CTF Frame in Linux kernel

Indu Bhagat

Toolchains Track
CTF Frame - overview

• A simple unwind format for virtual stack unwinding
  - Fast unwinding using a small unwinder
  - No stack machine, no complex expression encoding
  - Supported on x86_64 and aarch64

• Oversimplification: Akin to interpreted DWARF EH frame info, but for reduced state
  - CTF Frame only recovers: CFA, FP, RA

• Oversimplification: Similar to ORC in principle
CTF Frame - overview

- Not tied to CTF debug for unwinding
  - .ctf_frame can be used without .ctf section
CTF Frame Row Entry

- One self-sufficient FRE per PC (similar to ORC) to recover CFA, FP, RA. Helps unwind fast (less CPU intensive unwind)

```c
// CTF_FRE_addr1 / addr2 / addr3 =
struct {
    uint<8> / uint<16> / uint<32> fre_start_address;
    CTF_Frame_FRE_Info fre_info;
    union {
        int<8>[fre_info.offset_num] offsets_1B : fre_info.offset_size == 1B;
        int<16>[fre_info.offset_num] offsets_2B : fre_info.offset_size == 2B;
        int<32>[fre_info.offset_num] offsets_4B : fre_info.offset_size == 4B;
    } offsets;
};
```
CTF Frame – more details

• .ctf_frame section – loadable, allocated section in a segment of its own (PT_GNU_CTF_FRAME)

• Supported in GNU Toolchain: Generated by GAS by parsing the .cfi_* directives embedded by the compiler. ld support to merge .ctf_frame.

• Size*: 0.8x (x86_64) , 0.7x (aarch64) relative to EH Frame
# CTF Frame – what can it do?

<table>
<thead>
<tr>
<th>Feature</th>
<th>DWARF-based .eh_frame</th>
<th>CTF Frame</th>
<th>Other application specific formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td>Yes</td>
<td><strong>Almost</strong>*</td>
<td>Yes?</td>
</tr>
<tr>
<td>Fast</td>
<td>Somewhat</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Small unwind info</td>
<td>Yes</td>
<td>Yes</td>
<td>Somewhat</td>
</tr>
<tr>
<td>Small (simple) unwinder</td>
<td><strong>No</strong></td>
<td><strong>Yes</strong></td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td>Application specific</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ABI/Arch support</td>
<td>Extensive</td>
<td>Aarch64, AMD64</td>
<td>x86_64</td>
</tr>
<tr>
<td>Toolchain support</td>
<td>Yes</td>
<td><strong>Yes</strong></td>
<td><strong>No</strong></td>
</tr>
</tbody>
</table>
How can CTF Frame help in the Linux kernel

• Can CTF Frame help objtool for ORC generation?
  – Simple to decode as compared to DWARF opcodes in .eh_frame
  – Simplify (can avoid?) control-flow reconstruction needs (the case of AARCH64 ?)
How can CTF Frame help in the Linux kernel?

- Are there other unwinding needs, e.g., unwinding userspace stacks, where CTF Frame format can be used?
In spirit of improving CTF Frame format

• What compiler optimizations are known to work against the needs of reliable backtraces?
• What patterns/code stubs generated by the toolchain need improved unwind info?
CTF Frame – fully asynchronous?

- NOTE 1 - Some .cfi_* directives are skipped:
  - .cfi_negate_ra_state => No unwind around PAC insn in aarch64
  - .cfi_signal_frame => cannot tag signal frame FDEs
  - .cfi_escape => treated as black box atm, skipped altogether

- Short Answer: No, but its close.

- Q: How much impact does this have for the usecase of reliable stacktraces in the Linux kernel?

- NOTE 2 - CTF Format representation is designed for the most common case:
  - aarch64 return register: LR
  - Only CFA, FP, RA can be recovered
    - X86_64: RA = cfa - 8
    - aarch64: RA = LR
CTF Frame format – key ideas

• One self-sufficient FRE per PC (similar to ORC) to recover CFA, FP, RA. Helps unwind fast (less CPU intensive unwind)
• `readelf --ctf-frame=.ctf_frame <OBJ>`

```
func idx [20]: pc = 0x401636, size = 322 bytes

<table>
<thead>
<tr>
<th>STARTPC</th>
<th>CFA</th>
<th>FP</th>
<th>RA</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000000401636</td>
<td>sp+8</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>00000000000401637</td>
<td>sp+16</td>
<td>c-16</td>
<td>u</td>
</tr>
<tr>
<td>0000000000040163a</td>
<td>fp+16</td>
<td>c-16</td>
<td>u</td>
</tr>
<tr>
<td>00000000000401777</td>
<td>sp+8</td>
<td>c-16</td>
<td>u</td>
</tr>
</tbody>
</table>
```