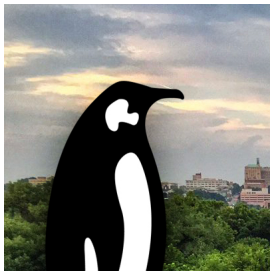

Contribution ID: **237** Type: **not specified**

Graph-based ABI analysis for fun and profit

Monday, 13 November 2023 11:20 (50 minutes)

Analyzing an ELF binary such as a shared library or a Linux kernel image to deduce properties about its exported API and ABI has been done in multiple ways in the past. Most approaches have in common the general mechanism of first extracting information from the binary, then storing it in an intermediate format and lastly comparing it against the result of another extraction. Applications for this **ABI monitoring** mechanism include preserving the backwards compatibility of an API at ABI level or gaining insights about changes caused by arbitrary influences to how the binary is produced (different compiler, different flags, actual code changes, etc.).

While DWARF is the usual source of type information accompanying ELF symbol information, its primary purpose is not to describe an ABI surface, but rather to provide supplemental information that is needed to effectively debug such binaries. More recently, other formats have been explored as alternative sources of type information. Among them are CTF and BTF.

An experimental implementation of a BTF reader and comparison algorithm eventually became the (open source) STG (Symbol Type Graph) project to explore the advantages and disadvantages of an entirely graph-based approach. STG can consume BTF and XML inputs and we've been using STG to enforce stable Android (kernel and library) ABIs. Our ABI graph equivalence algorithm is founded on sound mathematical principles.

Adding native DWARF support to STG was a major undertaking as turning arbitrary DWARF into a representative graph required a lot of research and experimentation and we took a data-driven approach in how we overcame challenges (particularly around type deduplication).

We would like to take this opportunity to go into much more detail about the STG internals, design choices we made, interesting things we learned, including how to:

- design a compact storage format that is suitable for version control systems
- design an in-memory graph data structure for high performance
- traverse the graph structure efficiently in the presence of cycles
- efficiently extract and deduplicate DWARF type information
- filter type information that is available but not relevant to public ABIs

Primary authors: PROCIDA, Giuliano (Google); MÄNNICH, Matthias (Google)

Presenter: MÄNNICH, Matthias (Google)

Session Classification: Toolchains

Track Classification: Toolchains Track