BÀI 29: KỸ THUẬT DEBUG NÂNG CAO

"Bug là tính năng, Debug là nghệ thuật"

🎯 Mục tiêu bài học

1. **Tư duy Debug:** Chia để trị (Divide and Conquer).
2. **User Space Debug:** `strace`, `ltrace`, `gdb`, `valgrind`.
3. **Kernel Space Debug:** `dmesg`, `Oops` messages, `Dynamic Debug`.
4. **Remote Debug:** Debug từ xa qua mạng.

1. Strace - Theo vết System Call

Khi chương trình chạy sai mà không in lỗi gì cả, hãy dùng `strace`.

> **Tác dụng:** Liệt kê tất cả các system call (`open`, `read`, `write`...) mà chương trình gọi tới Kernel.

bash

Lệnh strace

./my_app

Kết quả ví dụ

open("/dev/led", O_RDWR) = -1 EACCES (Permission denied)
---> Biết ngay lỗi do chưa cấp quyền (chmod).

2. Valgrind - Săn lỗi bộ nhớ

Lỗi **Memory Leak** (rò rỉ bộ nhớ) hoặc **Segmentation Fault** rất khó tìm bằng mắt thường.

bash

Cài đặt

sudo apt install valgrind

Chạy kiểm tra

valgrind --leak-check=full ./my_app

> Valgrind sẽ chỉ ra chính xác dòng code nào `malloc` mà quên `free`.

3. GDB (GNU Debugger)

Ông vua của Debugger. Cho phép dừng chương trình, xem biến, chạy từng dòng.

Trên Embedded (GDB Server):
Do Board yếu, ta chạy `gdbserver` trên Board và `gdb-multiarch` trên PC.

1. **Board:** `gdbserver :9000 ./my_app`
2. **PC:** * `gdb-multiarch ./my_app` * `(gdb) target remote 192.168.1.10:9000` * `(gdb) continue`

4. Kernel Oops & Panic

Khi Driver lỗi (ví dụ truy cập con trỏ NULL), Kernel sẽ in ra một đoạn "Oops".

text
Unable to handle kernel NULL pointer dereference at virtual address 00000000
PC is at my_driver_write+0x20/0x80 [my_driver]

> **Cách đọc:** Nhìn vào `PC` (Program Counter) để biết hàm nào (`my_driver_write`) và độ lệch (`+0x20`) gây ra lỗi. Dùng lệnh `addr2line` hoặc xem file `my_driver.ko` objdump để tìm ra dòng code C tương ứng.

🛠️ PHẦN THỰC HÀNH (LAB 29)

Đi tìm bug ẩn giấu

Yêu cầu

Giảng viên cung cấp một file thực thi `buggy_app` (kèm source nhưng source bị giấu lỗi).

Lỗi giả lập:

1. Chương trình bị treo (Deadlock).
2. Chương trình ngốn RAM từ từ (Leak).
3. Chương trình crash khi nhập chuỗi quá dài (Buffer Overflow).

Nhiệm vụ: Sinh viên dùng `strace`, `valgrind`, `gdb` để tìm ra nguyên nhân và dòng code lỗi.

Q & A

Chúc các bạn fix hết bug cho Đồ án!