江苏科技大学

课程实验报告

课　程：　无线网与移动终端安全技术

课　题：　Linux 环境下实现对称加密

学　院：　计算机学院

姓　名：　陈四贵

班　级：　1822107101

学　号：　182210710119

指导老师：　张迪明

目　　录

一、实验目的

1.掌握 linux 操作系统下 C++程序编写、编译和调试的基本方法;
2.掌握对称加密算法的原理及实现。

二、实验流程

1.安装 Firefox、Chrome、360 安全浏览器（非必需，但从兼容性考虑建议安装）；

2.配置浏览器网络代理;

> 1.地址：218.3.140.153
> 2.端口：23128

3. 登录实验平台;

> http://192.168.2.11:9000/。

4. 创建实验平台基础系统;

> 1.novnc 初始密码 password
> 2.系统管理员密码 resu

5. 熟悉 linux 系统下 C++编程语言的使用方法;

6. 实现一种对称加密算法。

三、基础验证

1. 安装 build-essential 编程环境;

```
sudo apt install build-essential
```

2. 选取一种对称加密算法;

> DES，AES…

3. 构建相应的对称加密算法源文件;

```
touch /tmp/demo_des.cpp
```

4. 编译和运行；

```
g++ -E filename.c -o filename.i
将 C 文件转化为 C++文件，这个过程也叫做预处理过程
g++ -S filename.i -o filename.s
将预处理过程生成的.i 后缀的文件转化为汇编文件，里面存储的是相应的汇编代码，这个过程叫做编译。
g++ -c filename.s -o filename.o
将汇编文件中的汇编代码翻译成相应的机器语言，这个过程叫做汇编。
g++ filename.o -o filename.exe
这条指令是完成链接这个过程的，它通过链接器 ld 将运行程序的目标文件和库文件链接在一起，生成最后的可执行文件
生成可执行文件后，我们就能够调用相应的程序了。
```

5. 观察结果。

四、进阶验证

问：思考对称加密算法的硬件加速实现（CPU 或 GPU）。

答：硬件加速就是利用硬件模块来代替软件算法以充分利用硬件所固有的快速特性（硬件加速通常比软件算法的效率要高），从而达到性能提升、成本优化的目的。在嵌入式系统中，也经常会用到版权硬件加密加速器，比较常见的有 AES、SHA、DES、3DES 等。当前，主要有以下两大加速方式：

(1)FPGA 现场可编程门阵列，可针对某个具体的软件算法进行定制化编程，譬如业内的智能网卡；

(2)ASIC 专用集成电路，它是面向专门用途的电路、专门为一个用户设计和制造的，譬如 Intel 的 QAT 卡仅支持特定加解密、压缩算法。

五、实验记录

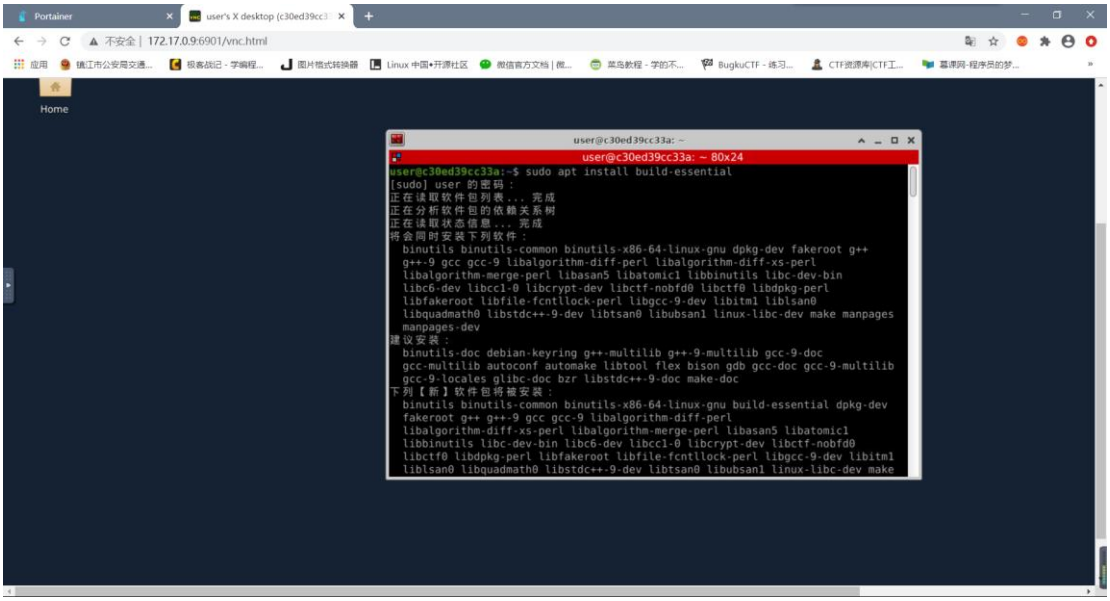
1.使用" sudo apt install build-essential"命令安装 build-essential 编程环境；

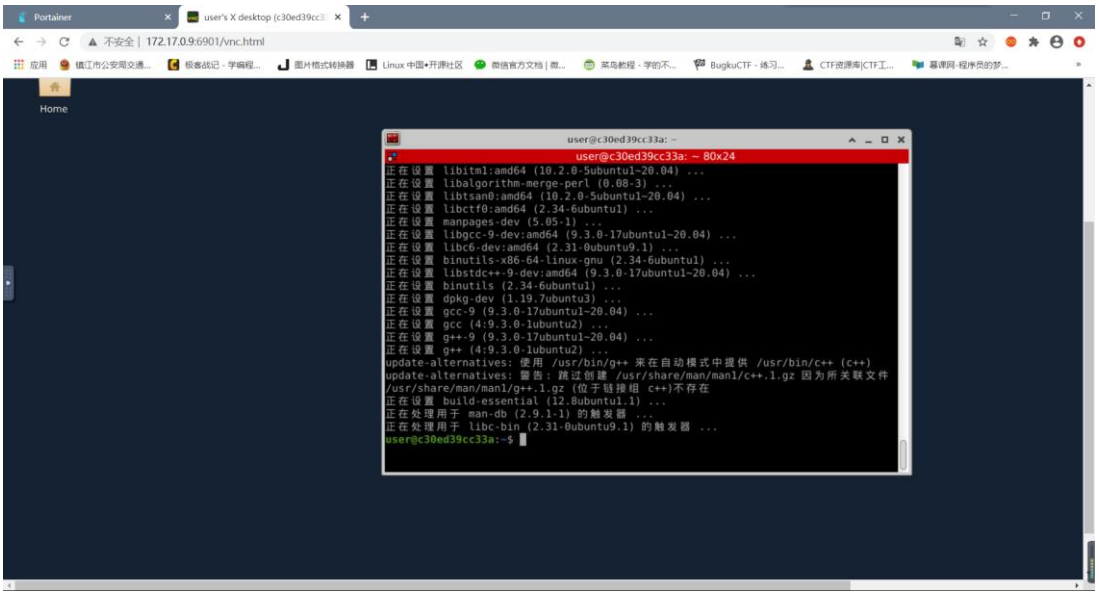
2.使用"touch /tmp/demo_ca.cpp"构建 cpp 文件，并通过"vi demo_ca.cpp"命令在 vim 编辑器编写相关算法的 c++实现代码；

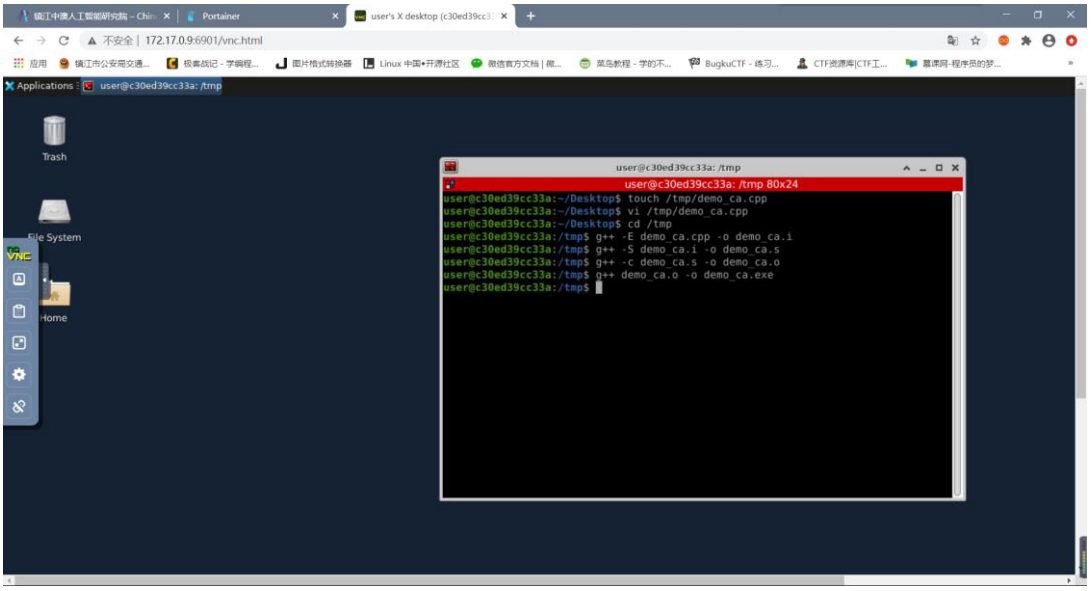
3.使用"基础验证"中相关命令依次生成.i、.s、.o、.exe 文件；

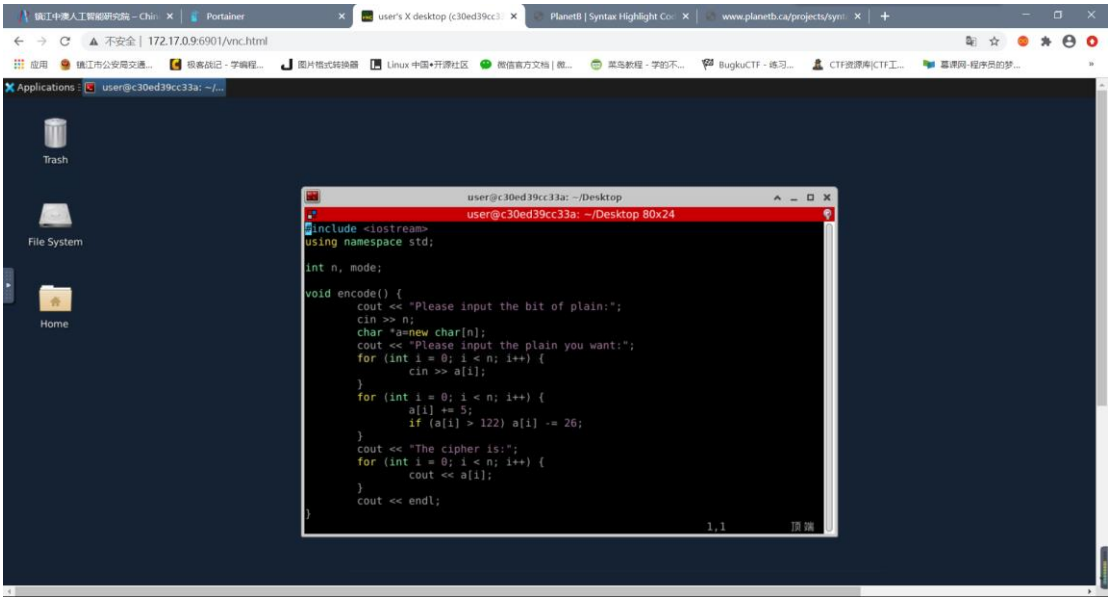
4.使用"g++ demo_ca.cpp"命令编译 cpp 文件，并使用"./a.out demo_ca.cpp"命令执行我们写的 cpp 文件。

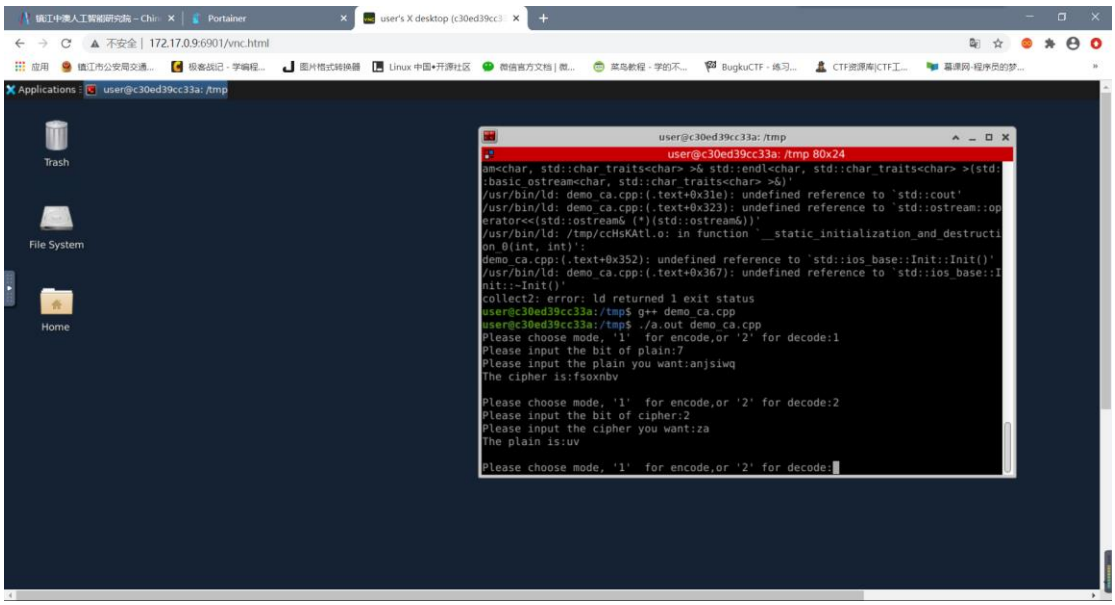
具体图示如下：


Portainer
user's X desktop (c30ed39cc3
不安全 | 172.17.0.9:6901/vnc.html
Home
user@c30ed39cc33a: ~
user@c30ed39cc33a: ~ 80x24
user@c30ed39cc33a:~$ sudo apt install build-essential
[sudo] user 的密码：
正在读取软件包列表 ... 完成
正在分析软件包的依赖关系树
正在读取状态信息 ... 完成
将会同时安装下列软件：
binutils binutils-common binutils-x86-64-linux-gnu dpkg-dev fakeroot g++
g++-9 gcc gcc-9 libalgorithm-diff-perl libalgorithm-diff-xs-perl
libalgorithm-merge-perl libasan5 libatomic1 libbinutils libc-dev-bin
libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdpkg-perl
libfakeroot libfile-fcntllock-perl libgcc-9-dev libitm1 liblsan0
libquadmath0 libstdc++-9-dev libtsan0 libubsan1 linux-libc-dev make manpages
manpages-dev
建议安装：
binutils-doc debian-keyring g++-multilib g++-9-multilib gcc-9-doc
gcc-multilib autoconf automake libtool flex bison gdb gcc-doc gcc-9-multilib
gcc-9-locales glibc-doc bzr libstdc++-9-doc make-doc
下列【新】软件包将被安装：
binutils binutils-common binutils-x86-64-linux-gnu build-essential dpkg-dev
fakeroot g++ g++-9 gcc gcc-9 libalgorithm-diff-perl
libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1
libbinutils libc-dev-bin libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0
libctf0 libdpkg-perl libfakeroot libfile-fcntllock-perl libgcc-9-dev libitm1
liblsan0 libquadmath0 libstdc++-9-dev libtsan0 libubsan1 linux-libc-dev make


Portainer
user's X desktop (c30ed39cc3
不安全 | 172.17.0.9:6901/vnc.html
Home
user@c30ed39cc33a: ~
user@c30ed39cc33a: ~ 80x24
正在设置 libitm1:amd64 (10.2.0-5ubuntu1~20.04) ...
正在设置 libalgorithm-merge-perl (0.08-3) ...
正在设置 libtsan0:amd64 (10.2.0-5ubuntu1~20.04) ...
正在设置 libctf0:amd64 (2.34-6ubuntu1) ...
正在设置 manpages-dev (5.05-1) ...
正在设置 libgcc-9-dev:amd64 (9.3.0-17ubuntu1~20.04) ...
正在设置 libc6-dev:amd64 (2.31-0ubuntu9.1) ...
正在设置 binutils-x86-64-linux-gnu (2.34-6ubuntu1) ...
正在设置 libstdc++-9-dev:amd64 (9.3.0-17ubuntu1~20.04) ...
正在设置 binutils (2.34-6ubuntu1) ...
正在设置 dpkg-dev (1.19.7ubuntu3) ...
正在设置 gcc-9 (9.3.0-17ubuntu1~20.04) ...
正在设置 gcc (4:9.3.0-1ubuntu2) ...
正在设置 g++-9 (9.3.0-17ubuntu1~20.04) ...
正在设置 g++ (4:9.3.0-1ubuntu2) ...
update-alternatives: 使用 /usr/bin/g++ 来在自动模式中提供 /usr/bin/c++ (c++)
update-alternatives: 警告：跳过创建 /usr/share/man/man1/c++.1.gz 因为所关联文件
/usr/share/man/man1/g++.1.gz (位于链接组 c++)不存在
正在设置 build-essential (12.8ubuntu1.1) ...
正在处理用于 man-db (2.9.1-1) 的触发器 ...
正在处理用于 libc-bin (2.31-0ubuntu9.1) 的触发器 ...
user@c30ed39cc33a:~$


不安全 | 172.17.0.9:6901/vnc.html
Applications
user@c30ed39cc33a: /tmp
Trash
File System
Home
user@c30ed39cc33a: /tmp
user@c30ed39cc33a: /tmp 80x24
user@c30ed39cc33a:~/Desktop$ touch /tmp/demo_ca.cpp
user@c30ed39cc33a:~/Desktop$ vi /tmp/demo_ca.cpp
user@c30ed39cc33a:~/Desktop$ cd /tmp
user@c30ed39cc33a:/tmp$ g++ -E demo_ca.cpp -o demo_ca.i
user@c30ed39cc33a:/tmp$ g++ -S demo_ca.i -o demo_ca.s
user@c30ed39cc33a:/tmp$ g++ -c demo_ca.s -o demo_ca.o
user@c30ed39cc33a:/tmp$ g++ demo_ca.o -o demo_ca.exe
user@c30ed39cc33a:/tmp$





六、实验小结

1.在使用 Linux 系统的终端命令行编程时，因为涉及到某一目录下多个文件的使用。所以最好使用 cd 命令将操作目录转到该目录下，避免不必要的冗杂代码；

2.在 linux 操作系统上开发程序光有 gcc 是不够的，还需要有 build-essential，作用是提供编译程序必须软件包的列表信息，即只有编译程序有了这个软件包，才能知道头文件位置，进而知晓库函数的位置；

3.linux 操作系统使用 wine 可以兼顾 windows 软件的优点，但 wine 对 win64 位的兼容和 win32 位的兼容存在些许问题，这是一个大坑。但是在编写程序时我们往往不需要一步步生成编译文件再去链接生成 exe 可执行文件，这对我们而言有些繁琐。此时我们可以通过"g++ demo_ca.cpp"命令编译 cpp 文件，再使用"./a.out demo_ca.cpp"命令执行我们写的 cpp 文件，借助已有的环境，避免繁琐的步骤；

4.在编写程序时，如果引用了较多的库，可能由于没有安装对应的库而导致编译失败。

七、实验源代码

```
#include <iostream>
using namespace std;

int n, mode;

void encode() {
    cout << "Please input the bit of plain:";
    cin >> n;
    char *a=new char[n];
    cout << "Please input the plain you want:";
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    for (int i = 0; i < n; i++) {
        a[i] += 5;
        if (a[i] > 122) a[i] -= 26;
    }
    cout << "The cipher is:";
    for (int i = 0; i < n; i++) {
        cout << a[i];
    }
    cout << endl;
}

void decode() {
    cout << "Please input the bit of cipher:";
    cin >> n;
    char *a = new char[n];
    cout << "Please input the cipher you want:";
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    for (int i = 0; i < n; i++) {
        a[i] -= 5;
        if (a[i] < 97) a[i] += 26;
    }
    cout << "The plain is:";
    for (int i = 0; i < n; i++) {
```

```
39.         cout << a[i];
40.     }
41.     cout << endl;
42. }
43.
44. int main() {
45.     do {
46.         cout << "Please choose mode, '1'  for encode,or '2' for decode:";
47.         cin >> mode;
48.         if (mode == 1) encode();
49.         else decode();
50.         cout << endl;
51.     } while (true);
52. }
```