添加内核补丁

1.实验背景

补丁就是存放着不同版本之间差异的文件。通过打补丁的方法升级源代码，可以不用下载整个源代码，在本地的源代码基础上获得代码更新。
详细的打补丁的方法可以查看内核/Document/applying-patchs.txt

2.实验目的

给低版本内核打上补丁

3.实验原理

Linux 下安装高版本补丁到低版本的步骤大体分为以下几步：

1 选取补丁
（1）补丁出处：
理论上讲，各大开源网站(如 github，google 的 android 源码目录)的每次 commit 记录都可以作为补丁添加到我们的内核中，但是由于开源网站上分支众多、每个分支的代码更改次数也相当多，工作量巨大，所以我们需要针对性的寻找我们感兴趣并且比较重要的补丁。
网站 http://kernelnewbies.org/LinuxVersions 上针对每个 linux 内核版本之间的重大补丁做了总结，并且大部分都提供了补丁的详细介绍和验证方法，从这里选取补丁可以大大减少我们的工作量。除此之外，git 上的一些分支的代码可能 merge 到发布的内核中，在这些分支中则需要我们筛选找到我们感兴趣的代码。

（2）选取原则：
一般来说，打补丁的困难程度取决于以下因素：

- 补丁提交时间与我们的内核版本相近程度
- 改动的文件数量
- 改动的文件是否跨越了较多的模块
- 涉及的模块改动是否频繁
- 代码更改行数

2 制作补丁
选取好补丁之后，需要对补丁的代码进行分析，找出其他补丁中与之相关的代码。
一个高版本的补丁可能会牵扯到很多其它的补丁，一个比较小的补丁可能牵扯到很多其他的补丁，会造成代码量越改越多，所以我们需要分析代码之间的关系，将无关的代码剔除，保留只与本功能有关的代码。这一步是比较困难的一步，因为分析代码要对此模块有一定的了解，而且大多数情况都要进行这一步，所以，在选取补丁时最好选取自己熟悉的模块。
手动挑选出相关代码之后，将相关代码复制到源码相关处，因为版本和分支不同的原因，多数情况下文件的行数是无法对应的，所以需要手工查找对应代码。

3 编译内核

生成新的内核源码后进行编译、安装。

4 安装并验证补丁
完成新内核源码的编译后，安装好新的内核之后进入对补丁的验证阶段。
首先要确定更改后的代码是否生效。可以使用使用内核提供的 printk
打印日志的方式验证更改的代码确实已经执行。由于内核进行大量的打印日志可能会造成日志存储区溢出，如果要对打印速度进行限制可以选择 printk_ratelimit 函数。

4.实验知识点

1.制作补丁
diff
（1）对于单个文件
找出from-file与to-file差别，生成能够从 from-file到to-file升级的补丁
diff –uN from-file to-file > x.patch
（2）对于文件夹中多个文件
找出from-directory与to-directory差别，生成能够从 from-directory到to-directory升级的补丁
diff –uNr from-directory to-directory > x.patch

2.打补丁
patch
（1）对于单个文件
将补丁x应用到源文件src中，生成dst文件
patch src < x.patch -o dst
（2）对于文件夹中多个文件
将补丁x应用到源文件src中，src被修改
进入 src所在的目录
patch –p0 <x.patch
p0：当前文件夹

3.补丁内容说明
---：源文件
+++：目标文件
@@ -x,y +m,n @@：
源文件修改范围从第 x 行开始，共 y 行
修改之后对应的目标文件从 m 行开始，共 n 行
缩进：表示该部分进行修改
+ ：增加一行
-：减少一行
无符号：表示引用这一行，不进行增加或减少

5.实验流程

1.本次选择5.4版本内核的网络模块中对一个参数SOMAXCONN的patch，页面地址为https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=19f92a030ca6d772ab44b22ee6a01378a8cb32d4

关于该补丁的说明：

net: increase SOMAXCONN to 4096

```
SOMAXCONN is /proc/sys/net/core/somaxconn default value.

It has been defined as 128 more than 20 years ago.

Since it caps the listen() backlog values, the very small value has
caused numerous problems over the years, and many people had
to raise it on their hosts after beeing hit by problems.

Google has been using 1024 for at least 15 years, and we increased
this to 4096 after TCP listener rework has been completed, more than
4 years ago. We got no complain of this change breaking any
legacy application.

Many applications indeed setup a TCP listener with listen(fd, -1);
meaning they let the system select the backlog.

Raising SOMAXCONN lowers chance of the port being unavailable under
even small SYNFLOOD attack, and reduces possibilities of side channel
vulnerabilities.
```

这个参数主要限制了接收新 TCP 连接侦听队列的大小。对于一个经常处理新连接的高负载 web 服务环境来说，默认的 128 太小了。大多数环境这个值建议增加到 1024 或者更多，谷歌使用 1024 已经有 15 年了。有实验显示将这个数字改为 4096 后，超过 4 年时间没有出现一次由于这个改变导致传统应用崩溃的申诉。大的侦听队列对防止拒绝服务 DoS 攻击也会有所帮助。

首先，在旧版本内核下查看默认的 SOMAXCONN 参数值

cat /proc/sys/net/core/somaxconn

2.新建实验目录

mkdir -p mylinux-patch/include/linux

3.拷贝元目录

cp -a linux-5.3/include/linux/ mylinux-patch/include/linux/*

拷贝本次实验目录代码替换原文件

cp patch_src/socket.h mylinux-patch/include/linux/

4.利用 diff 命令生成补丁

diff -auNr -x '\.' linux-5.3/include/linux/ mylinux-patch/include/linux/ > mypatch*

查看 patch 内容

```
amos@ubuntu:~$ cat mypatch
diff -auNr -x '\.*' linux-5.3/include/linux/socket.h mylinux-patch/include/linux
/socket.h
--- linux-5.3/include/linux/socket.h            2019-02-15 16:09:54.000000000 +0
800
+++ mylinux-patch/include/linux/socket.h        2020-02-12 22:52:10.182936459 +0
800
@@ -262,7 +262,7 @@
 #define PF_MAX          AF_MAX

 /* Maximum queue length specifiable by listen.  */
-#define SOMAXCONN       128
+#define SOMAXCONN       4096

 /* Flags we can use with send/ and recv.
    Added those for 1003.1g not all are supported yet
```

5.用 patch 命令打补丁

cd linux-5.3/

patch -p1 < ../mypatch

6. 重新编译内核并安装（可以编译后使用 qemu 运行内核）

7. 重启进入新内核，查看新内核的 somaxconn 参数值

```
amos@ubuntu:~$ cat /proc/sys/net/core/somaxconn
4096
```

如果默认值已经变化为 4096，则实验成功。