MD5 Collision Attack Lab

1.简介：

安全散列函数要满足两个特性：单向性和抗碰撞特性。所谓单向性就是给定散列值找到输入M，使得hash（M）=h 是不可能的。抗碰撞特性是找到两个不同的M1和M2，使得hash(M1) = hash(M2)是不可行的。

2004年王晓云教授及其合作者给出了针对MD5的碰撞攻击。2017年2月google公司宣称针对SHA的攻击。本实验的目的是使学生了解散列函数的这些特性。

实验环境：ubuntu12.04 VM， 下载md5\_patch.zip文件（http://www.cis.syr.edu/~wedu/seed/Labs\_16.04/Crypto/Crypto\_MD5\_Collision/files/md5\_patch.zip），并运行脚本（patch.sh）来按照md5collgen程序。

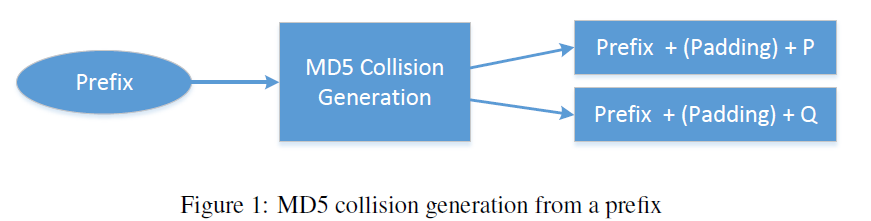
2.实验

2.1 任务1：产生两个MD5散列值相同的不同文件

本任务中，将产生两个带有相同md5值的不同文件。这两个文件的起始部分是相同的。我们可以使用md5collgen程序来实现。命令如下所示，输出为out1.bin和out2.bin.

$ md5collgen -p prefix.txt -o out1.bin out2.bin

产生这两个文件的道理如下图1所示。



可以使用如下命令来比较两个文件的差别，并检查md5散列值

$ diff out1.bin out2.bin

$ md5sum out1.bin

$ md5sum out2.bin

查看out1.bin和out2.bin的文件内容是否相同，可以使用虚拟机中的bless编辑器，并**回答一下问题：**

--Q1：如果前缀文件不是64的倍数，会发生什么？

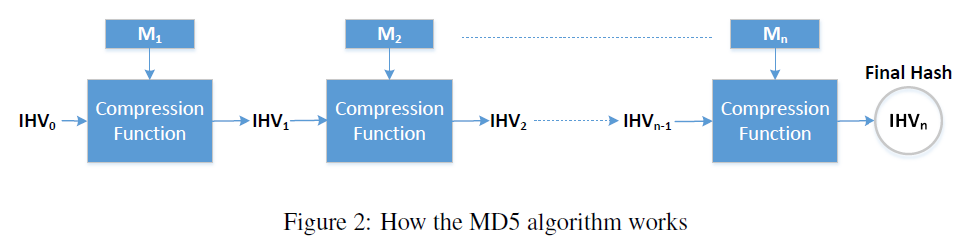
--Q2：如果前缀文件正好64个字节，重新运行md5collgen会发生什么？

--Q3：md5collgen生成两个输出文件的数据（128字节）完全不同吗？

--Q4：windows下的hex编辑器有哪些？

**2.2 任务2：理解MD5特性**

根据MD5的工作原理，如图2所示。



给定两个输入M和N，如果MD5(M) = MD5(N),即M和N的md5值相同，那么对于任何输入T，MD5(M || T) = MD5(N || T), 其中||表示连接操作。

也就是，如果输入M和T的散列值相同，在其后面加上后缀T则会产生两个具有相同散列值的输出。这一点不仅对MD5散列算法适用，对其他散列算法也是一样的。

可以用cat命令将两个文件合并为一个。下面的命令将file2和file1连接在一起，并将结果放在file3中。

$ cat file1 file2 > file3

2.3 **任务三：产生具有相同MD5值的两个可执行文件。**

本任务中，给定下面的C程序。你的任务就是创建一个程序的两个版本，使得数组xyz的内容不同，但是其可执行文件的散列值是相同的。

#include <stdio.h>

unsigned char xyz[200] = {

/\* 为xyz数组赋值\*/

};

int main()

{

int i;

for (i=0; i<200; i++){

printf("%x", xyz[i]);

}

printf("\n");

}

你可以修改源文件，产生上面的C程序的两个版本，编译之后，可执行文件有相同的MD5散列值。但是在二进制文件进行操作更容易。你可以将xyz数组内放入随机值，并将上述代码编译成二进制文件。为了在二进制文件中找到xyz数组，所以可将其填入固定的值，比如0x41，也就是字母A的ASCII码值。也就是将上面程序中的注释部分改成如下：

unsigned char xyz[200] = {

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

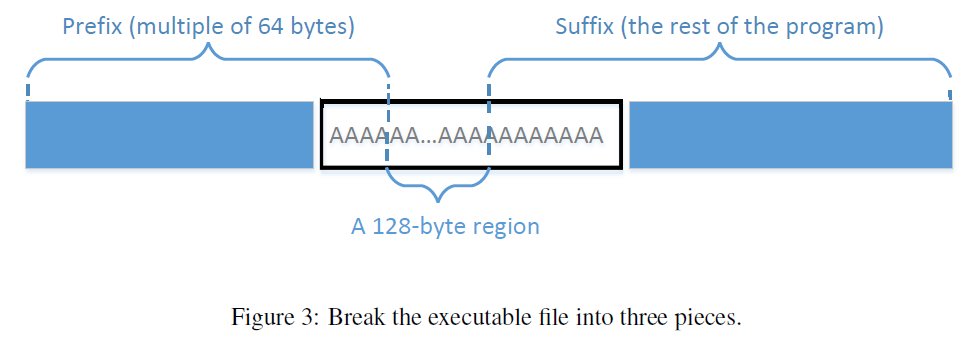
0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

... ...

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

}

在数组内部，我们可以找到两个位置，将这个可执行文件分成3个部分：前缀、128字节和后缀。前缀的长度需要是64位的整数倍。图3给出了文件分割的情形。



在前缀上运行md5collgen，产生两个具有相同md5散列值的输出。这里假定用P和Q来表示输出的第二个部分（即前缀后面的部分，长128字节）。所以有如下等式：

MD5 (prefix k P) = MD5 (prefix k Q)

根据MD5的特性，如果将相同的后缀添加到上述两个输出，那么产生的数据的散列值相同。下面的式子对任何后缀都是成立的：

MD5 (prefix k P k suffix) = MD5 (prefix k Q k suffix)

所以，我们只要使用P和Q来替换数组的128字节（两个分割点之间），就能产生两个二进制文件，其散列值相同。他们的输出是不同的，因为程序会打印出各自的数组。

工具：可以使用bless来查看二进制可执行文件，找到数组的位置。对于二进制文件分割，可以使用head和tail命令。比如下面的例子：

$ head -c 3200 a.out > prefix

$ tail -c 100 a.out > suffix

$ tail -c +3300 a.out > suffix

上面的命令将保存a.out的3200字节到prefix文件。第二个命令将a.out的后100个字节保存到suffix文件中。第三个命令将文件第3300字节到文件末尾的内容保存到a.out。可以使用这两个命令，将binary文件从任何位置分开。如果要将文件合并，则可以使用cat命令。

如果使用bless的copy-and-paste也可以将二进制文件的数据复制到其他文件，可以使用edit->select range菜单，可以选择数据，而不是手动要选择的字节数。

2.4 任务4：使得两个程序的行为不同

在前面的任务中，已经成功创建了两个具有相同md5值的两个程序，但是其行为是不同的。但是，其行为仅仅是打印的数据是不同的；他们仍然执行了相同的指令序列。本任务中，将再进一步。

假定你写了个软件，该软件是好的功能。你将该软件发送到一个受信任的权威机构。该权威机构对你的软件做了测试，并其认证你的软件功能是好的。权威机构将发给你证明，声明这个软件的功能是好的。为了防止你再获得证明后修改程序，你的程序的md5值页包含在文件中；这个证明被权威机构签发，所以你不能做任何改变，免得签名非法。

你还想要你的恶意软件也想获得权威机构的证明，但是你如果仅仅是把恶意程序交给权威机构，那么获得证明的机会为0.你有了个注意。你计划准备两个不同的程序。一个程序将执行好的程序，而另一个程序是恶意代码，你找到一种方法让这两个程序拥有相同的md5散列值。然后你把好的程序交给权威机构发证明。因为这个版本功能是好的，所以能通过验证，这时你有了一个证明，该证明中包含了你的好程序。因为你的恶意代码程序拥有相同的散列值，所以你的恶意程序也是合法的。所以你成功地为的恶意程序获得了证明。如果其他人相信权威机构颁发的证明，那么他们就会下载的恶意程序。

**你的任务是发动上面的攻击**。也就是，你需要创建两个程序，其md5散列值相同。但是一个程序执行好的代码，另一个程序执行恶意的指令。在你的工作中，执行什么样的好的或者坏的程序并不重要，只要能够说明两个程序执行的指令是不同的即可。

**指南**。在任务3的基础上，利用相同的前缀和md5collgen产生的两个散列值相同的程序，如果能在md5collgen产生的输出上增加有意义的后缀，那么添加到两个程序上的后缀也应该是相同的。下面提供一种方法作为参考。（注：方法很多，鼓励使用自己的想法）

创建两个数组X和Y，比较两个数组的内容；如果相同，则好的代码被执行；否则执行恶意代码部分。伪代码如下：

Array X;

Array Y;

main()

{

if(X’s contents and Y’s contents are the same)

run benign code;

else

run malicious code;

return;

}

可以使用固定的值（如0x41）来初始化X和Y，这样可以很容易在可执行文件中找到其位置。我们的任务就是改变这两个数组的内容，产生两个不同的版本，但其md5值两天。一个版本中X和Y内容相同；另个版本中X和Y的内容不同，执行恶意代码。这个过程和任务3中类似。图4给出了下面两个程序：

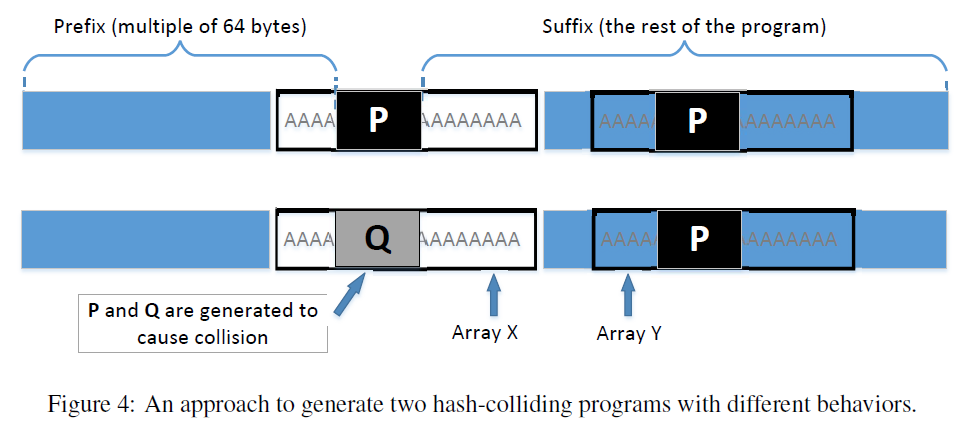


图4中，两个二进制文件的md5值相同，只要P和Q是按特定方法产生的。在第一个版本中，数组X和Y的内容相同，而第二个版本中其内容是不同的。所以要做的是要改变两个数组的内容，而不改变程序的逻辑。

**提交：**完成任务一中的几个问题，任务三和任务四。并使用文字说明和实验结果截图给出详细过程和实验总结。