kaslr介绍 --*崔高生*

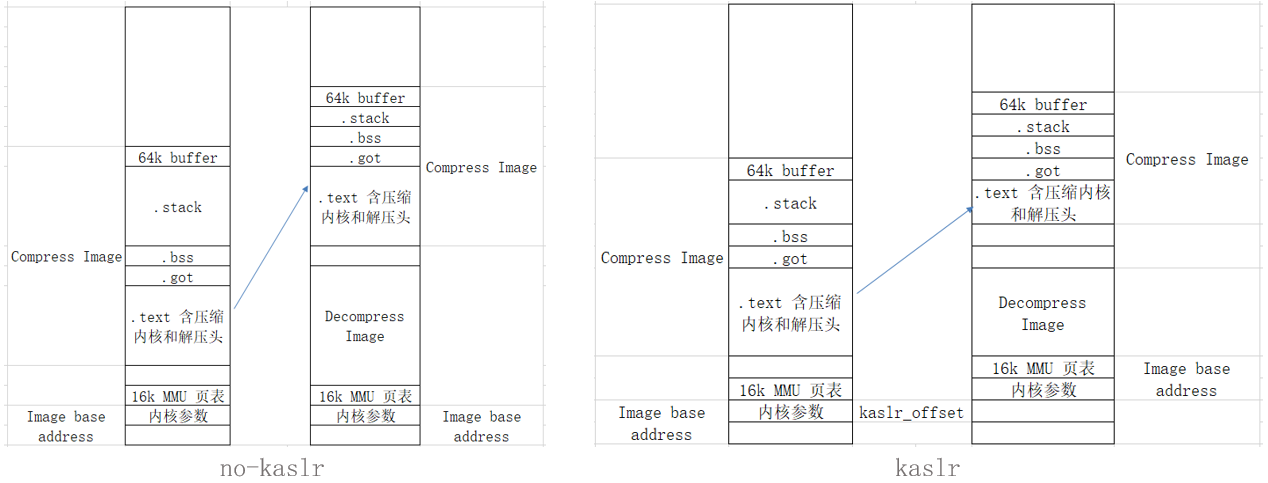
## 简介：

kaslr全称为kernel address space layout randomization，是linux内核的一个非常重要的安全机制，该机制可以让kernel image映射的执行地址相对于链接地址有个偏移，使内核符号变得随机，提升内核的安全性和防攻击能力。kaslr的实现原理的核心思想在于随机偏移，在内核启动阶段，通过获取一个随机的偏移值来校正内核加载地址，该偏移值既可以通过dtb传递，也可以基于随机源生成，在完成内核数据随机映射之后，还需要对符号地址进行重定位，校正内核代码的符号寻址，以此确保内核代码的正常执行。

## 二、kaslr实现原理

Linux内存的地址空间布局整体划分为两个部分：内核地址空间和用户地址空间，通常二者以1:3的比例划分，内核地址空间由内核代码分配使用，所有进程共享内核地址空间，而用户地址空间则为进程独有。

在内核启动加载时，内核镜像的加载、解压和运行均在内核地址空间完成，下图为ARM32架构在内核启动过程中的内核地址空间的布局转换，在内核地址空间中，整个内存区域分为两个部分：高端内存区域和低端内存区域，其中高端内存区域为非线性映射区，其存在的意义主要是进行虚拟地址空间向物理地址的动态映射和动态扩展，而低端内存区域为线性映射区，其映射原理是将物理地址与虚拟地址进行1:1映射，且地址偏移确定以后，线性映射区的映射关系就不会再发生变化（所以在汇编代码里面，通常就是通过物理地址与虚拟地址的偏移值确定线性区的地址映射关系）



Arm32受限于寻址范围，整个寻址空间为4G，依据内存地址空间模型，其线性映射区大小约为760MB，而非线性映射区仅有240MB，若将内核镜像加载在非线性映射区域，则其随机性会收到比较大的影响，因此在现有方案中，我们在线性映射区内进行镜像的加载和随机偏移。

Linux内核镜像是一个具有自解压功能的压缩文件，在启动过程中，会先启动一个booloader，再由bootloader将内核zImage，dtb，根文件系统加载到内存，然后跳转到zImage的第一条执行开始执行解压缩代码，将原始的内核数据解压到指定内存处，再跳转到内核代码入口进行内核初始化操作，bootloader的主要功能点如下：

* 找到并初始化内存；
* 初始化和使能一个串口输出，这个对于调试很重要；
* 获取CPU类型，指的是具体体现结构CPU的类型，比如ARM Cortex-x系列；
* 初始化kernel tagged list，向kernel传递系统内存和根文件系统位置和其它信息；
* 加载 initramfs (基于ram的一种fs)；
* 初始化设备树，跳转到kernel代码段入口stext；

kaslr的工作主要在内核自解压过程中完成，在内核启动阶段，kaslr功能代码会获取一个kernel image的偏移值，这个偏移值是由随机种子生产的伪随机数，随机种子既可以通过dtb或者bios传递设置，也可以基于硬件随机源或其他随机性对象生成。基于此再对整个可偏移内存区域进行region化管理，剔除其中不可进行内核镜像解压的内存区域，比如initrd、dtb、zImage以及保留内存已占用的内存区域，确保解压后的内核镜像不会与其他的系统资源发生内存互踩情况，影响OS正常启动，在剩余的空闲位置中，基于伪随机数生成kaslr\_offset，即内核随机偏移位置，然后将内核解压到该指定的内核随机偏移位置。

完成内核解压之后，还需要做一个内核符号重定位的工作，vmlinux在编译链接时，所有的内核符号地址及其调用关系均是预定义的，在完成内核镜像随机之后，原有的符号地址将失效，因此需要对内核符号进行重定位，从而保持内核代码的寻址调用关系，从而确保内核功能不受偏移影响。

## 三、kaslr实现步骤

kaslr内核构建阶段：

1. 链接选项中添加-fPIC，-PIE参数，将内核编译成地址无关的镜像
2. 生成重定位段.rel\*，用于内核符号重定位

kaslr解压缩阶段：

1. kaslr\_offset伪随机偏移值计算，解析cmdline，解析dts
2. 镜像解压到指定内存，解压、移动镜像

kaslr系统启动阶段：

1. kaslr\_offset记录内核偏移量，对内核符号进行地址重定位

## 四、kaslr特性使能和调试

* 使能条件

1. kaslr功能通过CONFIG\_RANDOMIZE\_BASE进行控制
2. cmdline中不能存在nokaslr参数，否则kaslr不被使能

* 随机种子

通过dts指定随机种子，如下：



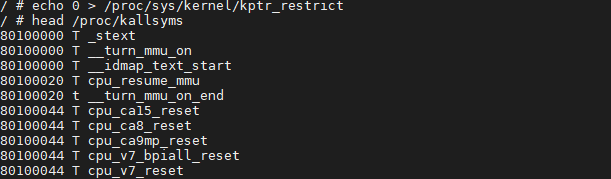
注：ARM会基于DTB文件做CRC运算，生成随机种子

* 验证测试

通过内核符号地址信息，可以观察内核符号加载状态，以此判断kaslr特性是否生效，命令如下：

echo 0 > /proc/sys/kernel/kptr\_restrict

head /proc/kallsyms



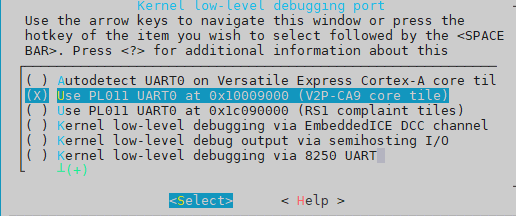
* 打印内存布局

不同的架构在kaslr实现过程中，都会预埋一些代码在DEBUG模式，用于进行内核功能调试，比如，可以通过config配置打印内存布局，以qemu vexpress-a9架构为例，其设置如下：

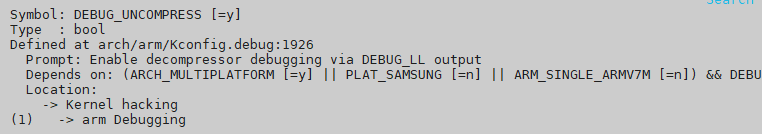
1. 打开Debug开关



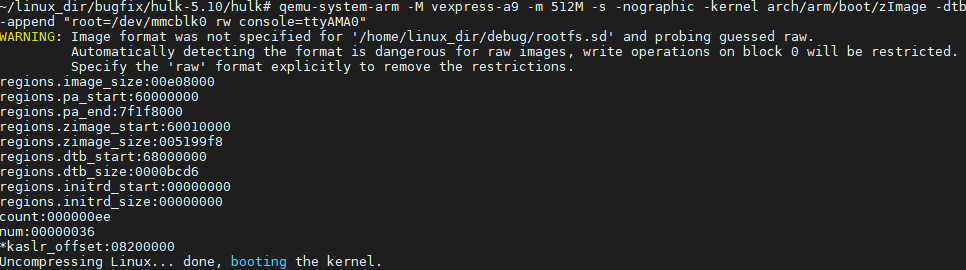
1. 设置串口输出



1. 调试解压缩代码



4、内存地址布局如下：



* Gdb调试内核

对比正常的内核启动方式，使能kaslr之后，内核符号信息发生变化，无法再基于编译生成的vmlinux进行内核断点调试，为解决这个问题，gdb支持动态符号表加载，即通过add-symbol-file动态加载vmlinux符号表，并可以根据实际的vmlinux加载偏移设置偏移量，如下：

arm-linux-gnueabi-gdb vmlinux 0x60010000

add-symbol-file arch/arm/boot/compressed/vmlinux 0x60010000

通过结合上述配置，即可完成内核kaslr功能的配置使用，并可以基于gdb深入研究使能kaslr后的内核功能。

## 五、kaslr in openEuler

* openEuler 21.09 kernel 支持ARM内核地址随机化

https://gitee.com/openeuler/kernel/blob/openEuler-21.09/arch/arm/boot/compressed/kaslr.c

* openEuler 21.09 kernel 支持ARM64、X86、PowerPC等架构的kaslr

## 六、kaslr开源社区实现

https://lwn.net/Articles/569635/