LightNVM project

`lightnvm`

实验目的和意义

熟悉Open-Channel SSD架构和lightNVM

环境搭建

- Qemu-Based NVMe emulator
  https://github.com/DFC-OpenSource/qemu-ox.git
  编译运行，安装一个ubuntu16.04-desktop镜像,启动运行

> ./x86_64-softmmu/qemu-system-x86_64 -monitor stdio -m 4G -smp 4 -nographic -s -drive file=/home/lf/os-img/drive-file.bak,format=raw,media=disk -boot order=dc -device ox-ctrl,lnvm=1,debug=1,volt=1,serial=deadbeef -netdev user,net=172.16.122.0/24,hostfwd=tcp::8822-:22,id=net0 -device virtio-net-pci,netdev=net0 -serial pty --enable-kvm

-device ox-ctrl,lnvm=1,debug=1,volt=1,serial=deadbeef包含emulator主要逻辑，volt为使用DRAM对Flash进行模拟。

- nvm-cli升级https://github.com/linux-nvme/nvme-cli.git，系统自带的nvm-cli不包含lnvm插件，从GitHub下载并编译安装

  > ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996:~/test$ sudo nvme lnvm list
  > Number of devices: 1
  > Device Block manager Version
  > nvme0n1 gennvm (1,0,0)

- liglightnvm编译安装https://github.com/OpenChannelSSD/liblightnvm.git.

利用C-API访问NVMe设备

```c
#include <stdio.h>
#include <liblightnvm.h>
int main(int argc, char **argv)
{
    struct nvm_dev *dev = nvm_dev_open("/dev/nvme0n1");
    if (!dev) {
        perror("nvm_dev_open");
        return 1;
    }
    nvm_dev_pr(dev);
    nvm_dev_close(dev);

    return 0;
}
```

利用liblightnvm-tools进行设备访问,write_wm能够产生随机的数据写入，并且会覆盖OOB区域

> cd test/liblightnvm/build/cli/
> sudo nvm_addr write_wm /dev/nvme0n1 0x00000000003f0000 0x00000001003f0000 0x00000002003f0000 0x00000003003f0000
> sudo NVM_CLI_META_PR="1" nvm_addr read_wm /dev/nvme0n1 0x00000000003f0000 0x00000001003f0000 0x00000002003f0000 0x00000003003f0000 -v -o a.bin

- 编译运行pblk，可以在Linux-4.13.0-*的内核代码中的Driver/LightNVM中找到pblk的源码，修改Makefile,编译内核模块

```
#
# Makefile for Open-Channel SSDs.
#

KERNEL_VER ?= $(shell uname -r)
KERNEL_DIR:=/lib/modules/$(KERNEL_VER)/build
PWD:=$(shell pwd)

obj-m                           := pblk.o
pblk-objs                       := pblk-init.o pblk-core.o pblk-rb.o \
                                   pblk-write.o pblk-cache.o pblk-read.o
```

```
\
                                    pblk-gc.o pblk-recovery.o pblk-map.o
\
                                    pblk-rl.o pblk-sysfs.o

pblk:
        make -C $(KERNEL_DIR) M=$(PWD) modules
clean:
        make -C $(KERNEL_DIR) M=$(PWD) clean
        #rm -rf *.o *.mod.c *.ko *.symvers *.order *.unsi
```

Use pblk target as a standard FTL，配置块设备，挂载文件系统

> $ sudo nvme lnvm create -d nvme0n1 -t pblk -n nvme0n1_ox -b 0 -e 31
> Format and mount the block device:
> $ sudo fdisk /dev/nvme0n1_ox (if you want a partition table)
> $ sudo mkfs.ext3 /dev/nvme0n1_ox1
> $ sudo mount /dev/nvme0n1_ox1

思路扩展

- 模拟一个文件系统的攻击向量，使用pblk挂载EXT3文件系统，使用istat, blkcat等工具获取inode等数据块的信息。

```
root@lukas-VirtualBox:/home/lukas/sdb# istat /dev/sdb1 14
inode: 14
Allocated
Group: 0
Generation Id: 613173638
uid / gid: 0 / 0
mode: rrwsr-xr-x
size: 1037528
num of links: 1

Inode Times:
Accessed:   2018-05-09 21:28:23 (CST)
File Modified:  2018-05-09 21:26:54 (CST)
Inode Modified: 2018-05-09 21:27:01 (CST)

Indirect Blocks:
1033

Direct Blocks:
2048 2049 2050 2051 2052 2053 2054 2055
2056 2057 2058 2059 2060 2061 2062 2063
1072 1073 1074 1075 1076 1077 1078 1079
1080 1081 1082 1083 1084 1085 1086 1087
1568 1569 1570 1571 1572 1573 1574 1575
1576 1577 1578 1579 1580 1581 1582 1583
1584 1585 1586 1587 1588 1589 1590 1591
1592 1593 1594 1595 1596 1597 1598 1599
1120 1121 1122 1123 1124 1125 1126 1127
1128 1129 1130 1131 1132 1133 1134 1135
1136 1137 1138 1139 1140 1141 1142 1143
1144 1145 1146 1147 1148 1149 1150 1151
1152 1153 1154 1155 1156 1157 1158 1159
1160 1161 1162 1163 1164 1165 1166 1167
1168 1169 1170 1171 1172 1173 1174 1175
1176 1177 1178 1179 1180 1181 1182 1183
```

- 进行错误注入,借用模拟器的内存拷贝功能

```
static void volt_nand_dma (void *paddr, void *buf, size_t sz, uint8_t dir)
{
    switch (dir) {
        case VOLT_DMA_READ:
            memcpy(buf, paddr, sz);
            break;
        case VOLT_DMA_WRITE:
            memcpy(paddr, buf, sz);
            break;
    }
}
```

- ECC bypass讨论。要想实现利用的话，需要注入足够多的错误，跳到另外一个有效的码字的区域。

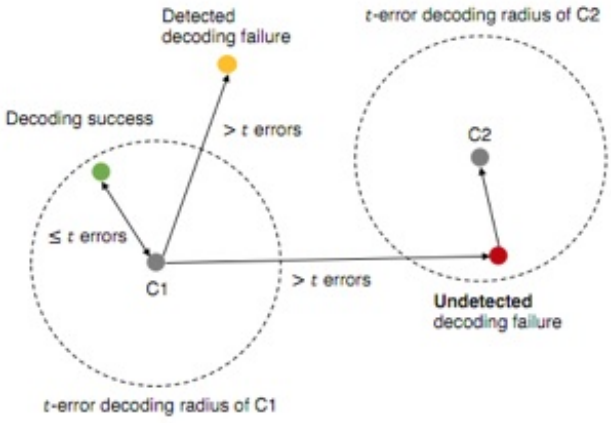

Figure 1: The three possible decoding events that can be induced by injecting errors into a Flash codeword (C1).

- 没有解决：如何通过获取pblk的映射信息？

- pblk官网文档提到如下借口，但是经过内核编译，没有成功利用；后考虑在读的时候使用printk打印信息，此工作还有待完成。

  ```
  echo “0-1023” > /sys/block/pblkdev/pblk/l2p_map
  ```

总结

经过本实验，了解了lightnvm架构，并对文件系统的文件组织有了一些了解，为下一步基于此进行研究和系统开发打下基础。