CTF in the GNU Toolchain

Nick Alcock <nick.alcock@oracle.com>
Indu Bhagat <indu.bhagat@oracle.com>

Contents

What is CTF?

- Model of C type system (C89 plus some C99 features, so far) plus a mapping of func/data symbols to types (incl arg types); each container-full of types is known as a *dictionary*
- Generated by GCC (patches under review), merged and deduplicated by linker (GNU ld patches under review): just pass -gt and everything works; used by GDB (patch under review) when DWARF is not available
- ~2% the size of the DWARF debuginfo (*before* dedup): smaller afterwards: ~5.5MiB for the whole kernel (using an old deduplicator: the new one will be better)
- CTF section linker is implemented in a library in binutils (libctf.a): a specialized kernel CTF deduplicator uses this library to generate an out-of-executable CTF archive for the kernel; userland CTF needs no specialized tools
- Spec: http://www.esperi.org.uk/~oranix/ctf/ctf-spec/index.html, http://www.esperi.org.uk/~oranix/ctf/ctf-spec.pdf
- The libctf library (which reads and writes CTF) is in binutils as of a few months ago: https://sourceware.org/git/gitweb.cgi?p=binutils-gdb.git;a=blob;f=include/ctf-api.h;hb=HEAD

Comparison to Solaris CTF

- Direct descendant of Solaris CTF with different magic number and section name (“.ctf”).
- More sections (label (currently unused), func / object symbol, "variable", mapping strings to types, used for kernel data syms)
- Far higher limits (2^32 types and strtab size, 2^25 struct/union/enum members, 32 type kinds etc)
- Different encoding for bitfields (slices, plus legacy int/float encoding: support for enum bitfields).
- New archive format for grouping multiple CTF dicts into a single entity

Comparison to BTF

- Similar differences to the previous slide: BTF and Linux CTF are related: BTF was inspired by Solaris CTF and is very similar. But there are more differences.
- CTF has one-level parent/child relations between dicts (dicts can share a parent with common types): BTF doesn’t, so doesn’t handle conflicting types between CUs (as far as I can tell)
- Different type kinds (CTF has floats of various sorts). CTF has no function proto type (giving each argument a name), but only a function pointer type: this looks useful, will add it
- CTF can share strtabs etc with the containing ELF object, which makes no sense for BTF
- CTF is much older than BTF, but its change rate is reminiscent of a younger project: radical changes started last year and continue, with a backward-compatibility promise for data (old formats always readable)

Why not just use BTF?

- CTF has wider scope: all of C99/C11/GNU C and eventually other languages
- CTF has wider range limits. The kernel is big. CTF will soon be more compact, as well
- libctf will be able to emit BTF anyway soon, or indeed translate BTF to CTF and vice versa: so you can keep using BTF while not needing a kernel-specific deduplicator or translator from DWARF if you like – in the future when people can rely on having a suitably recent GCC and binutils
- GCC has gained BPF support, and CTF support is in progress: so we will be able to generate CTF for BPF programs

Next steps: API

- In-kernel-usable libctf: a no-malloc read-only variant
- Support for big enum values > 2^32 (the kernel has some)
- New section for kernel function symbols (mapping strings to return / arglist types, like the function section does for ELF syms).

Next steps: user-visible

- Support more of GNU C: the kernel uses vector extensions, etc
- Translation to/from BTF, legacy CTF, and similar formats, whenever possible
- Add a backtrace section: describe most parameters while being much simpler than DWARF (no interpreters), and oriented to online debugging: still being designed

Next steps: CTF format

- Compactness compactness compactness!
  - Smaller structure representing types when the type ID is low: shuffle CTF before emission to put frequently-referenced types at low IDs; structure members gain a per-structure ‘constant prefix’
  - Burrows-Wheeler compression; LZMA compression
- Get rid of archives: teach CTF about translation units

Next steps: Helping the kernel

- dwarves contains independent code to read Solaris/FreeBSD CTF: maybe libctf can do the job for it
- Acting as a source of data for the kernel backtracers? This means we can dump args and their types and chase them in backtraces. Several approaches spring to mind:
  - Separate userspace helper using libctf: easier, not usable in atomic context or when panicking
  - In-kernel: means we need the non-malloc variant and CTF in a loaded section
- With regard to backtracing, we are sort of the opposite of orc (compactness over simplicity), but maybe we can take ideas from orc anyway (we are already borrowing from DWARF)
- Perhaps we can use this to help the kABI checker?
- More ideas here: https://lwn.net/Articles/795384/