Profiling Entropy Collection in the Linux Random Number Generator

Justin MacIntosh
Richard Roberts

Advisor: Nadia Heninger

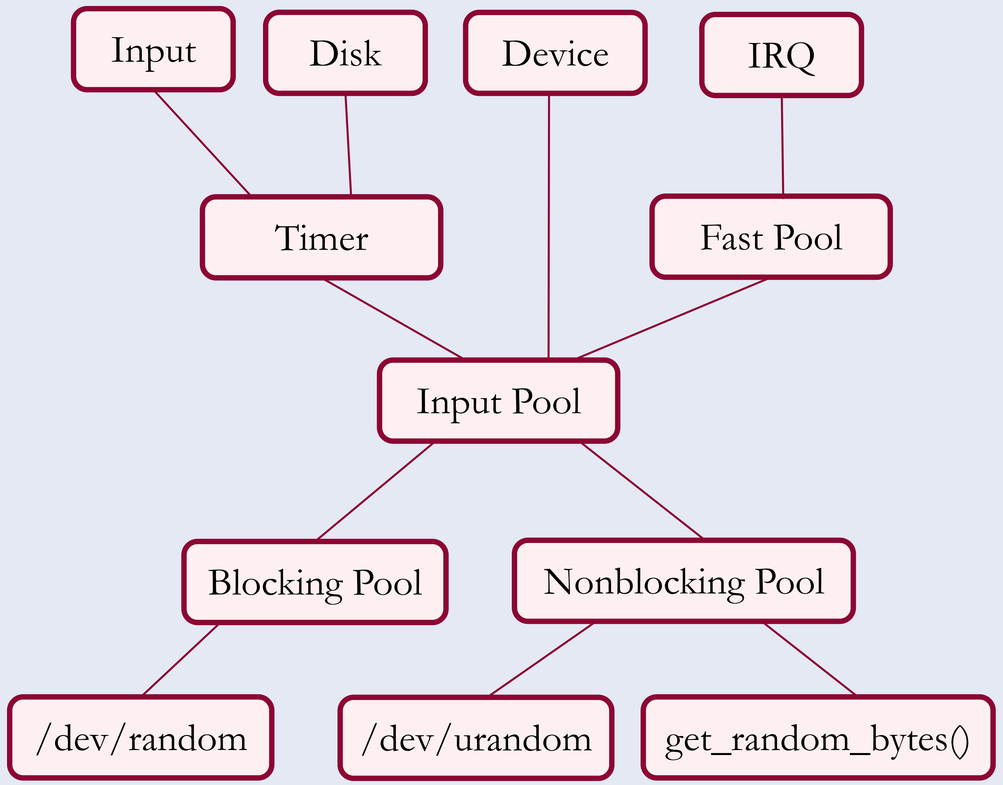
Abstract

The Linux Random Number Generator (LRNG) is the primary source for random number generation on devices running Linux-based operating systems.

Our goals are to create a tool that allows researchers and kernel developers to profile the LRNG without intervening with its functionality, and to use this tool to gather and analyze data from multiple real-world workflows.

Random Numbers

- Unpredictable random numbers are necessary to maintain privacy and security
  - Cryptographic key generation
  - Execution stack management
- Generating unpredictable values from a deterministic process is difficult
- Defenses are useless if ‘random’ numbers are actually predictable



Our Process

Experimentation Machine

- Manipulate Kernel (v 4.2) to send logs
- Main source: drivers/char/random.c
- Ensure sending logs adds no entropy (cannot store on local disk)
- Netpoll sends logs over UDP
- Begin logging after computer boots
- Log when entropy is added and when random values are requested
- Keep running totals of events
- Send logs on command

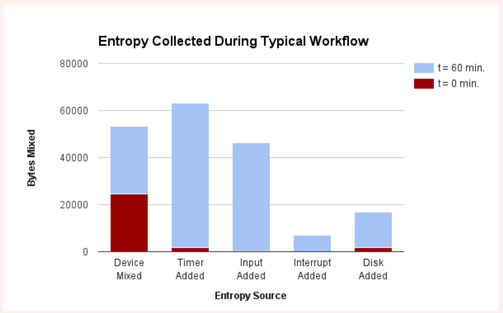
Log Collection Machine

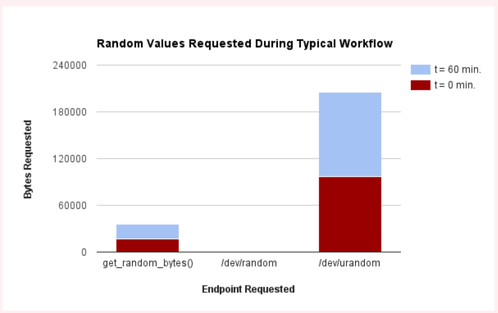
- Logs defined by XML schema
- Python client parses incoming logs
- Parsed logs stored in SQLite database
- Can display results in real time

Entropy

- Entropy is a measure of unpredictability
- LRNG harvests entropy from its environment, such as user input
  - Entropic events are mixed into pools
  - Random numbers extracted from pools
- User input may be an attack vector

Analytical Results





- get_random_bytes() is called every time a new application is executed (for ASLR)
- GPG key generation was the only program tested that requested from /dev/random
- The privacy-minded TOR browser requested approximately the same number of bytes as the Firefox browser
- Newer versions of the Linux kernel finish the boot process with more entropy than older versions (notably, v 3.5)