BPF IETF standardization update & roadmap

Dave Thaler

IETF BPF WG chartered tasks

The working group will produce one or more documents on the following work item topics (with intended document status annotations, e.g. [PS] Proposed Standard and [I] Informational):

- [PS] the BPF instruction set architecture (**ISA**) that defines the instructions and low-level virtual machine for BPF programs,
- [I] **verifier expectations** and building blocks for allowing safe execution of untrusted BPF programs,
- [PS] the **BPF Type Format (BTF)** that defines debug information and introspection capabilities for BPF programs,
- [I] one or more documents that recommend conventions and guidelines for **producing portable BPF program binaries**,
- [PS] cross-platform **map types** allowing native data structure access from BPF programs,
- [PS] cross-platform **helper functions**, e.g., for manipulation of maps,
- [PS] cross-platform BPF **program types** that define the higher level execution environment for BPF programs, and
- [I] an **architecture and framework** document.

Current state of the ISA document

- WG state: Submitted to IESG for Publication
- IESG state: In Last Call (ends tomorrow 2024-05-16)
  - OPSDIR Last Call Review done:
    - “As always, well-considered and well-written”
  - GENART Last Call Review due 2024-05-16
  - SECDIR Last Call Review due 2024-05-16
  - …
- IANA review state: Review needed

Future states

- **Waiting for AD Go-Ahead:** AD waiting for document update to address IETF Last Call comments
- **IESG Evaluation:** On agenda for IESG telechat to ballot
- **IESG Evaluation :: Revised I-D Needed:** AD waiting for document update to address IESG comments
- **RFC Ed Queue:** RFC editor converting to RFC format (and updating any references or IANA assignments needed) and assigning tentative #
- **Authors' Final Review (AUTH48):** Author(s) check that RFC editor pass is ok.
- **RFC Published:** Immutable RFC appears with assigned #

IETF Last Call feedback

#134: abstract is inadequate

- Suggest expanding it to same as introduction

#135 (AD) and #140:

- use MUST/SHOULD/MAY language?
- Explicitly say BTF is out of scope / future work
- Remove “0-10” as register range, leave to psABI. Same for return r0?

#136: clarify sign extension of 64-bit use of 32-bit imm

- mov32 says “dst = src” . When src is imm, it’s *unsigned* but doesn’t say so
- jle says “PC += offset if dst <= src” where comparison is unsigned
- Given mov32 doesn’t say, can be ambiguous to readers whether:
  a) dst <= (u64)(u32)imm // NO
  b) dst <= (u64)(s64)imm // YES

#141: Fix the description of ‘src’ in ALU instructions [Puranjay’s patch]

Next steps after ISA

- ~~[PS] the BPF instruction set architecture (**ISA**) that defines the instructions and low-level virtual machine for BPF programs,~~
- [I] **verifier expectations** and building blocks for allowing safe execution of untrusted BPF programs,
- [PS] the **BPF Type Format (BTF)** that defines debug information and introspection capabilities for BPF programs,
- [I] one or more documents that recommend conventions and guidelines for **producing portable BPF program binaries**,
- [PS] cross-platform **map types** allowing native data structure access from BPF programs,
- [PS] cross-platform **helper functions**, e.g., for manipulation of maps,
- [PS] cross-platform BPF **program types** that define the higher level execution environment for BPF programs, and
- [I] an **architecture and framework** document.

verifier expectations

- Intended to describe what a verifier is expected to ensure, e.g.:
  - No undefined instructions or registers
  - Adherence to API contracts (helpers, kfuncs, ctx, etc.)
  - Safe memory pointer dereferencing
  - Termination
- Existing documents to help inform what to put in standardization:
  - verifier.rst currently in tree but not under standardization
    - Contains details about the Linux kernel verifier
  - Academic papers like PREVAIL paper in PLDI ’19
  - Various emails in the archive such as recent thread:
    - [BPF Security] what security properties does verifier guarantee?

BTF

- btf.rst currently in tree but not under standardization
- First step is format transformations to create a file under standardization
  - IETF packet format for structures
  - Tables instead of “#defines”
  - Omit Linux kernel specific info/terms

psABI

- The only other file in Documentation/bpf/standardization today
- Proposed scope:
  - Topics for any psABI definition to answer (see earlier slide)
  - Today's Linux kernel answers as an example of how to answer
- Maybe also:
  - A specific psABI definition can have an ID usable with tools (e.g., compilers)
  - Each runtime can then document which psABI it uses (by ID)
    - IANA registry for IDs and pointers to associated documentation?
  - But we also don't want to encourage lots of psABIs

ELF profile for BPF

- draft-thaler-bpf-elf is one proposal to document
  - Contains info from elf.rst
  - Contains func/line info from btf.rst (but likely should be in BTF spec once one exists in standardization)
  - Contains some TODOs that also depend on BTF spec (.maps section contents, .BTF* section contents)
  - Needs BTF spec done first or concurrently
- Some debate happened on whether to do in IETF or SystemV or…
  - ELF itself is a SystemV spec
  - Intent was that it is a BPF-specific profile so could be in IETF
    - But defines “e_machine must be set to EM_BPF (247)” in the ELF header
  - Discussion punted until after ISA is done to focus on that one
  - Does “one or more documents that recommend conventions and guidelines for producing portable BPF program binaries,” include psABI and ELF, or only psAPI, or …?