به نام خدا

آزمایشگاه سیستم عامل

آشنایی، اجرا و اشکال‌زدایی هسته سیستم عامل xv6

(بخش سوم: اشکال‌زدایی)

مقدمه

کد در حین اجرا ممکن است دارای اشکال باشد. یکی از ابزارهای تسریع اشکال‌زدایی، اشکال‌زداها هستند. در همین راستا، اشکال‌زداهای متعددی برای سیستم‌عامل‌ها و معماری‌های مختلف ارائه شده است. ابزار اشکال‌زدای گنو[1] (GDB)، یک اشکال‌زدای متداول در سیستم‌های یونیکسی بوده که در این بخش از آزمایش روش کار با آن به طور اجمالی بررسی خواهد شد. به عنوان مثال ممکن است یک کد بزرگ، هنگام اجرا دچار خرابی شود. استفاده از GDB می‌تواند دستور منجر به ازکارافتادگی و زنجیره فراخوانی توابع منتهی به این دستور را نمایش دهد. جهت اشکال زدایی با GDB، در گام نخست باید سیستم‌عامل به صورتی بوت شود که قابلیت اتصال اشکال‌زدا به آن وجود داشته باشد. مراحل اتصال عبارت است از:

۱) در یک ترمینال دستور make qemu-gdb اجرا گردد.

۲) سپس در ترمینالی دیگر، فایل کد اجرایی به عنوان ورودی به GDB داده شود.

چنان‌چه در بخش دوم این آزمایش ذکر شد کد اجرایی شامل یک نیمه هسته و یک نیمه سطح کاربر بوده که نیمه هسته ثابت و نیمه سطح کاربر بسته به برنامه در حال اجرا بر روی پردازنده دائماً در حال تغییر است. به این ترتیب، به عنوان مثال، هنگام اجرای برنامه cat، کدهای اجرایی سیستم شامل کد هسته و کد برنامه cat خواهند بود. جهت اشکال‌زدایی بخش سطح کاربر، کافی است دستور gdb _cat و جهت اشکال‌زدایی بخش هسته دستور gdb kernel فراخوانی شود. دقت شود در هر دو حالت، هر دو کد سطح هسته و کاربر اجرا می‌شوند. اما اشکال‌زدا فقط روی یک کد اجرایی (سطح کاربر یا هسته) کنترل داشته و تنها قادر به انجام عملیات بر روی آن قسمت خواهد بود.

۳) نهایتاً با وارد کردن دستور target remote tcp::26000 در GDB، اتصال به سیستم‌عامل صورت خواهد گرفت.

با وارد کردن دستور c کد سیستم به اجرا ادامه خواهد داد. هم‌چنین با فشردن دکمه Ctrl + C می‌توان به اشکال‌زدا بازگشت.

اجرای اولیه اشکال‌زدا

برنامه cat، فایل ورودی را از دیسک خوانده و سپس محتوای آن را در ترمینال نمایش می‌دهد. خواندن از دیسک مستلزم انتقال از سطح کاربر به سطح هسته است. زیرا دسترسی به دستگاه‌های ورودی/خروجی، یک عملیات ممتاز است. بدین منظور یک تابع به نام read() در سطح کاربر وجود داشته که خود منجر به فراخوانی یک تابع در سطح هسته موسوم به sys_read() می‌گردد. چنان‌چه در پروژه بعدی آزمایشگاه خواهید دید، تابع دوم فراخوانی سیستمی نام داشته و یکی از فراخوانی‌های سیستمی پیاده‌سازی شده در هسته سیستم‌عامل است. در شکل زیر بخشی از برنامه cat که مربوط به خواندن از دیسک است نشان داده شده است.

[1] GNU Debugger

```
 7 void
 8 cat(int fd)
 9 {
10   int n;
11
12   while((n = read(fd, buf, sizeof(buf))) > 0) {
13     if (write(1, buf, n) != n) {
14       printf(1, "cat: write error\n");
15       exit();
16     }
17   }
18   if(n < 0){
19     printf(1, "cat: read error\n");
20     exit();
21   }
22 }
```

چنان‌چه مشاهده می‌شود در خط ۱۲، فایل محتوای فایل خوانده شده و در ترمینال چاپ می‌گردد. فراخوانی سیستمی sys_read() نیز که در فایل sysfile.c پیاده‌سازی شده است در شکل زیر نشان داده شده است.

```
69 int
70 sys_read(void)
71 {
72   struct file *f;
73   int n;
74   char *p;
75
76   if(argfd(0, 0, &f) < 0 || argint(2, &n) < 0 || argptr(1, &p, n) < 0)
77     return -1;
78   return fileread(f, p, n);
79 }
```

یک سازوکار مهم در اشکال‌زدایی نرم‌افزار نقطه توقف[2] است. این ابزار برای توقف اجرای برنامه در یک نقطه از پیش تعیین شده، طراحی شده است. این نقطه، آدرسی از کد برنامه می‌باشد. این آدرس را می‌توان بر اساس نام تابع، شماره خط در فایل کد منبع یا مقدار عددی آن به اشکال‌زدا اطلاع داد. به عنوان مثال اگر به دلایلی قصد اشکال زدایی در تابع read() یا بخش سطح هسته آن یعنی sys_read() وجود داشته باشد، می‌توان با استفاده از دستورهای مختلفی که در جدول زیر نشان داده شده است، عمل نمود.

	تابع سطح کاربر read()	فراخوانی سیستمی sys_read()
نام تابع	3b (cat.c:)cat	b (sysfile.c:)sys_read
مکان در کد منبع	b cat.c:12	b sysfile.c:71
مکان در حافظه	b *0x98	b *0x80104b10

۱) یک Breakpoint روی فراخوانی سیستمی sys_read() قرار داده[4] و دستور bt را اجرا نمایید. در مورد خروجی این دستور و رابطه اجزای آن توضیح دهید.

---

[2] Breakpoint

[3] دقت شود در این شیوه تعیین Breakpoint، تنها می‌توان ابتدای تابع دربرگیرنده نقطه اشکال‌زدایی را تعیین نمود. لذا Breakpoint روی خط ۹ (و نه خط ۱۲) قرار می‌گیرد. همچنین بخش داخل پرانتز، اختیاری است.

[4] دقت شود با توجه به این که sys_read() در هسته است، باید نقطه فراخوانی آن در سطح کاربر را متوقف نمایید. این نقطه در فایل usys.S قرار دارد. دقت نمایید read() فراخوانی سیستمی نیست. بلکه یک تابع کتابخانه‌ای است که فراخوانی سیستمی را فراخوانی می‌کند.

Breakpoint را می‌توان پیش از اجرای برنامه یا در زمان توقف اجرا (به سبب توقف روی یک Breakpoint قبلی یا توقف ناشی از Ctrl + C) قرار داد.

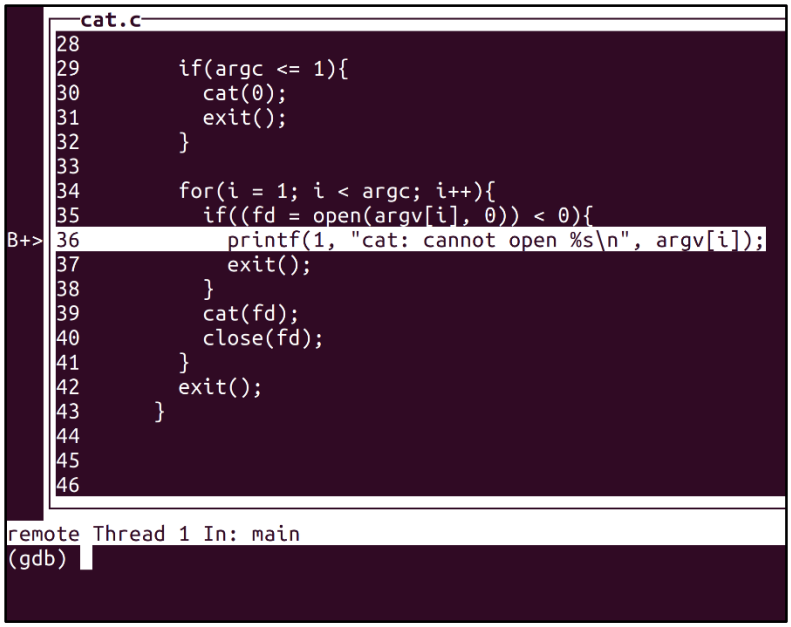
۲) یک حلقه بی‌نهایت در فراخوانی سیستمی sys_read() ایجاد نموده و نقطه اشکال در سطح هسته و کاربر را برای برنامه cat توسط GDB نشان دهید.

پس از توقف روی Breakpoint می‌توان با اجرای دستورهای n(ext)، s(tep) و fin(ish) به ترتیب به دستور بعدی، به درون دستور بعدی (اگر فراخوانی تابع باشد) و به خارج از تابع کنونی (یعنی بازگشت به تابع فراخواننده) منتقل شد. به عبارت دیگر، اجرا گام‌به‌گام قابل بررسی است. بدین معنی که پیش از اجرای خط جاری برنامه سطح کاربر یا هسته، امکان دستیابی به اطلاعات متغیرها و ثبات‌ها فراهم می‌باشد. به این ترتیب می‌توان برنامه را از جهت وجود حالات نادرست، بررسی نمود.

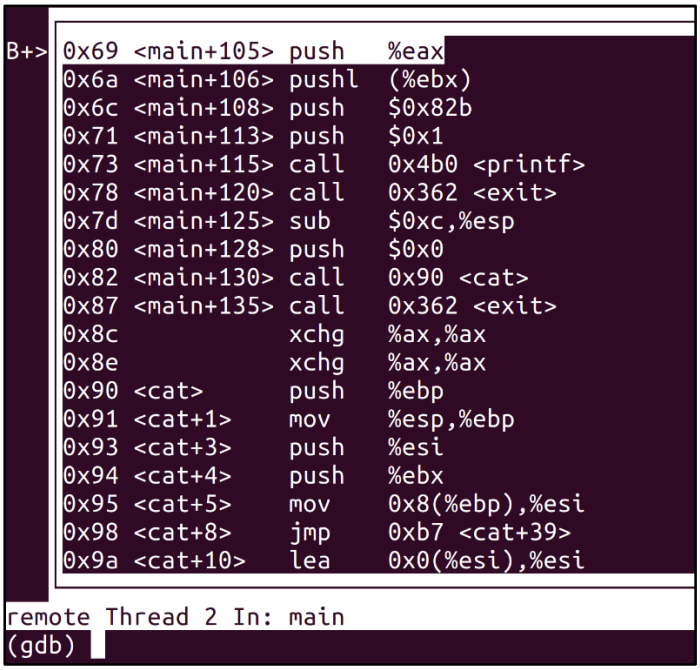
۳) با استفاده از دستور objdump آدرس توابع kernel و برنامه cat_ را استخراج نموده و بازه آن‌ها را بررسی کنید. این آدرس‌ها را گزارش نمایید (فقط توابع). چه نتیجه‌ای می‌گیرید؟

آشنایی با قابلیت‌های سطح پایین‌تر

اشکال‌زدایی برنامه در سطوح مختلفی قابل انجام است. با توجه به این که ممکن است بخشی از کد سطح بالا به دنبال بهینه‌سازی‌های کامپایلری اجرا نشده یا ترتیب اجرای آن تغییر کند، نیاز به اشکال‌زدایی در سطح کد اسمبلی خواهد بود. زیرا کدی که در واقع اجرا می‌گردد، همین کد است. ضمن این که ممکن است برخی از فایل‌های کد منبع، خود به دلایلی از جمله بهینه‌سازی یا عدم پشتیبانی زبان برنامه‌نویسی به صورت کد اسمبلی نوشته شده باشند. جهت آشنایی با این قابلیت، ابتدا یک Breakpoint در خط ۳۶ برنامه cat قرار دهید. سپس با وارد کردن ورودی مناسب، این خط را اجرا نمایید. جهت سهولت در مشاهده روند اجرا دستور layout src را در اشکال‌زدا اجرا نمایید. مشاهده خواهید نمود که مطابق شکل زیر، خط ۳۶ برنامه، علامت‌گذاری شده است.



در این حالت، اجرای دستور layout asm کد اسمبلی همین بخش از برنامه را نشان می‌دهد که در شکل زیر قابل مشاهده است.



مشاهده می‌کنید چهار دستور push، پیش از فراخوانی printf (توسط دستور call) وجود دارد که وظیفه قرار دادن پارامترها و مقادیر برخی ثبات‌ها در پشته را بر عهده دارند. جهت انتقال به دستور بعدی در سطح کد اسمبلی نمی‌توان از دستور s یا n استفاده کرد. زیرا تمامی این خطوط کد اسمبلی، مربوط به یک خط از کد سی (خط ۳۶) هستند.

۴) دستورهای معادل s و n در سطح کد اسمبلی را نام ببرید.

۵) برنامه cat محتوای یک فایل را در ترمینال می‌نویسد. برای نوشتن در ورودی/خروجی نیاز به دسترسی به سرویس‌های هسته سیستم‌عامل است. لذا فراخوانی سیستمی sys_write رخ خواهد داد. یک Breakpoint روی آن قرار دهید. سپس دستور bt را وارد نمایید. خروجی آن چیست؟ توضیح دهید.

۶) در مورد خروجی layout src و layout asm توضیح دهید. (راهنمایی: دستور نخست خروجی layout asm، شماره فراخوانی سیستمی sys_write را در ثبات eax قرار می‌دهد. در واقع طبق واسط باینری برنامه‌های کاربردی[5] (ABI)، این ثبات، باید بدین صورت مقداردهی شود. دو دستور بعدی چه نقشی دارند؟)

جهت انتقال به مد عادی اشکال‌زدایی ابتدا باید کلیدهای Ctrl + x و سپس کلید c فشار داد.

اشکال‌زدایی بر اساس داده

تا اینجا اشکال‌زدایی بر اساس قرار دادن نقاط توقف روی آدرس‌های برنامه بررسی شد. یک روش متفاوت اشکال‌زدایی، اشکال‌زدایی بر اساس داده‌ها بوده که به کمک قطعه کد زیر در ابتدای فایل فرضی foo.c قابل بیان است.

```
int i;
for (i = 0; i < 1000000; i++)
     x = rand();
```

[5] Application Binary Interface

اگر فرض شود مقدار ۱- برای x یک مقدار نامعتبر است، چگونه می‌توان وقوع این حالت را به کمک اشکال‌زدا تشخیص داد؟ البته یک روش، تغییر در کد و افزودن شرطی جهت تشخیص این حالت است. اما تشخیص بدون تغییر در کد توسط دستور زیر میسر می‌شود.

```
b foo.c:2 if (x = -1)
```

اما اگر متغیر x، یک متغیر سراسری بوده و در چندین فایل کد منبع مورد دسترسی قرار بگیرد، چه باید کرد؟ در این حالت تشخیص با تغییر در کد نیز ساده نخواهد بود. در شرایط پیچیده‌تر، ممکن است این دسترسی‌ها به طور غیرمستقیم و توسط اشاره‌گرها رخ دهد. در این حالت بهتر است به جای علامت‌گذاری خط کد (توسط Breakpoint) داده‌ها را علامت‌گذاری نمود. بدین‌منظور watchها طراحی شده‌اند که خود چندین نوع دارند. ساده‌ترین آن‌ها با استفاده از دستور زیر قابل تعریف است.

```
watch *0x12345678
```

در این‌جا نوشتن در این آدرس حافظه منجر به توقف اجرا در نقطه دسترسی می‌گردد. می‌توان watch را روی متغیرهای محلی هم گذاشت. البته باید به حوزه تعریف متغیرها نیز دقت نمود. دقت کنید موارد استفاده watch تنها به نکات مذکور در بالا ختم نمی‌شود. در کدهای همروند یا موازی اشکال‌زدایی بسیار دشوارتر از کدهای ترتیبی است. اگر دسترسی به متغیری به صورت همروند یا موازی صورت پذیرد و مثلاً متغیری مقدار نادرستی بگیرد، استفاده از watch جهت تشخیص حالت اشکال و عامل آن بسیار راهگشا خواهد بود. به‌خصوص که در این شرایط، تکرارپذیری آزمایش نیز به سادگی ممکن نبوده و بسیاری از اجراها منجر به خروجی صحیح می‌گردند.

نکات مهم

- برای تحویل پروژه ابتدا یک مخزن خصوصی در سایت GitLab ایجاد نموده و سپس پروژه خود را در آن Push کنید. سپس اکانت UT_OS_TA را با دسترسی Maintainer به مخزن خود اضافه کنید. کافی است در محل بارگذاری در سایت درس، آدرس مخزن، شناسه آخرین Commit و گزارش پروژه را بارگذاری نمایید .
- همه اعضای گروه باید به پروژه‌ی آپلود شده توسط گروه خود مسلط باشند و لزوماً نمره افراد یک گروه با یکدیگر برابر نیست.
- در صورت مشاهده هرگونه مشابهت بین کدها یا گزارش دو گروه، نمره ۰ به هر دو گروه تعلق می‌گیرد.
- تمامی سؤالات را در **کوتاه‌ترین اندازه ممکن** در گزارش خود پاسخ دهید.