PRNG in IOS

Gaus – PSIRT IM
<gaus@cisco.com>
Overview

- How it started
- What it looked like
- How it was improved
- How was tested
- Possible further improvements
What were looking at

- “Big Bang”, who knows how it looked like?
- Starry night, Copyright NASA, STScI, HubbleSite
- “Strange Attractors and TCP/IP Sequence Number Analysis” by Michal Zalewski
Beginning

- It started by ISNs in TCP session
- They should be unpredictable but they were not
Improving ISNs

- One way is to make ISN as random as possible
- What exactly “random” means?
- Unpredictable and next-bit test
- The existing PRNG was not adequate for the purpose
- The solution is to introduce a new one
How PRNG works

• The universal recipe is the same:
  ➢ Take some fresh entropy and put it into a pot
  ➢ Add more entropy whenever you have a chance and stir it in
  ➢ Serve when needed but not forget to stir
Some mixing tools

- MD5
- SHA
- AES
- DES
More mixing tools
The challenge

• Where to find entropy
• IOS is closed system and it does not have:
  – Hard disk
  – Mouse
  – Keyboard
Some unsuccessful ideas

- MAC or IP addresses
- Packet length, timing between packets
- Environmental temperature
- CPU fan rotation
- Wireless noise, microphone, camera
- “Something” from the memory
More promising ideas

• truerandom() function but IOS is not preemptive
• Timing between consecutive passes in a simple loop
• Time when the function is invoked
• “something” from the memory
How it looks today

- PRNG uses GF-based mixing function and it is extracted using MD5
- Entropy is slow to accumulate
- PRNG passes all statistical tests
How to test a sequence

- How random is your “random” sequence?
- Is “111111111111111111” more random than “010101010101010101” or “10011010100101101”?
- We can only test for statistical properties of a sequence.
Tools used for testing

- Diehard
- NIST Statistical Test Suite
- Some others were tried but were not adequate
Diehard

- Not really user friendly
- Need some knowledge to interpret the results
- Very powerful
- Needs large input (~8*10^9 bits)
• Nicer interface
• Sometimes can be hard to select right parameters and input sequence length
• An par with Diehard
A sample of Diehard output

:: This is the BIRTHDAY SPACINGS TEST ::
:: Choose m birthdays in a year of n days. List the spacings ::
:: between the birthdays. If j is the number of values that ::
:: occur more than once in that list, then j is asymptotically ::
:: Poisson distributed with mean m^3/(4n). Experience shows n ::
:: must be quite large, say n>=2^18, for comparing the results ::
:: to the Poisson distribution with that mean. This test uses ::
:: n=2^24 and m=2^9, so that the underlying distribution for j ::
:: is taken to be Poisson with lambda=2^27/(2^26)=2. A sample ::
:: of 500 j's is taken, and a chi-square goodness of fit test ::
:: provides a p value. The first test uses bits 1-24 (counting ::
:: from the left) from integers in the specified file. ::
:: Then the file is closed and reopened. Next, bits 2-25 are ::
:: used to provide birthdays, then 3-26 and so on to bits 9-32. ::
:: Each set of bits provides a p-value, and the nine p-values ::
:: provide a sample for a KSTEST.
A sample of Diehard output (cont.)

BIRTHDAY SPACINGS TEST, M= 512 N=2**24 LAMBDA= 2.0000

Results for newrnd.bin

For a sample of size 500: mean

newrnd.bin using bits 1 to 24 2.000

duplicate number observed number expected spacings
0       65.       67.668
1       137.      135.335
2       138.      135.335
3       93.       90.224
4       41.       45.112
5       15.       18.045
6 to INF 11.        8.282

Chisquare with 6 d.o.f. = 2.04 p-value= 0.084410
### A sample of STS output

<table>
<thead>
<tr>
<th>Statistical Test</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>0.604458</td>
</tr>
<tr>
<td>Block Frequency ($m = 100$)</td>
<td>0.833026</td>
</tr>
<tr>
<td>Cusum-Forward</td>
<td>0.451231</td>
</tr>
<tr>
<td>Cusum-Reverse</td>
<td>0.550134</td>
</tr>
<tr>
<td>Runs</td>
<td>0.309757</td>
</tr>
<tr>
<td>Long Runs of Ones ($M = 10000$)</td>
<td>0.657812</td>
</tr>
<tr>
<td>Rank</td>
<td>0.577829</td>
</tr>
<tr>
<td>Spectral DFT</td>
<td>0.086702</td>
</tr>
<tr>
<td>NonOverlapping Templates ($m = 9, B = 000000001$)</td>
<td>0.496601</td>
</tr>
<tr>
<td>Overlapping Templates ($m = 9$)</td>
<td>0.339426</td>
</tr>
<tr>
<td>Universal ($L = 7, Q = 1280$)</td>
<td>0.411079</td>
</tr>
<tr>
<td>Approximate Entropy ($m = 5$)</td>
<td>0.731449</td>
</tr>
<tr>
<td>Random Excursions ($x = +1$)</td>
<td>0.000000</td>
</tr>
<tr>
<td>Random Excursions Variant ($x = -1$)</td>
<td>0.000000</td>
</tr>
<tr>
<td>Lempel Ziv Complexity</td>
<td>0.398475</td>
</tr>
<tr>
<td>Linear Complexity ($M = 500$)</td>
<td>0.309412</td>
</tr>
<tr>
<td>Serial ($m = 5, \nabla \Psi_m^2$)</td>
<td>0.742275</td>
</tr>
</tbody>
</table>
Possible improvements

- The current PRNG is not the fastest in the block
- Possible replacements with AES-based
- Retaining entropy over reloads
Links

- http://lcamtuf.coredump.cx/newtcp/
- http://www.cs.berkeley.edu/~daw/rnd/mabrand
More links

- http://stat.fsu.edu/pub/diehard/