xcover: Cross-language test coverage with eBPF

FOSDEM 2026

The Problem

Testing functional coverage requires instrumentation

Language-specific tools (Go cover, LLVM cov) need recompilation

Can't measure coverage on production binaries

What is xcover?

Functional test coverage profiler

No binary instrumentation required

Cross-language (works with any ELF binary)

Uses kernel instrumentation via eBPF uprobes

How it works

`BPF_PROG_TYPE_TRACE_UPROBE_MULTI` program

Attach uprobe to each function in binary

Track which functions execute via cookies

Calculate coverage: `ack_funcs / total_funcs * 100`

One program, multiple attachments

1. Userspace passes cookies to uprobe_multi attach via libbpfgo
2. eBPF function handler notifies userspace with the cookie
3. Function identified in userspace by unique cookie

Data structures (eBPF side)

```
/* Ring buffer for function events (256MB) */
struct {
    __uint(type, BPF_MAP_TYPE_RINGBUF);
    __uint(max_entries, 1 << 28);
} events SEC(".maps");

/* Hash map to track seen functions */
struct {
    __uint(type, BPF_MAP_TYPE_HASH);
    __uint(max_entries, 40960);
    __type(key, u64);    /* Function cookie */
    __type(value, u8);   /* Seen marker */
} seen_funcs SEC(".maps");

/* Event sent to userspace */
struct event_t {
    __u64 cookie;
};
```

Data structures (userspace side)

```
/* Userspace Go map: cookie → function info */
type UserTracee struct {
    funcs map[cookie]funcInfo
}

type funcInfo struct {
    name   string   // Function name
    offset uint64   // Offset in ELF
}
```

Cookie generation:

```
for _, sym := range funcSyms {
    // Hash function name to generate cookie
    t.funcs[cookie(utils.Hash(sym.Name))] = funcInfo{
        name:   sym.Name,
        offset: helpers.SymbolToOffset(exePath, sym.Name),
    }
}
```

Uprobe multi attachment

Batch attachment of uprobes

```
func (t *UserTracer) attachProbe(ctx context.Context) {
    batchSize := bpfUprobeMultiAttachMaxOffsets
    offsets := t.tracee.GetFuncOffsets()
    cookies := t.tracee.GetFuncCookies()

    for i := 0; i < len(offsets); i += batchSize {
        end := i + batchSize
        if end > len(offsets) {
            end = len(offsets)
        }
        t.probe.Attach(ctx, t.tracee.exePath,
                       offsets[i:end], cookies[i:end])
    }
}
```

Uprobe multi support in libbpfgo

Benefits:

- Single syscall attaches thousands of uprobes
- No perf events, no FD per uprobe

Enables efficient batch attachment from Go

github.com/aquasecurity/libbpfgo/pull/486

The eBPF program

```
SEC("uprobe/handle_user_function")
int handle_user_function(struct pt_regs *ctx) {
    __u64 cookie = bpf_get_attach_cookie(ctx);

    /* Skip if already reported */
    if (bpf_map_lookup_elem(&seen_funcs, &cookie))
        return 0;

    /* Mark as seen */
    u8 seen = 1;
    bpf_map_update_elem(&seen_funcs, &cookie, &seen, BPF_ANY);

    /* Send event to userspace */
    struct event_t *event = bpf_ringbuf_reserve(&events,
                                sizeof(struct event_t), 0);
    if (event) {
        event->cookie = cookie;
        bpf_ringbuf_submit(event, ringbuffer_flags);
    }
    return 0;
}
```

Userspace processing

```
func (t *UserTracer) handleEvent(data []byte) {
    var event Event
    binary.Read(bytes.NewBuffer(data), binary.LittleEndian, &event)

    // Lookup function by cookie from ELF symbols
    fun, ok := t.tracee.funcs[event.Cookie]
    if !ok {
        return
    }

    // Track acknowledged functions (deduplicate)
    if _, ok := t.ack.Load(event.Cookie); !ok {
        if t.verbose {
            fmt.Fprintln(t.writer, fun.name)
        }
        t.ack.Store(event.Cookie, struct{}{})
    }
}
```

CLI usage example

```
$ xcover run --detach --path /path/to/bin --exclude "^runtime.|^internal"
$ xcover wait
xcover is ready
$ /path/to/bin test1
$ /path/to/bin test2
$ /path/to/bin test3
$ xcover stop
xcover is stopped
$ cat xcover-report.json | jq .cov_by_func
89.9786897
```

Implementation challenges

Symbolization from ELF symbol tables

Stripped binaries need debug info (e.g., `.gopclntab` for Go)

Ring buffer sizing for high-frequency functions

Kernel limit: ~1M offsets per uprobe_multi attach

Thank you

- github.com/maxgio92/xcover
- github.com/aquasecurity/libbpfgo/pull/486
- docs.ebpf.io/ebpf-library/libbpf/userspace/bpf_program__attach_uprobe_multi
- lwn.net/Articles/930066
- kernel.org/doc/html/latest/trace/uprobetracer.html
- github.com/torvalds/linux/blob/1f97d9dcf53649c41c33227b345a36902cbb08ad/kernel/trace/bpf_trace.c#L43