

Tilfeldigheter og determinisme.

Kapittel 10. Building Secure Software: How to Avoid Security Problems the Right Way

Brage Breivik

brage@ii.uib.no

5. oktober 2005

INF 329 - Utvalgte emner i programutviklingsteknologi:

Utvikling av sikre applikasjoner





Oversikt

- ◆ Pseudotilfeldige tallgeneratorer
 - Egenskaper
 - Eksempler
 - Eksempler på angrep
- ◆ Entropi
 - Hardware vs. Software løsninger
 - Hvordan beregne hvor mye entropi en har
- ◆ Praktiske løsninger



Tilfeldige tall

- ◆ Tilfeldige tall er viktige innen sikkerhet, bl.a til generering av nøkler, stokking av kort og mye annet
- ◆ Siden datamaskiner er deterministiske, er de veldig dårlige til å oppføre seg tilfeldig (selv om feilmeldingene de gir ofte kan se tilfeldige ut)
- ◆ Eneste gode kilde til tilfeldige tall er å måle fysiske fenomen som radioaktivitet



Pseudotilfeldige tallgeneratorer

- ◆ Siden datamaskiner er dårlige til å lage ”ekte” tilfeldige til, bruker man pseudotilfeldige tallgeneratorer(PRNG)
- ◆ PRNG'er trenger input, og for en gitt input gir den alltid samme output
- ◆ Dette er viktig for å kunne gjenskape resultater, f.eks til debugging



Frø (seeds)

- ◆ For å få gode output fra en PRNG, må man gi gode input, eller frø
- ◆ Entropi er et mål på hvor mye "ekte" tilfeldighet det er i en mengde data.
- ◆ Brukes for å si noe om hvor god input man gir til en PRNG
- ◆ Viktig å holde frøene hemmelig



Frø, forts.

- ◆ Dersom frøet vårt inneholder n bits med entropi, er den best måten å angripe det på ”brute-force”, som vil ta 2^n operasjoner, dersom PRNG'en vår er god
- ◆ En god PRNG skal altså være slik at gitt nok entropi, skal den produsere tall som vil være vanskelige å gjette, selv for en angriper som kjenner alle detaljer i algoritmen



Eksempler på PRNG

- ◆ Ikke kryptografisk
 - Linear congruential generator
- ◆ Kryptografiske
 - Blum-Blum-Shub
 - Tiny
 - Kryptografiske hash-funksjoner

Linear congruential generator



- ◆ Er den algoritmen som blir brukt i kall til C's Random()

- ◆ Har formen:

$$X_{n+1} = (aX_n + b) \text{ mod } c$$

der a , b og c er vanligvis primtall

- ◆ Er lett å angripe kryptografisk, og må aldri brukes når det er nødvendig med sikkerhet



Blum-Blum-Shub

- ◆ Blum-Blum-Shub er basert på faktorisering av store tall
- ◆ Er fryktelig treg i praksis
- ◆ Viktig likevel, siden den bygger på et enkelt matematisk prinsipp, og ikke slik som andre PRNG'er, på hash-funksjon og block ciphers



Tiny

- ◆ Virker ved å kryptere en teller ved hjelp av AES
- ◆ Har innebygd behandling av entropi
- ◆ Blir ansett for å være sikker mot alle kjente kryptografiske angrep



Kryptografiske hash-funksjoner

- ◆ Tar en hemmelig teller som input
- ◆ Bruker en kryptografisk hash-funksjon som SHA-1 eller MD5
- ◆ Avhengig av ikkereverserbarheten til hash-funksjonen
- ◆ Blir brukt blant annet i Java



Angrep mot PRNG'er

- ◆ Kryptoanalytiske angrep
 - Et angrep som kan skille generator output fra virkelig tilfeldige tall
- ◆ Angrep mot den interne statusen til PRNG'en
 - PRNG'en må beskytte en informasjon om sin interne status så godt som mulig
 - Dersom et slikt angrep er vellykket, vil en angriper kunne forutsi alle fremtidige tall, i hvert fall til ny entropi blir tilført systemet



Hvordan fuske i online-poker

- ◆ I 1999 oppdaget SSG ved Cigital en alvorlig feil i implementasjonen av Texas Hold'em Poker
- ◆ Denne feilen tillot en uærlig spiller å beregne hvordan alle kortene lå i stokken i sanntid
- ◆ Kan dermed vite på forhånd om han vil vinne eller tape denne hånden, og svindle seg til store penger



Hvordan fuske i online-poker

- ◆ Feilen lå i stokkealgoritmen
- ◆ Denne ble ironisk nok publisert i en FAQ for å vise hvor trygt og rettferdig systemet var
- ◆ I koden ble et kall til `randomize()` gjort
- ◆ Dette ble kalt med antall millisekunder siden midnatt som frø



Hvordan fuske i online-poker

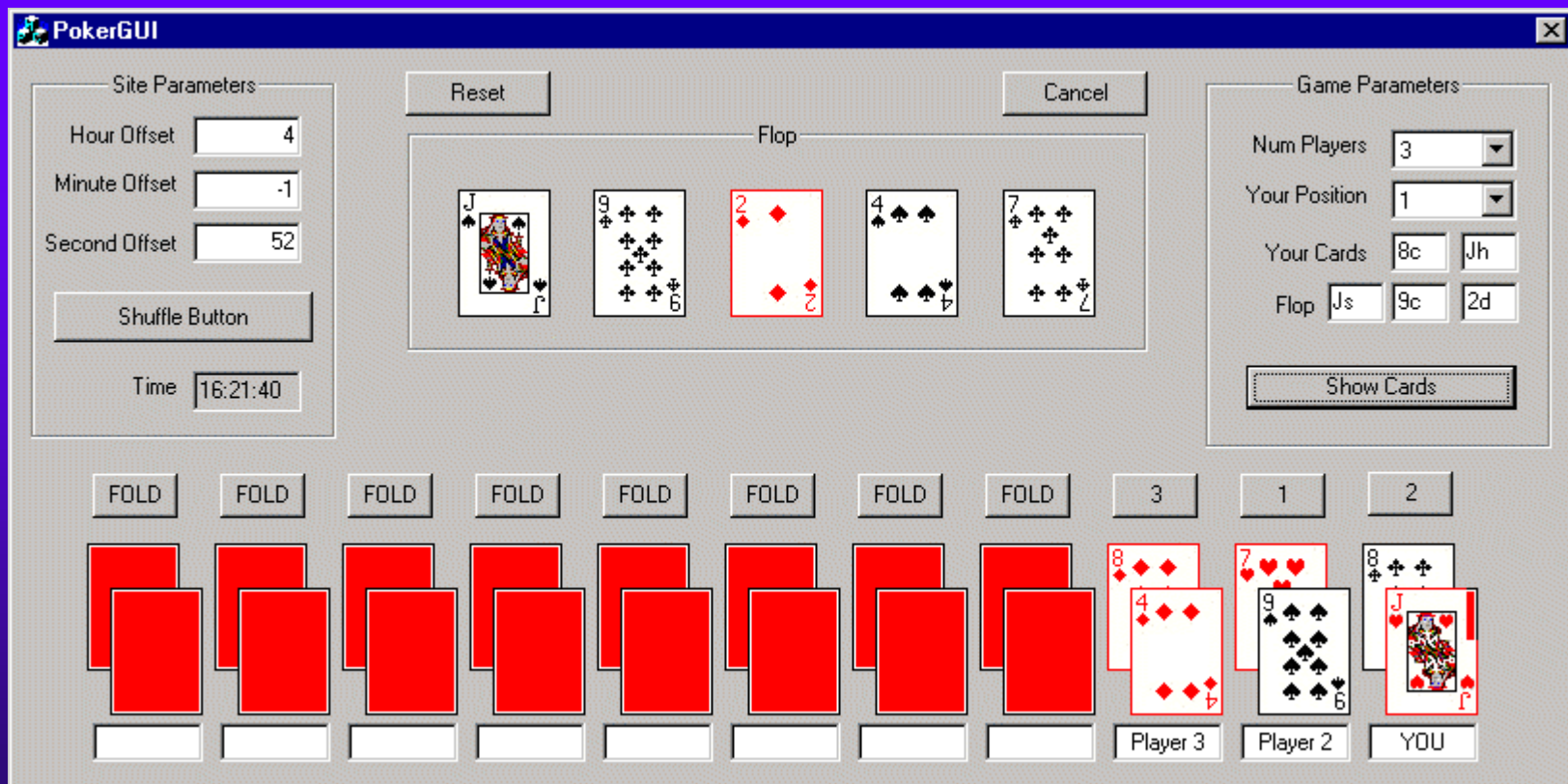
- ◆ I en virkelig kortstokk er det $52!$ (ca 2^{226}) mulige måter å stokke den på
- ◆ Her brukte de et 32-bits frø, noe som reduserte antall muligheter til 4 milliarder
- ◆ Siden det bare er 86,400,000 ms i et døgn, blir antall mulighet redusert til dette
- ◆ Ved å synkronisere sin klokke med serverklokken, ble antall muligheter redusert til ca 200,000, et antall en pc kan søke gjennom på under et sekund



Hvordan fuske i online-poker



Hvordan fuske i online-poker





Hvordan fuske i online-poker

A World of Action!

Help
Suggestions

Fold in Turn

Jonnyboy

brute

RickC

mamajoe

Fold Show

Options
Sit Out
Leave

584330

mamajoe: Hey guys, Big B is in!

Dealer Text



Entropi

Innsamling og beregning



Entropi

- ◆ En god PRNG gir ingen sikkerhet i seg selv, den er avhengig av frø med mye entropi
- ◆ Mennesker er ingen god kilde for entropi
 - Bruker nettverksadresser, vertsnavn, folks navn og programmererens mors pikenavn
- ◆ En vanlig teknikk er å bruke systemklokken
 - Bare 32-bits, noe som er for lite, og disse inneholder heller ikke mye egentlig entropi
- ◆ Vi trenger altså bedre metoder



Maskinvareløsninger

- ◆ Å bruke dedikert maskinvare for å produsere entropi pleier å gi mer og bedre kvalitets entropi enn software-løsninger
- ◆ Ofte er det ikke mulig å bruke, f.eks fordi et system skal distribueres rundt hele verden
- ◆ Koster også penger
- ◆ Det vil derfor også være behov for software-løsninger



Maskinvareløsninger

- ◆ Bruker vanligvis en naturlig, stokastisk prosess
 - Termisk støy fra en halvleder diode
 - En geigerteller som måler radioaktiv nedbrytning



Måling av radioaktiv nedbryting

- ◆ Umulig å forutsi om den neste nedbrytingen vil bruke mer eller mindre tid enn den forrige
- ◆ En slik måling gir en bit med entropi
- ◆ En kan eventuelt måle tiden mellom hver nedbryting, og bruke de individuelle sifrene som individuelle tilfeldige tall



Problemer med denne metoden

- ◆ Denne metoden kan være eller ikke være god nok
 - Hva om hundredelssifferet er nesten alltid 0, men sjelden 1?
 - Hva om klokken ikke er nøyaktig nok?
- ◆ Dette er problemer som er vanlige ved slike typer maskinvareløsninger, og som må løses for at et slikt system skal virke
- ◆ Det finnes teknikker for å fjerne sammenhenger mellom bitsene i entropi fra maskinvare



ComScire QNG

- ◆ Den vanligste kommersielle innretning for å generere tilfeldige tall det året boken kom ut
- ◆ Produserer 20000 bits i sekundet, noe som er mer enn nok fr de fleste systemer
- ◆ Gjør statistiske tester på tallene mens de blir generert



Lavarand

- ◆ Lavarand er en noe morsommere måte å generere tilfeldige tall på
- ◆ Den bygger på det innbygde kaoset i lavalamper
- ◆ Et digitalkamera tar et bilde av seks lavalamper og bruker dette for å lage et frø til Blum-Blum-Shub algoritmen
- ◆ Bør ikke brukes til noe sikkerhetskritisk siden de genererte tallene blir offentliggjort på websidene deres
- ◆ Det virker som om prosjektet nå er nedlagt, hjemmesiden er i hvert fall forsvunnet



Innebygd RNG i Intel chipsets

- ◆ I 1999 annonserte Intel at de ville innebygde RNG'er på sine chipset
- ◆ Alle Pentium III med 8xx chipsets skulle få dette
- ◆ Dersom dette hadde blitt standard ville det blitt et betydelig fremskritt innen sikkerhet
- ◆ Det ser ut til at Intel siden har gått vekk fra dette



Software løsninger

- ◆ Fordi en datamaskin er totalt deterministisk, vil rene software-løsninger alltid kunne kompromitteres av en angriper med tilstrekkelige ressurser
- ◆ Dette er mer et teoretisk enn et praktisk problem
- ◆ De fleste applikasjoner baserer seg på slike løsninger, fordi maskinvareløsninger ofte er veldig tungvinte



Software-løsninger

- ◆ Dersom vi kan gjøre forutsetningen at en maskin ikke totalt kompromittert, kan vi likevel få entropi fra software
- ◆ Dette gjør vi ved å måle input som burde være uforutsigbare, som f.eks tastetrykk
- ◆ De få inputene en datamaskin får, kan være nok til å få den til å oppføre seg på en måte som er vanskelig å simulere



Måling av mus- og tastaturinput

- ◆ En vanlig kilde til entropi er å måle mus og tastaturbruk fra bruker
- ◆ Best dersom de blir utført på initiativ fra en bruker, for da får de et tilfeldig tidspunkt
- ◆ Et problem er at kilder for tilfeldighet burde være hemmelige, noe spesielt musepeking ikke er på de fleste systemer
- ◆ Et annet problem er dersom verken mus eller tastatur blir brukt i det aktuelle tidsrommet
- ◆ Vi bør ikke beregne mer enn 2 bits entropi pr hendelse vi måler



Måling av systemets responstid

- ◆ Responstiden til et system avhenger av mange faktorer som hva brukeren gjør, hvilke andre prosesser som kjører og prioriteringer av disse
- ◆ Vi kan derfor hente ut noe entropi ved å måle denne
- ◆ En utfordring da blir å måle hvor mye entropi vi faktisk finner, noe som er en veldig komplisert oppgave
- ◆ De fleste vanlige software-bibliotek prøver ikke en gang å beregne hvor mye entropi de samler, de samler bare mye og håper at dette skal være nok



Beregning av entropi

- ◆ Forfatterne har implementert en C-kode som måler hvor lang tid et systemkall tar, og sjekker forskjellene i tid
- ◆ De kjører så denne koden en million ganger for å få et statistisk grunnlag for å beregne hvor mye entropi dette gir
- ◆ Det viser at for de 6 minst signifikante bitsene har vi en halv bit entropi for hver fordi hver av dem blir 0 eller 1 minst $\frac{1}{4}$ av gangene
- ◆ Dette burde tilsi at vi skulle få 3 bits entropi for hver kjøring, men forfatterne sier at vi bør bare beregne ca $\frac{1}{3}$ (et nokså tilfeldig tall) av dette for å være på den sikre siden



Andre måter å samle entropi

- ◆ Tester viser at en bedre måte å samle entropi på, er å ta tiden på hvor lenge det tar å starte en tråd som gjør ingenting
- ◆ Dette gir opptil 6 bits pr kall, men tredjedelsregelen kutter dette ned til 2
- ◆ Selv om vi er redd for et angrep på en av entropikildene våre, er det bedre å bruke den enn å la være, men bruk den sammen med andre kilder
- ◆ Det er en fordel å ikke gjøre all entropiinnnsamling på en gang, men heller fordele det ut over tid, omtrent som garbage-collection



Truerand

- ◆ Truerand en en teknikk for innsamling av entropi skrevet av Don Mitchell og Matt Blaze
- ◆ Den er UNIX-spesifikk
- ◆ Måler avviket i tid mellom system klokken og genereringen av interrupts på prosessoren
- ◆ Empiriske målinger viser at dette er en god kilde til entropi



Hvordan lese "hemmelige" Netscape-meldinger

- ◆ I 1996 ble et funnet en alvorlig feil i en tidlig implementasjon av SSL
- ◆ SSL krypterer ved hjelp av en hemmelig engangsnøkkel
- ◆ Dersom denne er forutsigbar, er hele systemet åpent for angrep
- ◆ Det er altså viktig at denne kommer fra en uforutsigbar kilde



Hvordan lese "hemmelige" Netscape-meldinger

- ◆ Netscapes problem var at de brukte et dårlig frø til sin PRNG
- ◆ Dersom man kjente tidspunktet, prosess-ID'en og foreldreprosess-ID'en kunne man bestemme frøet
- ◆ En angriper som bruker samme maskin som offeret kan lett finne ut prosess-ID'ene
- ◆ Sniffere kan plukke opp det nøyaktige tidspunktet en pakke blir sendt, noe som gir et godt utgangspunkt for å gjette tidspunktet på systemet som kjører Netscape
- ◆ Dette gjorde at ekte nøkler kunne knekkes på under 30 sekunder



Behandling av entropi

- ◆ En entropibehandler har ansvaret for ta inn vilkårlige mengder data fra entropikilder, sammen med et estimat for hvor mye entropi det inneholder, og gi ut en sekvens av tilfeldige tall
- ◆ Skiller seg fra en PRNG på den måten at den lager bits som tilnærmet rent entropiske
- ◆ Er treg, og bør derfor brukes til å lage frø for en PRNG



Tilfeldigheter i praksis

- ◆ For ikke lenge siden måtte man snekre sammen sin egen implementasjon dersom man trengte sikre tilfeldige tall i sin applikasjon
- ◆ Nå begynner kryptografisk sterke tilfeldige tallgeneratorer å dukke opp i operativsystemer og programmeringsspråk



Tiny

- ◆ Som nevnt tidligere er Tiny en PRNG og en entropiinnnsamlingsalgoritme
- ◆ PRNG'en er et simpelt primitiv som bygger på sikkerheten til AES, og som periodisk tilfører nye frø for å unngå at en angriper skal kunne samle informasjon om den interne tilstanden



Tiny

- ◆ Entropi innsamlingen har en mer kompleks infrastruktur
- ◆ For å beskytte mot mulige sårbarheter i kildene bruker den mange kilder som sammen bidrar med et minimum hver
- ◆ Bruker en "slow pool", og bytter ikke frø ofte, slik at den samler mye entropi for hver gang
- ◆ Ingen garanti for at den alltid produserer en bit entropi pr bit output



Tilfeldige tall for Windows

- ◆ CryptGenRandom
- ◆ Usikkert hva denne egentlig gjør, men dokumentasjonen påstår den er kryptografisk sikker
- ◆ Vi vet ikke hvordan entropy blir samlet inn eller brukt i systemet
- ◆ Anbefaler derfor å bruke andre mekanismer dersom mulig



Tilfeldige tall for Linux

- ◆ Versjoner av Linux fra 1995 eller seinere har innebygd støtte for kryptografisk sikre tilfeldige tall
- ◆ Kallet `/dev/random` gir tilfeldige tall basert på innhentet entropi, dersom nok entropi ikke er hentet inn vil system henge til det er gjort
- ◆ `/dev/urandom` er en ikkeblokkerende utgave
- ◆ Ikke like sikre som f.eks Tiny, men sannsynligvis gode nok til alle praktiske formål



Tilfeldige tall i Java

- ◆ `java.security.SecureRandom` kan brukes for generere kryptografisk sterke tilfeldige tall
- ◆ Er i motsetning til de andre løsingen uavhengig av maskinvare og operativsystem
- ◆ Du kan velge om du vil gi den et frø, eller om den skal generere sitt eget
- ◆ Frøet blir hashet med SHA-1 algoritmen
- ◆ Lager frø vha tråd-timing
- ◆ Sier ingenting om hvor mye entropi den samler, man må bare stole på at det er nok
- ◆ Boken påstår den kan bruke veldig lang tid, det gjør den ikke lenger



Oppsummering

- ◆ Bruk så gode kilder til entropi som mulig innenfor de rammen du har, hardware om mulig
- ◆ Bruk flere kilder
- ◆ Vær på vakt overfor kilder som en angriper kan overvåke
- ◆ Gjør et estimat over hvor mye entropi som blir samlet