BAR: BAP Annotated Reference

BAR: BAP Annotated Reference

Introduction

Binary Analysis Platform is a framework for writing program analysis tools, that target binary files. The framework consists of a bunch of libraries, plugins and frontends. The libraries provide code reusability, plugins facilitate extensibility and frontends serve as entry points.

The BAP Standard Library, also known just as bap library, is the core component, around which the rest of the Platform is built. Start by reading its manual. Frontends come with comprehensive manuals, that can be accessed by using `--help` command line options, or via the `man` command, if the manpath is configured correctly. Finally, you can access a man page for a plugin using `--<PLUGIN>-help` command line option of a frontend, e.g., `bap --map-terms-help`.

The document is autogenerated from the library mli files, using our `bapdoc` utility, that relies on the standard `ocamldoc` and enhanced html generator `argot`. The referece part of the doc is optimized for using from an IDE powered by `merlin`. Although it should be also readable and searchable directly from the browser. The type manifest search is capable of finding values by type signatures, by using fuzzy search techniques and unification over different representation over semantically same types.

Libraries

Core libraries

- bap - BAP standard library
- regular - regular inductive types
- future - coinductive types
- graphlib - graph library

Hardware Architectures

- arm - ARM architecture
- x86-cpu - x86 family

Programming Languages

- api - interface for adding new languages
- abi - interface for adding new ABI
- c - support library for C language

Auxiliary libraries

- traces - loading execution traces
- bml - an extensible DSL for term transformation
- byteweight - an interface to byteweight implementation
- demangle - custom name demanglers
- ida - call IDA from OCaml
- microx - a library for code microexecution
- build - BAP build system as an ocamlbuild plugin
- text-tags - Use semantics tags to format your texts

Frontends

- `bap` - bap main frontend
- `bap-mc` - machine code playground
- `bap-byteweight` - create, obtain and evaluate byteweight signatures

Plugins

- `abi` - apply abi information to a project
- `api` - add parameters to subroutines based on known API
- `arm` - provide ARM lifter
- `byteweight` - find function starts using Byteweight algorithm
- `cache` - provide caching services
- `callsites` - annotate callsites with subroutine's arguments
- `cxxfilt` - provide c++filt based demangler
- `demangle` - demangle subroutine names
- `dump-symbols` - dump symbol information as a list of blocks
- `elf-loader` - read ELF and DWARF formats in a pure OCaml
- `emit-ida-script` - extract a IDA python script from bap

- `frontc-parser` - parse c files with FrontC
- `ida` - use ida to provide rooter, symbolizer and reconstructor
- `llvm` - provide loader and disassembler using LLVM library
- `map-terms` - map terms using BML DSL
- `objdump` - use objdump to provide a symbolizer
- `phoenix` - output project information in a phoenix format
- `piqi-printers` - provides piqi serialization for main data types (BIL, IR)
- `print` - print project in various formats
- `propagate-taint` - propagate taints through a program
- `read-symbols` - read symbol information from file
- `taint` - taint specified terms
- `trace` - manage execution traces
- `warn-unused` - warn about unused results of certain functions
- `x86` - provide x86 lifter

Tools

- `baptop` - run BAP interactively
- `bapbuild` - build BAP plugins
- `bap-server` - call to BAP via JSON RPC
- `bapbundle` - bundle data with your code