گزارش فاز اول پروژه سیستم‌عامل

امینه احمدی‌نژاد، صبا زرباف

گام اول:

بررسی عملکرد dmesg:

dimesg دستوری است که پیام‌های هسته‌ی لینوکس را با خواندن بافر هسته چاپ می‌کند. از این دستور برای مشکل‌یابی و یا گرفتن اطلاعات سخت‌افزار استفاده می‌شود.

در اندروید ابتدا termux را از play store دانلود کرده و نصب می‌کنیم (از terminal IDE هم می‌توان استفاده کرد، ولی برای نصب ساده‌تر کامپایلر gcc از این برنامه استفاده کردیم). برای نصب کامپایلر gcc از دستور

```
pkg install clang
```

استفاده می‌کنیم. پس از نصب gcc، dmseg را با دستور

```
pkg install util-linux
```

نصب می‌کنیم. اگر دستور dmesg را به تنهایی اجرا کنیم همه‌ی پیام‌های هسته چاپ شده و صفحه پر می‌شود و نمی‌توانیم پیام‌های بالایی را ببینیم. ولی اگر از دستور dmesg l less استفاده کنیم فقط پیام‌های اولیه را چاپ می‌کند. برای ذخیره ۵۰ خط اول از دستور

```
head -50 source file.name  > destination file.name
```

استفاده می‌کنیم. در زیر شرح بعضی از خطهای خروجی dmesg آمده است.

در خط نهم فایل dmesg.txt عملیات fast string غیرفعال شده است:

```
[    0.000000] Disabled fast string operations
```

در خط هجدهم آدرسی از حافظه استفاده (رزرو) شده است:

```
[    0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
```

در خط بیست و نهم بیت NX را فعال کرده که آن یک بیت برای امنیت فضای قابل استفاده است و نشان می‌دهد که چه ناحیه‌هایی استفاده شده و نشده است تا دیگر اتفاقاتی مثل exception رخ ندهد.

```
[    0.000000] NX (Execute Disable) protection: active
```

در خط سی و سوم خانه‌ای از حافظه حذف شده است:

```
[    0.000000] e820: remove [mem 0x000a0000-0x000fffff] usable
```

در خط چهل و دوم MTRR فعال شده است که مجموعه‌ای از قابلیت‌های پردازنده را کنترل می کند تا نرم‌افزار بتواند سیستم را با کنترل دسترسی به محدوده‌ی حافظه توسط CPU ذخیره کند.

```
[    0.000000] MTRR variable ranges enabled:
[    0.000000]   0 base 000000000 mask FC0000000 write-back
```

بررسی عملکرد logcat:

Logcat ابزاری است که پیام‌های سیستمی اندروید را با stack trace (یعنی جایی که پردازه تولید شده) و توضیحات activity لازم را چاپ می‌کند و معمولا برای debug استفاده می‌شود. برای استفاه از این ابزار ابتدا alogcat را روی اندروید نصب می‌کنیم و سپس برای دریافت و ذخیره‌ی پنجاه خط آخر آن از دستور

```
tail -50 source file.name  > destination file.name
```

استفاده می‌کنیم. در ادامه بعضی از این خروجی‌ها توضیح داده شده‌اند.

در خط هفتم دستور r-xp برای فایل text. است. (read execute private)

W/art (3513): 7f8e68ad6000-7f8e68afb000 r-xp 00000000 08:01 1237863 / system/lib64/libutils.so

در خط بیست و چهارم هیچ دایرکتوری‌ای برای فایل پیدا نکرده‌است.

W/ (3513): Failed to bind-mount /system/lib64/x86_64/cpuinfo as /proc/cpuinfo: No such file or directory

در خط بیست و پنج فایل مذکور فقط قابل خواندن بود و نمی‌توانست آن را کپی کند.

W/ (3513): Cannot create code cache directory ./code_cache: Read-only file system.

در خط سی و هفتم هم دکمه‌ی کیبورد زده شده است.

D/EmojiAltPhysicalKeyDetector(1809): onKeyUp() : KeyEvent { action=ACTION_UP, keyCode=KEYCODE_DPAD_UP, scanCode=103, metaState=0, flags=0x8, repeatCount=0, eventTime=1583598, downTime=1583540, deviceId=5, source=0x101 }

گام دوم:

تحلیل سیستم کال اول:

این سیستم‌کال زمان حال را اعلام می‌کند. برای این سیستم کال از تابع ()gettimeofday استفاده می‌کنیم که متغیری از جنس struct timeval را به عنوان ورودی گرفته و خروجی آن تاریخ و زمان بر حسب ثانیه است. (برای تابع مذکور باید کتابخانه‌ی sys/time.h را include کنیم.)

سپس از localtime برای تبدیل ثانیه‌ها به تاریخ و ساعت و دقیقه و ... استفاده می‌کنیم. (برای استفاده از این تابع باید time.h را include کنیم)

تحلیل سیستم کال دوم:

این سیستم کال در یک بازه‌ی مشخص میزان مصرف پردازنده توسط کاربر را اعلام می‌کند. در این قسمت از سیستم کال gettimeofday() استفاده شده است، سپس با خواندن داده‌های فایل proc/stat/ و به دست آوردم نسبت تعداد پردازه‌های این مدت زمان بر پردازه‌های idle درصد مصرف cpu اعلام می‌شود.