Entropy

Finding Random Bits for OpenSSL

Denis Gauthier and Dr Paul Dale
Network Security & Encryption
May 19th 2016

ORACLE

Program Agenda

1. OpenSSL's Entropy
2. Finding Good Quality Entropy
3. Designing an Entropy Module
4. Summary

ORACLE

Definitions

- Entropy is a measure of possible patterns present within random data
- Entropy is a random string which has no shorter description than the string itself (Kolmogorov complexity)

OpenSSL Entropy: FIPS Case

- OpenSSL provides entropy for the non-FIPS case, but not for the FIPS case
- Bring Your Own Entropy (BYOE)
  - OpenSSL Security Policy: "**Module users (the calling applications) shall use entropy sources** that meet the security strength required for the random number generation mechanism as shown in [SP 80090] Table (Hash_DRBG, HMAC_DRBG), Table 3 (CTR_DRBG) and Table 4 (Dual_EC_DRBG). This entropy is **supplied by means of callback functions**. Those functions must return an error if the minimum entropy strength cannot be met."
- For years, FIPS validation authorities had a "no comment" entropy policy

OpenSSL Entropy: Non-FIPS Case

- For *nix platforms, seeds its PRNG **at startup** with 256 bits from:
  - Files: **/dev/urandom**; /dev/random; /dev/srandom
  - Sockets: /var/run/egd-pool; /dev/egd-pool; /etc/egd-pool; /etc/entropy
  - Other: Add in pid, uid & time (not much entropy here)

- One single global PRNG for all crypto for all time (while OpenSSL is running)
  - Never re-seeds – no forward secrecy

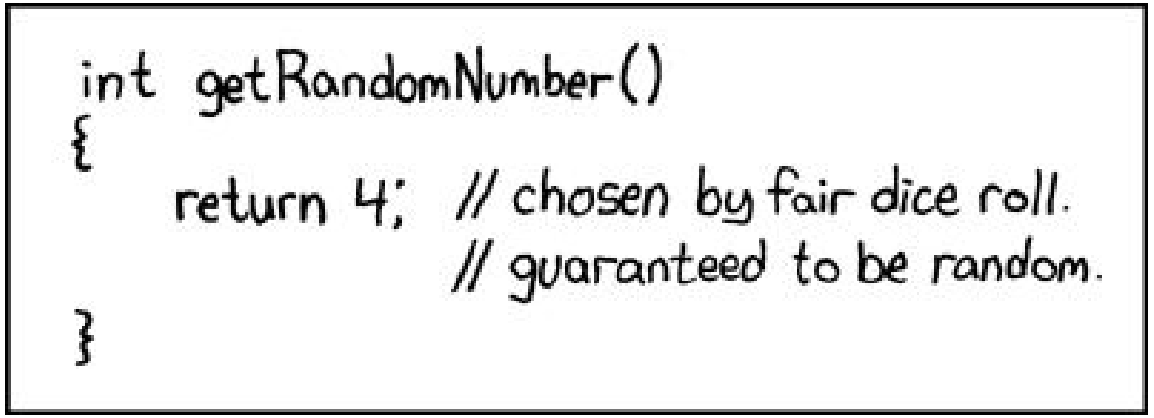
OpenSSL Entropy: Conclusions

- BYOE: Find some entropy for OpenSSL
- Periodically reseed the OpenSSL PRNG to provide forward secrecy

Reseeding: Not Everyone Will Get it Right

- Deadlines and competing priorities
- “A little learning is a dangerous thing” - Alexander Pope
- Example random seed from Ubuntu forums:
  - (unsigned int)time(0) + getpid()
  - developer advised to remove getpid()
- Not enough entropy

- XKCD.com



Reseeding: Finding Good Quality Entropy

- Do your entropy sources produce **enough entropy**?
- Do your entropy sources produce **unpredictable data**?
- Does your entropy come from **trusted sources**?

Reseeding: Enough Entropy

- There is no universal solution – the amount varies with each source and its operating environment
- Hardware: if available, HRNGs produce a lot of entropy
- Software: higher amounts come from timers (low order bits)
  - It is hard to predict when an event will take place
  - Most systems have many events interacting in unpredictable ways

Reseeding: Enough Entropy => Not Much

- time(2) or getpid(2)

- Keyboard strokes and mouse movements
  - No good on headless systems and when operators are away
  - When users try to generate entropy, movements become repetitive

- Use these but don’t rely on them

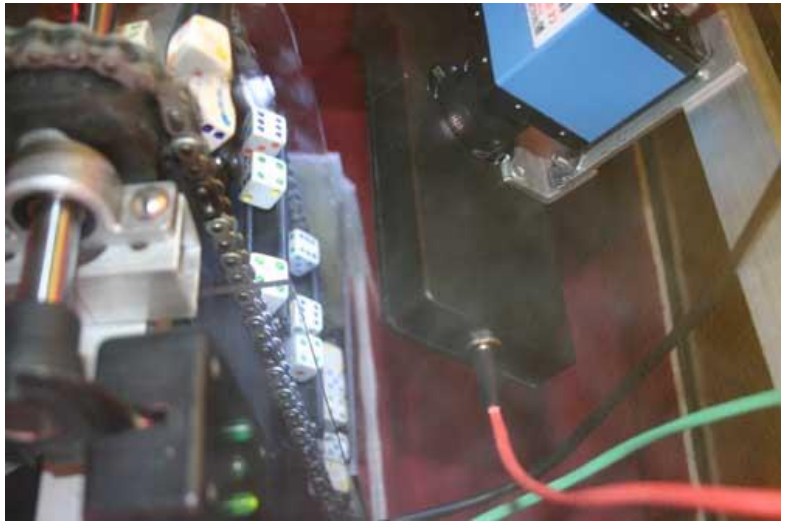
Reseeding: Enough Entropy => More

- Good but not always reliable:
  - Turbid, Randomsound
  - /dev/urandom

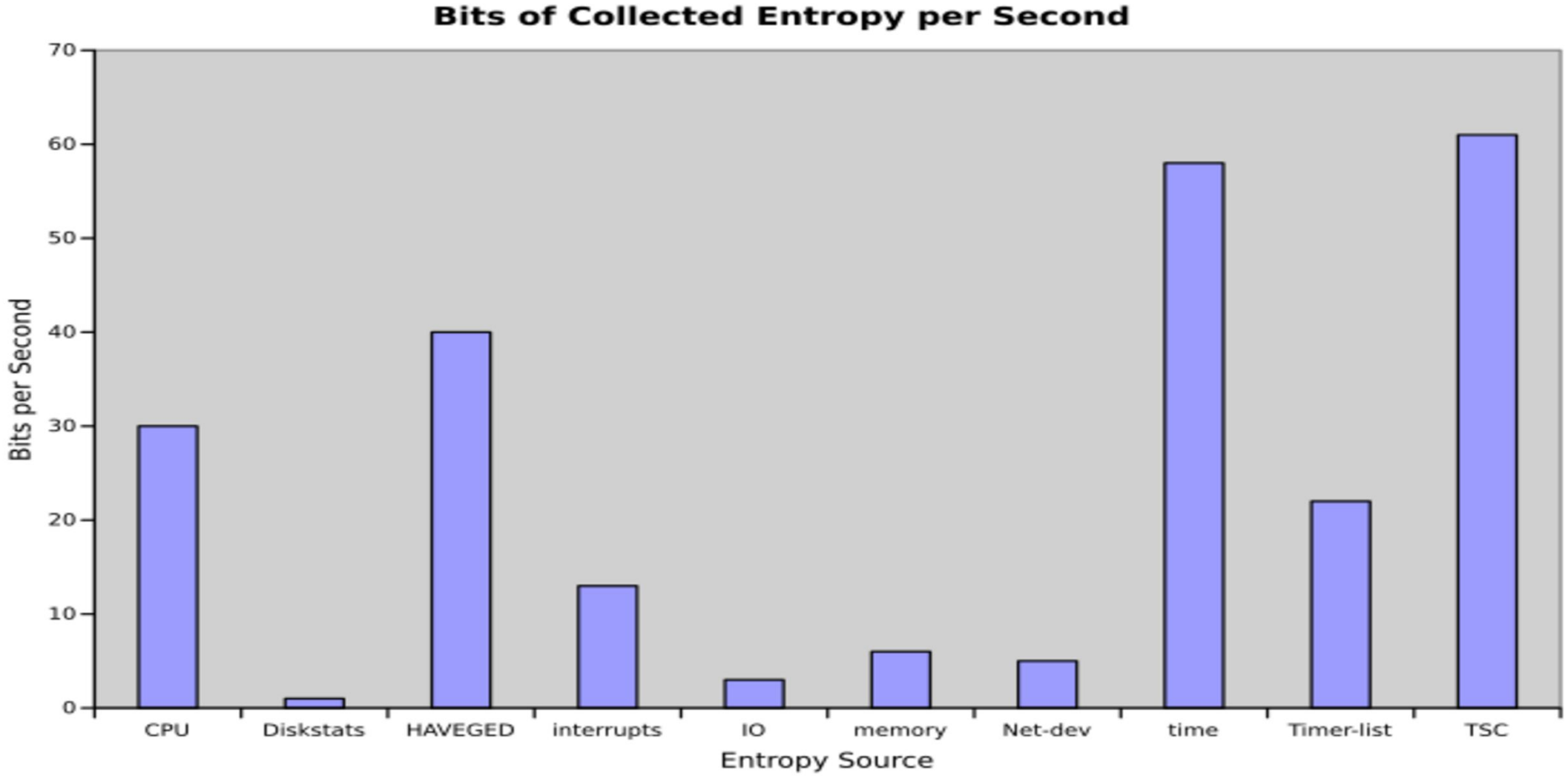
- Good but slow:
  - CPU & memory info, IO, process times, disk stats, interrupts, network interface stats, kernel timer stats
  - /dev/random on *nix systems

Reseeding: Enough Entropy => Plenty

- Software
  - CPU jitter (Haveged)
  - High speed timers
  - Instruction counter (TSC)
  - Timer jitter (Maxwell)
- Hardware:
  - RDRAND
  - Dedicated HRNGs



Reseeding: Enough Entropy => Generation Rates



Reseeding: Unpredictable Data

- Statistical tests
  - NIST SP800-90B python tool (IID and non-IID)
  - NIST SP800-22 statistical tests
  - Ent
  - Dieharder
  - TestU01 Crush
  - TestU01 Big Crush
- All give false negatives and false positives
- They will detect some non-randomness, but cannot prove randomness

Reseeding: Unpredictable Data

- Statistical tests cannot distinguish entropy from the output of a PRNG
- Some tests cannot distinguish entropy from simple functions
- Libc rand passes SP800-90B IID tests with 7.87 bits/byte entropy

  “Do not use this function in applications intended to be portable when good randomness is needed.” – rand(3) man page

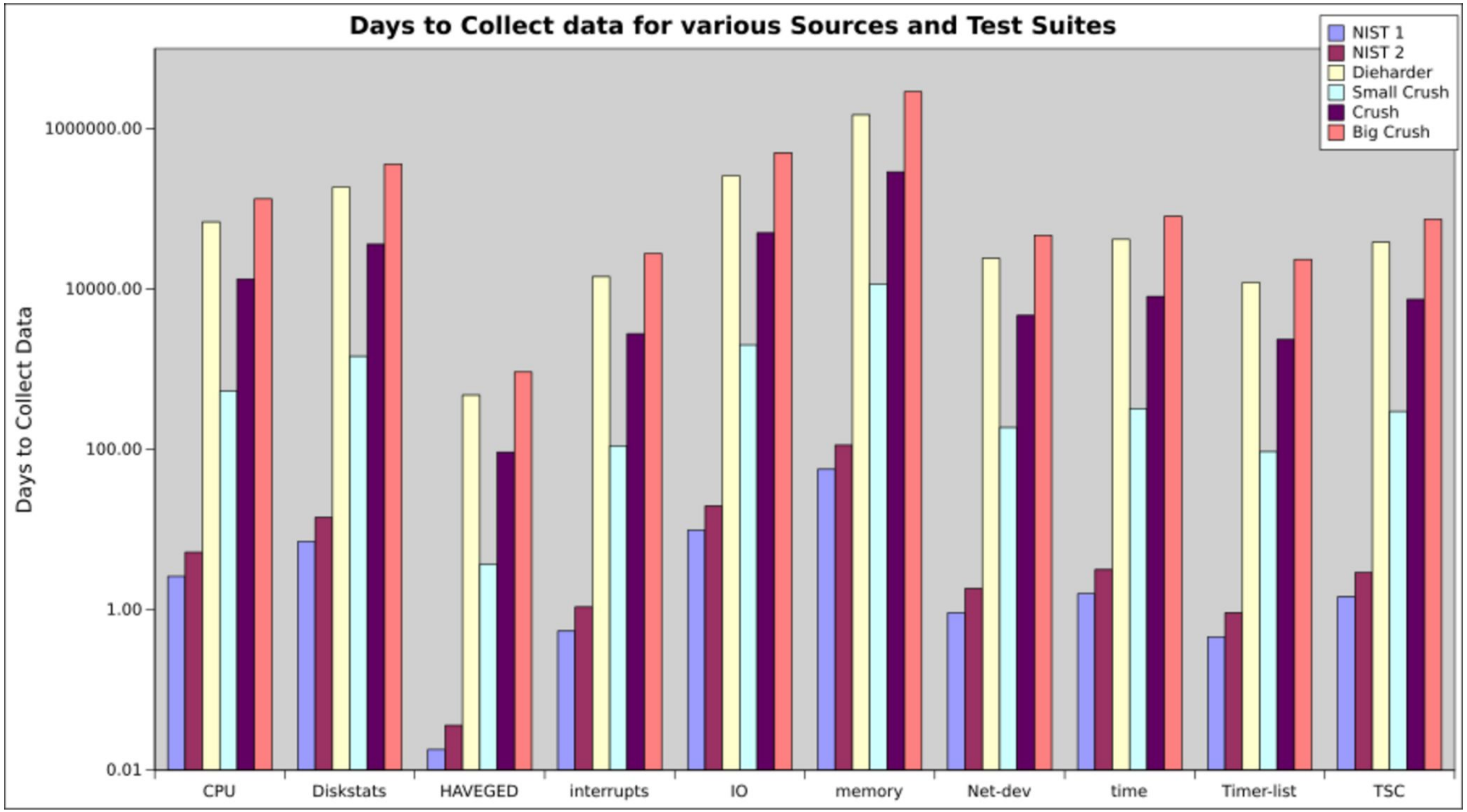
Reseeding: Unpredictable Data

- **Simple and non-random iterated function passes tests**

$$
\begin{aligned}
x_0 &= \pi - 3 \\
x_n &= frac(e^{1+2x_{n-1}}) \\
sample &= \lfloor 2^8 frac(2^{20} x_n) \rfloor
\end{aligned}
$$

Suite	Samples	Result	Entropy / byte	Note
SP800-90B IID	10^6	Pass	7.88698	
SP800-90B nonIID	10^6	Pass	5.74889	
SP800-22	10^6	Pass		
Ent	10^6	Pass	7.999803	
Dieharder	10^9	Pass		7 weak successes
TestU01 Crush	10^9	Fail		44 failures
TestU01 Big Crush	10^9	Fail		83 failures

Reseeding: Unpredictable Data => Time to Collect Samples



Reseeding: Unpredictable Data => Summary

- It is difficult/impossible to measure the quality of your entropy
- Do run statistical tests (and continuous health tests)
  - Catches some poor entropy
  - Estimate how much entropy the source delivers, BUT
- If someone substitutes your entropy, you won't know the difference
- Trust is a central issue when it comes to getting good entropy

Reseeding: Trusted Sources => Careful Evaluation

- Due diligence – evaluate your sources carefully
- Cardinal rule – Transparency is Good (but examine it well…)
- Technical considerations
- Social/political considerations

Reseeding: Trusted Sources => HRNGs

- HRNGs are usually in a black box (IC), but can you trust what is hidden in a black box?
- It would be possible for an HRNG to sabotage the entropy pool in some cases

"We cannot trust" Intel and Via's chip-based crypto, FreeBSD developers say

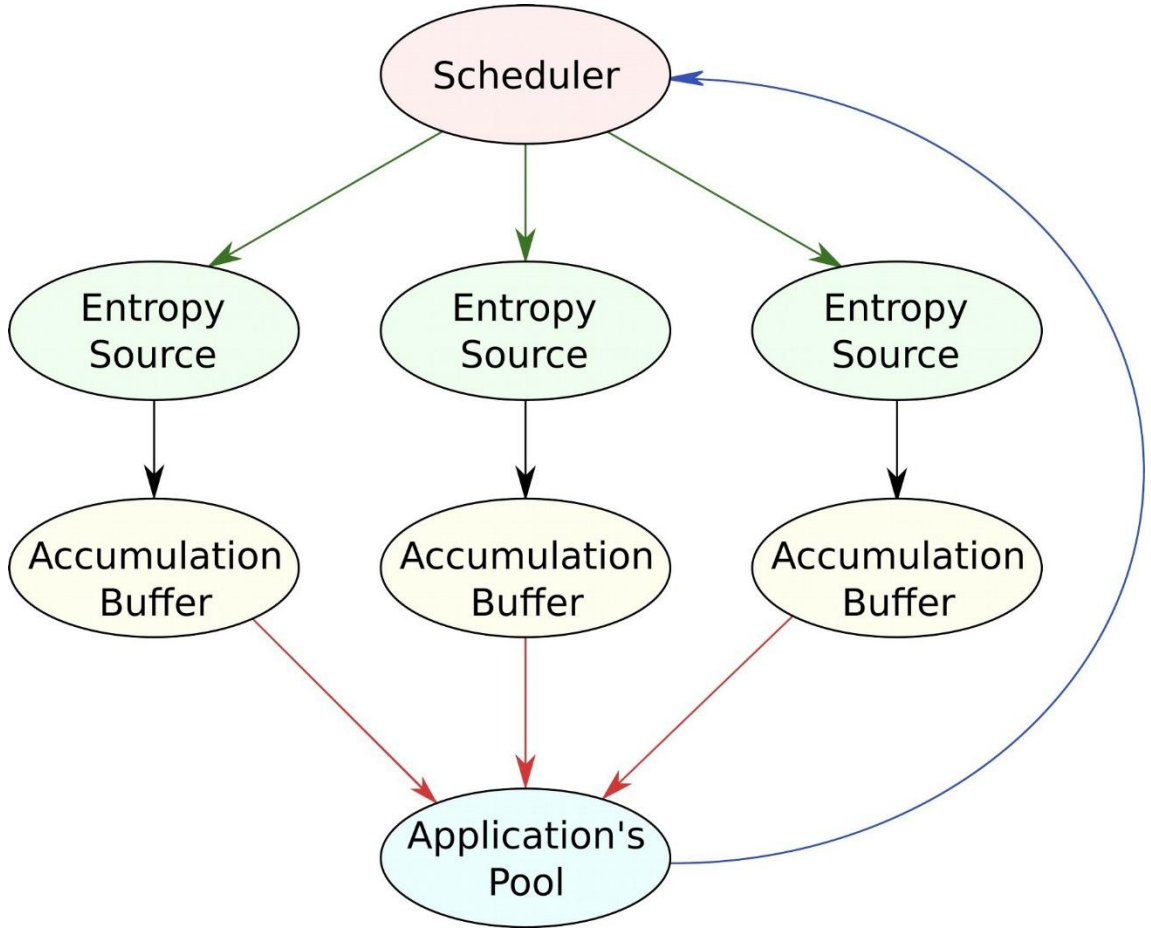
Following NSA leaks from Snowden, engineers lose faith in hardware randomness.

by **Dan Goodin** - Dec 10, 2013 11:00pm AEST

Share Tweet Email 143

Entropy Module Design Considerations

- Diversify
  - multiple independent sources
- Use a scheduler
  - Handles reseeding for forward secrecy
  - Every source has a different entropy generation rate and that determines the appropriate polling interval and when to push to the application's pool
  - Performance: don't starve your system
- Conditioning



Summary

- OpenSSL entropy is weak
- BYOE
- Reseed the OpenSSL PRNG
- Find some independent sources
  - Test it
  - Trust it

Bolt cutters not required – scissors will do.

Questions?

- Contact:

  denis.gauthier@oracle.com