Devsummit – Entropy pools

Taylor 'Riastradh' Campbell
campbell@mumble.net
riastradh@NetBSD.org

EuroBSDcon 2015
Stockholm, Sweden
October 2, 2015

Entropy pool

- Combine environmental observations into small scrambled state.
- Reveal obscured state to kernel or userland `/dev/urandom` for cryptographic or Monte Carlo purposes.
- Inputs: rndsources—clock skew, envsys, hardware RNG, ...
- Outputs: seed for `cprng(9)`, `/dev/urandom`

Security model

- Attacker sees some outputs of `/dev/urandom`: can't predict unseen outputs, past or future.
- Attacker sees kernel memory: can't predict past unseen outputs.

Current implementation

- Input:
  - Hardware driver calls `rnd_add_data`.
  - `rnd_add_data` acquires global mutex (!) and enters sample into global sample queue.
  - Softint processes sample queue.
  - For each sample: feed into 4096-bit LFSR.
- Output:
  - Compute 160-bit SHA-1 of 4096-bit LFSR state.
  - Feed hash back in as if input.
  - Reveal xor of two 80-bit halves of hash.

Crypto analysis?

- No scrutiny by cryptographers to my knowledge since it was written in 1997.
- Ad-hoc components: LFSR, SHA-1.
- Old crypto: SHA-1.

Performance analysis?

- One global sample queue protected by mutex.
- Single point of contention for all samples:
  - Every network packet?
  - Every (503rd) uvm fault?
  - Every . . . ?

Proposed new crypto

- Keccak-f1600: single fixed permutation of 1600-bit strings.
- Keccak-f1600 conjectured to 'look random'.
- Can use to build hash function, MAC, PRF, block cipher, stream cipher, . . .
- Keccak (SHA-3) sponge duplex construction:
  - State: 1600-bit Keccak state
  - Input: xor 1088 bits of samples into state, then apply Keccak permutation
  - Output: reveal first 256 bits of state, then apply Keccak permutation
- Proven to have same security as, e.g., SHA-3—reduces to security of Keccak permutation.

Proposed new state management

- Per-CPU entropy pool.
- Input: Xor up 1088 bits of samples into pool at a time. (No interprocessor synchronization.)
- Input: When buffer full, schedule softint to apply Keccak permutation; drop samples until that happens. (No interprocessor synchronization.)
- Output: Cross-call to extract output from all per-CPU entropy pools as input into a global entropy pool, then extract output from that one.

Questions

- Throughput of SHA-1/LFSR vs Keccak?
- Any other questions?

(. . . when will I have time to finish my draft implementation?)