požadavky na true-random generátor klademe a jaký typ generátoru by pro nás byl ideální. Málokdo by nám totiž oponoval, že je schopen rekonstruovat nebo predikovat natolik složitý přírodní proces, jako je radioaktivní rozpad, a to ve formě, v jaké je využit. Praxe opravdu potvrzuje, že tento generátor respektuje každý, zatímco u jakéhokoliv jiného principu se vždy najde nějaký kritik.

Současný stav a trendy

Radioaktivní materiál později nahradily polovodičové prvky, kde se entropie odvozuje od nedeterministických změn v napětí a proudu na vybraných přechodech. Pochopitelně si to vyžádalo trochu „víc vědy“, neboť náhodnost změn bylo potřeba prokázat, zjistit, za jakých podmínek nastávají, a odpovídající součástky kalibrovat. Nicméně se to podařilo zvládnout a dnes už se jedná o standardizovaný postup. Nárůst požadavků na přítomnost kvalitního generátoru v PC poté vedl k používání externích generátorů ve formě (poměrně drahých) přídavných desek. V současné době se začínají RNG realizovat v čipech, které se integrují přímo na základní desky už při výrobě nových osobních počítačů. Výrobce k tomu vedou nové bezpečnostní požadavky kryptografie, která se používá k ochraně dat v lokálních sítích, na internetu, v elektronickému obchodu, pro šifrování a podpis elektronické pošty v a dalších aplikacích. Prvním

> Tvorbě generátorů **náhodných čísel** je nutné věnovat stejnou **pozornost** jako jiným **bezpečnostním** prvkům.

příkladem tohoto postupu může být čip od Intelu (viz infotipy).

Zdroje náhody v počítači

Pokud nemáme k dispozici speciální čip, desku nebo externí generátor, nezbude nám než si pomoci sami. Uvedené metody samozřejmě nejsou ideální, my se však budeme snažit z našeho počítače vyždímat maximum. V připojené tabulce uvádíme příklady možných zdrojů – jde nám přitom o pokud možno fyzikální zdroje, kde bude nepredikovatelnost chování velmi dobrá a entropie příslušného zdroje co největší (a pokud možno měřitelná).

Zdroj náhody	Poznámka
Speciální periferie	
Šumové diody, speciální přídavné čipy nebo desky, tepelné rezistory	
Radioaktivní zářiče	
Běžné periferie	
Mikrofon	Zdrojem neurčitosti může být výstup z jednoho nebo více mikrofonů.
Videokamera, zvuková karta	Zdrojem neurčitosti může být snímaný signál a někdy i vlastní šum zpracovávajícího zařízení.
Pevné disky	Odchylky dob při čtení dat z disku (použitelné dříve, disková cache tuto možnost odbourala).
Systémové zdroje	
Statistika síťového provozu Statistika procesů operačního systému Statistika vstupních a výstupních operací	Systémová statistika, která není (nebo je velmi špatně) viditelná nebo dosažitelná analytikem síťového provozu.
Systémové datum a čas	
Činnost uživatele	
Psaní na klávesnici	Trvání stisku kláves, doba mezi stisky kláves, hodnoty kláves
Pohyb myši	Změna polohy kurzoru v čase, dobrovolný nebo vynucený pohyb

Příklady zdrojů entropie v počítači

Jak konstruovat generátor

Existuje více cest, jak zkonstruovat RNG. Ukážeme si zde standardní postup, který se skládá ze čtyř kroků:

- sběr entropie
- komprese
- nastavení kryptografického generátoru
- expanze

Jak už jistě tušíte ze třetího kroku, podstata spočívá v tom, že vlastní generování náhodných čísel přesouváme na kryptografické generátory. Umí totiž přesně to, co chceme, tj. generovat nepredikovatelnou posloupnost čísel – viz vlastnosti, které jsme si definovali v odstavci „požadavky na RNG“.

Kde je ale ta neurčitost, o kterou se snažíme? Neutíkáme od problému, když chceme neurčitost a používáme deterministické postupy? Nikoliv – vtip je totiž v tom, že v prvním kroku získanou entropii použijeme k nastavení generátoru do neurčitého a neznámého počátečního stavu. Kryptografický generátor je přitom konstruován tak, že do svého výstupu v každém kroku přenáší entropii svého počátečního stavu. Díky tomu, že využívá jednosměrné funkce nebo blokové šifry, není možné obrátit jeho chod zpět, a stačí jej tedy nastavit do neznámého, neurčitého a nedeterministického počátečního stavu.

Výhodné také je, že počáteční nastavení vyžaduje řádově pouze desítky až stovky náhodných bitů, zatímco následující megabajty náhodných čísel generuje kryptografický generátor sám, a to nesrovnatelně rychleji, než kdybychom získávali entropie z vlastního počítače přímo. Navíc můžeme generátor kdykoliv opět restartovat (pokud z dostupných zdrojů opět nasbíráme dost entropie – po vygenerování určitého objemu dat, po stanovené době apod.). Pojďme se už ale podívat na jednotlivé kroky podrobněji.

Sběr neurčitosti a její komprese

Cílem těchto kroků je získat z počítače K bitů neurčitosti. Řádově jde vždy o několik desítek až stovek bitů, pro jednoduchost zde uvažujme K = 160. Přitom se osvědčuje dbát několika standardních rad. Za prvé je vhodné neurčitost získávat současně z několika zdrojů (třeba ze všech uvedených v tabulce). Za druhé, získaná data prostě jen řadíme za sebou bez ladu a skladu a nemusíme je očišťovat od „deterministického balastu“. Staráme se jen o to, aby v jejich souhrnu bylo obsaženo požadované množství entropie (z bezpečnostních důvodů sbíráme dvoj- až trojnásobek). A za třetí, jakmile máme k dispozici dostatečné množství dat (záleží jen na jejich entropii, nikoli na velikosti), aplikujeme na ně hašovací funkci. Pro jednoduchost zde budeme uvažovat SHA-1, jejíž výstup je 160 bitů. Výsledkem hašování je řetězec 160 náhodných, nepredikovatelných bitů, které představují obraz vstupních dat obsahujících tuto entropii – tím je vyřešen problém „destilace“ entropie z nich.

Hašovací funkce zde vlastně vykonává dvě úlohy – jednak přenáší entropii, jednak komprimuje vstupní data na výstup. Mimo jiné se zde využívá skutečnosti, že hašovací funkce převádějí silně korelované vstupy (lišící se třeba o jeden bit nebo přehozením dvou bitů) na výstupy, v nichž předchozí algebraické vztahy a závislosti jsou zcela potřeny – nejsou prokazatelné ani výpočetně zjistitelné a vypadají zcela náhodně a nezávisle. Hašovací funkce zajistí, že v nasbíraných datech záleží jak na pořadí, tak na hodnotě každého bitu zdrojových dat.

Náhodná **čísla** jsou přímo používána jako **šifrovací** klíče.

Hašovací funkce je ale vzhledem k těmto vlastnostem také schopna vnějškově kamuflovat i špatný zdroj neurčitosti. Například posloupnost *SHA-1(systémový čas)*, snímaná jakkoli často, bude pro většinu hackerů nepřekročitelným zdrojem náhody, pokud nebudou znát tento předpis (touto cestou však jít nechceme). Připomeňme ještě, že vlastnostmi hašovacích funkcí jsme se blíže zabývali v Chipu 3/99 a konkrétně SHA-1 v Chipu 4/99. Vraťme se však ke sběru entropie.

infotipy

Chip od Intelu realizující RNG:
- *support.intel.co.jp/design/security/rng/rng.htm*

Jeho posouzení:
- *ww.cryptography.com*

Doporučení pro tvorbu RNG:
Dokument „Randomness Recommendations for Security (Eastlake, Crocker, Schiller)“:
- *ds.internic.net/frc/rfc1750.txt*

Dokument FIPS PUB 140-1, Security Requirements for Cryptographic Modules, NIST, 1994:
- *www.itl.nist.gov/div897/pubs/fip140-1.htm.*

Měření neurčitosti

Každý zdroj entropie z počítače, který využijeme, musíme předem dokonale prověřit. K měření entropie je možné použít Maurerův-Coronův test (viz Chip 1/00). Pokud si nevíme rady, existuje také velmi hrubý postup, jak ji odhadnout. Uvádíme ho zde jen proto, že se jedná o zavedený úzus, ale jako globálně bezpečný ho v žádném

případě nedoporučujeme. Je velmi jednoduchý. Data komprimujeme nejlepším komprimačním programem, který máme k dispozici, s cílem dosáhnout co největší komprese. Počet bitů zkomprimovaného souboru vydělíme dvěma a poté ještě bezpečnostní konstantou (2, 10, ... fantazii se meze nekladou). Výsledkem je přibližný počet bitů entropie.

Tento postup ale nemusí vždy fungovat. Vezměme například jako zdroj náhody pohyb myši. Vyzveme-li uživatele, aby náhodně pohyboval myší, a ten bude mít snahu to skutečně ve vlastním zájmu dělat, pak můžeme očekávat, že požadovaných 160 bitů entropie dosáhneme během několika málo sekund. Pokud však toto úsilí bude uživatel sabotovat, zcela jistě nám tato doba nestačí. Abychom ho přelstili, museli bychom požadovat, aby myší například obkresloval zadaný obrazec, a sofistikovaně kontrolovat, že to skutečně dělá, abychom mohli náhodné odchylky v jeho tazích vyhodnocovat jako zdroj entropie.

Podobné je to i s často využívaným měřením času při psaní na klávesnici. Běžně se sleduje doba mezi stisky kláves, trvání jejich stisku a obsah. Také toto se však dá sabotovat – a vůbec nejlepší je proto lidský činitel z těchto metod vyloučit. Z bezpečnostního hlediska bychom rovněž měli zvažovat situace, kdy a jak může být příslušný zdroj entropie ovlivněn případným útočníkem. Ať budeme ale využívat cokoli, musíme být přesvědčeni, že požadované množství entropie skutečně nasbíráme.

Standardní postupy expanze

Předpokládejme tedy, že jsme nějakou vhodnou metodou obdrželi 160bitovou hodnotu, která je zaručeně náhodná. Budeme ji v dalším využívat buď jako klíč (KEY) pro blokové šifry, nebo jako počáteční nastavení (SEED) pro hašovací funkce (h). Šifrování dat D klíčem KEY a vhodnou blokovou šifrou označme $E_{KEY}(D)$. Standardní postupy expanze náhodných dat lze jednoduše popsat pseudokódy podle obrázku.

Vstupem je hodnota SEED, výstupem je posloupnost r(1), r(2), ..., r(N). Protože tyto standardní postupy většinou jsou (nebo alespoň mohou být) známy útočníkovi, kvalita jejich výstupu závisí na kvalitě prvku, který útočník nezná, tj. hodnoty SEED. Tuto hodnotu musíme ochránit, vše ostatní za nás udělá kryptografie.

Od kvalitních **náhodných** čísel požadujeme, aby znalost jakéhokoliv množství předchozích **dat** nedávala **útočníkovi** žádnou šanci pro predikci následujících dat.

Varianty tvorby RNG

Uveďme si zde alespoň některé varianty základních postupů, s nimiž lze konstruovat různé typy RNG:

- po vygenerování určitého množství dat se automaticky přechází na nové nastavení, tj. na vytvoření nové hodnoty SEED
- během procesu generování probíhá kontinuálně sběr entropie pro nové nastavení
- výstupy z několika zdrojů entropie se mohou vzájemně doplňovat, kombinovat a obnovovat v různých časových intervalech
- při vytváření nové hodnoty SEED z různých zdrojů entropie je možné k nim přidat i starou hodnotu SEED
- při realizaci předchozího kroku se nepoužije stará hodnota SEED přímo, ale prostřednictvím h(SEED), z bezpečnostních důvodů se totiž SEED v počítači neuchovává přímo
- v kroku expanze je možné použít společně s řetězcem KEY i určitou tajnou hodnotu SECRET, jejíž ochraně je v daném systému věnována zvýšená pozornost

S využitím counter modu blokové šifry

E je bloková šifra s délkou bloku B
const je libovolná konstanta
counter = E_{SEED}(const)
for n = 1 to N
{
    r(n) = E_{SEED}(counter)
    counter = counter +1
}
Pozn.: N je nutné volit menší než $2^{B/2}$ a poté generátor restartovat.

S využitím counter modu a hašovací funkce

counter = SEED
|| je zřetězení dat
h je hašovací funkce

for n = 1 to N
{
    r(n) = h(SEED || counter)
    counter = counter +1
}

S využitím CBC modu blokové šifry

E je bloková šifra
XOR je operace exclusive or
d(0), d(1),... je proud pseudonáhodných dat generovaných
např. kongruentním generátorem (PRNG)

r(0) = d(0)
for n = 1 to N
{
    r(n) = E_{SEED}(r(n-1) XOR d(n))
}

Standardní postupy expanze pro generátory náhodných čísel

Závěr

Generátory náhodných čísel se často používají pro senzitivní bezpečnostní účely. Proto je nezbytné věnovat jejich tvorbě stejnou pozornost jako jiným bezpečnostním prvkům. Zde jsme uvedli standardní postupy tvorby takových generátorů, které využívají kryptografické postupy k tomu, aby zajistily nepredikvatelnost svých výstupů a současně obsahovaly entropii originálního zdroje.

Vlastimil Klíma / v.klima@decros.cz