Lab3 MD5碰撞实验

实验目的

- a) 使用md5collgen生成两个MD5值相同的文件,并利用bless十六进制编辑器查看输出的两个文件,描述你观察到的情况;
- b) 参考Lab3 task2.c的代码,生成两个MD5值相同但输出不同的两个可执行文件。
- c) 参考Lab3_task3.c的代码,生成两个MD5值相同但代码行为不相同的可执行文件。
- d) 回答问题: 通过上面的实验,请解释为什么可以做到不同行为的两个可执行文件具有相同的MD5值?

实验环境

Ubuntu20.04, gcc9.3.0

原理介绍

md5collgen

由前缀生成MD5碰撞,即返回两个md5值相同的文件,但是内容不完全相同,为前缀+128 字节填充

md5算法原理

MD5将输入数据划分为64个字节的块,然后在这些块上迭代计算散列。第一次迭代的IHV输入(IHV0)是一个固定的值。因此,将特定的suffix添加到具有相同 MD5 散列的任何两个不同消息中,通过连接原始消息和suffix消息,得到两个新的更长消息,这两个消息也具有相同的 MD5 散列。给定两个输入M,N如果MD5(M) = MD5(N),那么对于任何输入T,MD5(M T) = MD5(N | T)。

一些命令

cat

```
# 将多个文件连接为一个文件
cat file1 file2 > file3
# 将多个文件追加到已有文件后面
cat file4 file5 >> file3
```

head tail

```
# 将file1中前x个字节写入file2, -c表示读取的二进制文件
head -c x file1 > file2
# 将最后x个.....
tail -c x file1 > file2
# 从第x个字符开始
tail -c +x file1 > file2
```

注:地址是从0开始的。

实验过程

a)md5collgen生成两个前缀相同的文件

先创建一个文本文件test.txt

```
md5collgen -p test.txt -o out1.bin out2.bin
```

查看两文件的md5值:相同

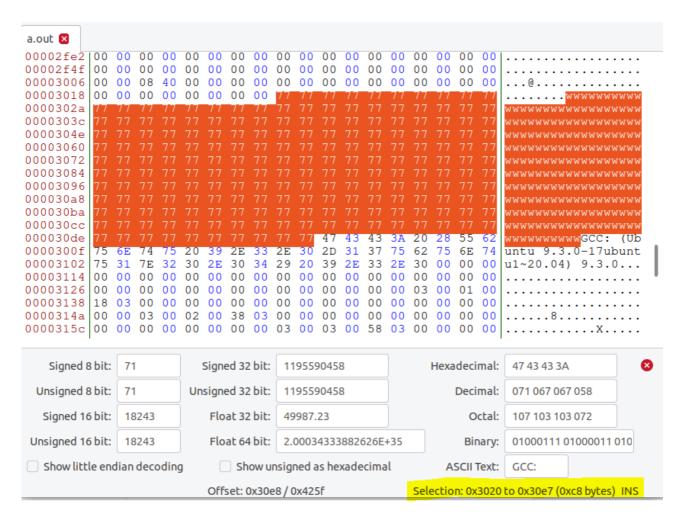
[10/10/22]seed@VM:~/.../share\$ md5sum out1.bin ea756735c677f4c0478f2519f83ac9e3 out1.bin [10/10/22]seed@VM:~/.../share\$ md5sum out2.bin ea756735c677f4c0478f2519f83ac9e3 out2.bin

查看十六进制:前缀相同但是填充内容有不同之处。

```
out1.bin 🛛
                               72 20 E5 85 B1 E4 BA AB E6 96 87|test for
00000000174 65 73
                  74 20 66 6F
00000012 E4 BB B6 E5 A4 B9 00
                              00 00 00 00 00 00 00 00 00 00 00
00000024 00 00 00 00 00 00
                           00
                              00
                                 00
                                    00 00 00 00 00 00
                                                       00 00 00
00000036 00 00 00 00 00 00
                              00 00
                                    00 10 E9 DB C1
                                                    56 CF CE CD
00000048 40 C8 4B 00 43 96 E6
                              AF A6 95 63 48 47 A7 E1 1E 2D 06
0000005a 37 0A B0 2D 38 C0 3D 09 FF D3 81 A2 45 B9 6F 04 4E 73
                                                                7...-8.=....E.o.Ns
0000006c D8 E7
              57
                  39
                        0D
                           3D
                              01
                                  71
                                     70
                                          66 54
                                                 5A CF
                                                          29
                     48
                                       ED
                                                       6D
                                                             19
                                                                 .. W9H. = . qp.fTZ.m).
0000007e CA 86 BC 7E 68 BA
                              0B A8
                                    06 65 A2 1B E0 25
                                                       D4 51 A9
                           63
                                                                ...~h.c...e...%.Q.
00000090 C6 16 81 26 28 F9 00
                              1E D5
                                    D7 14 C5 50 21 59 22 7F 67
                                                                 ...&(.....P!Y".g
000000a2 D7 FE 81 57
                     9F 02 95
                              64 A0
                                    8F EC
                                           92 5F C8 7F
                                                       C5 F3 CF
                                                                 ...W...d....
000000b4 BF A7 3C C3 DD A9 69
                                     56
                              6F BD
                                        63
                                           6C
                                                                 ..<...io.Vcl
out2.bin 🛛
00000000 | 74
               73 74 20 66
                           6F
                               72
                                  20
                                    E5 85 B1
                                              E4 BA AB E6 96 87
                                                                test for .....
00000012 E4 BB B6 E5 A4 B9 00 00 00 00 00 00 00 00 00 00 00
00000024 00 00 00 00 00 00
                              00 00 00 00 00 00 00 00 00 00 00
00000036
         00 00 00 00 00
                        0.0
                           00
                              00
                                 00
                                     00
                                       10 E9
                                              DB C1
                                                    56 CF
                                                          CE CD
                                        63 C8 47 A7 E1 1E 2D 06
00000048 40 C8 4B 00 43 96 E6 AF A6
                                    9.5
0000005a 37 0A B0 2D 38 C0 3D
                              09 FF D3 81 A2 45 B9 6F 04 4E 73
0000006c D8 67 58 39 48 0D 3D
                              01
                                 71
                                     70 ED 66 54 5A CF ED 29 19
                                                                .gX9H.=.qp.fTZ..).
0000007e CA 86 BC 7E 68 BA
                           63
                              0B A8
                                     06 65 A2 1B E0 25 D4 51 A9
                                                                ...~h.c...e...%.Q.
00000090 C6 16 81 A6 28 F9
                           00 1E D5 D7 14 C5 50 21 59 22 7F 67
                                                                ....(.....P!Y".g
000000a2 D7 FE 81 57 9F 02 95 64 A0 8F EC 12 5F C8 7F C5 F3 CF
                                                                ...W...d....
000000b4 BF A7 3C C3 DD A9 69 EF BD 56 63 6C
                                                                ..<...i..Vcl
```

b)生成两个MD5值相同但输出不同的两个可执行文件

填充数组,编译c文件生成a.out,使用bless查看,定位输出的数组:



输出数组范围在0x3020~0x30e7,转为十进制为12320~12519,选择12352(凑64的整数倍) 之前的作为prefix,中间改128字节,因此把12352+128=12480后面的截取出来作为suffix:

```
head -c 12352 a.out > prefix
tail -c +12480 a.out > suffix
```

根据 prefix 生成 md5 相同的两个文件

```
md5collgen -p prefix -o out1.bin out2.bin
```

取out1.bin out2.bin的后128字节:

```
tail -c 128 out1.bin > P
tail -c 128 out2.bin > Q
```

将三部分进行拼接,并对生成的文件赋予执行权限:

```
cat prefix P suffix > a1.out
cat prefix Q suffix > a2.out
chmod +x a1.out a2.out
```

执行a1.out a2.out输出结果如图,有不同之处:

查看两个执行文件的md5值,结果相同:

```
[10/10/22]seed@VM:~/.../share$ md5sum a1.out d38b8ed0a010bb93100584c1be5bdf5c a1.out [10/10/22]seed@VM:~/.../share$ md5sum a2.out d38b8ed0a010bb93100584c1be5bdf5c a2.out
```

c)生成两个MD5值相同但代码行为不相同的可执行文件

分别创建两个值相等的数组arr1,arr2,

```
unsigned char arr1[200] ={...};
unsigned char arr2[200] ={...};
int main(){
   if(!strcmp(arr1,arr2)){
      printf("in benign code\n");
   }else{
      printf("in malicious code");
   }
   return 0;
}
```

直接编译执行输出"benign code":

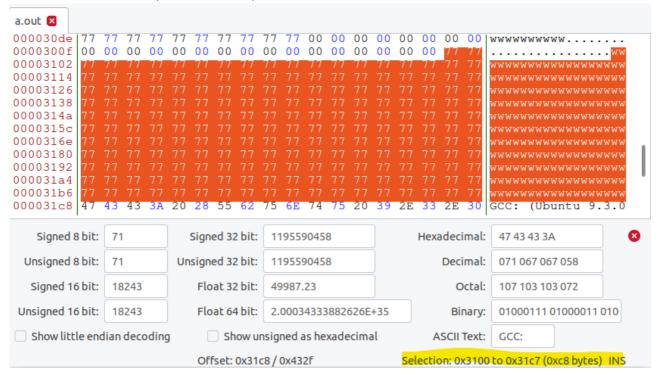
```
[10/10/22]seed@VM:~/.../share$ a.out in benign code
```

查看编译结果如下:

arr1:0x3020~0x30e7(12320~12519)

				•				,											
a.out 🛚																			
00003006	00	00	08	40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	@
00003018	00	00	00	00	00	00	00	00	77	77	77	77	77	77	77	77	77	77	wwwwwwwww
0000302a	77	77	77	77	77	77	77	77	77									77	wwwwwwwwwwww
0000303c	77																	77	wwwwwwwwwwww
0000304e	77																	77	wwwwwwwwwwww
00003060	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	wwwwwwwwwwww
00003072	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	wwwwwwwwwwww
00003084	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	wwwwwwwwwwww
00003096	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	WWWWWWWWWWWWW
000030a8	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	WWWWWWWWWWWWW
000030ba	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	WWWWWWWWWWWWW
000030cc	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	77	wwwwwwwwwwww
000030de	77	77	77	77	77	77	77	77	77	77	00	00	00	00	00	00	00	00	wwwwwwwww
0000300f	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	77	77	ww
Signed 8 bit:		0	0			Signed 32 bit:			0	0					Hexadecimal:			imal:	00 00 00 00
Unsigned 8 bit:		0	0			Unsigned 32 bit:			0	0						Decimal:			000 000 000 000
Signed 16 bit:		0	0			Float 32 bit:			0	0						Octal:			000 000 000 000
Unsigned 16 bit:		0	0			Float 64 bit:			0	0						Binary:			00000000 00000000 000
Show litt	le en	dian	deco	oding)	☐ Show unsigned as hexadecimal										Α	SCII	Text:	
						Off	set:	0x30	e8 / (0x43	2f				Sel	lectio	n: 0:	x3020	to 0x30e7 (0xc8 bytes) INS

arr2:0x3100~0x31c7(12544~12743)



arr1的位置与任务二相同,截取前12352作为前缀并生成两个md5相同的文件:

```
head -c 12352 a.out > prefix
md5collgen -p prefix -o out1.bin out2.bin
```

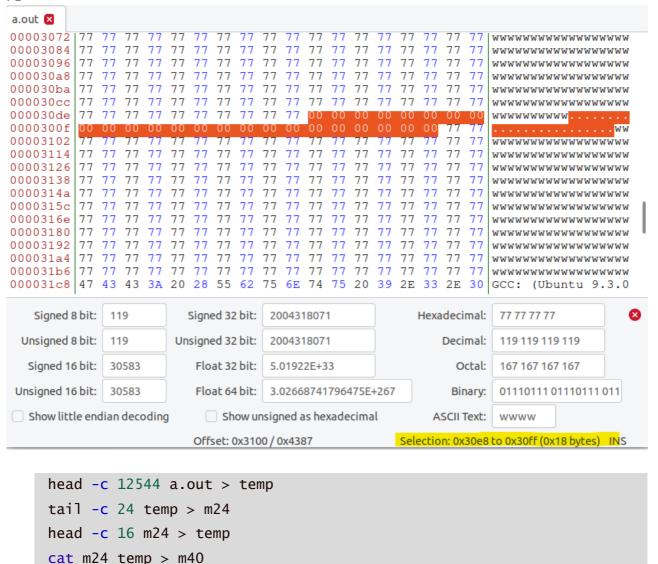
截取后缀(从GCC开始)

```
tail -c +12745 a.out > suffix
```

把其中一个新前缀p的后32(字符A)+128(md5填充物)=160个字节截取出来作为第二个数组的 主体:

```
tail -c 160 out1.bin > middle
```

获取填充字节0x00,由于数组200个字节,因此填充40个,注意两个数组间还有24字节的填充



开始拼接:

```
cat out1.bin m40 m24 middle m40 suffix > a1.out
cat out2.bin m40 m24 middle m40 suffix > a2.out
```

赋予权限并执行:

```
[10/10/22]seed@VM:~/.../share$ chmod +x a1.out a2.out
[10/10/22]seed@VM:~/.../share$ a1.out
in benign code
[10/10/22]seed@VM:~/.../share$ a2.out
in malicious code[10/10/22]seed@VM:~/.../share$ md5sum a1.out
f73a82ecb9467b90e4cdebb87df2a1e3 a1.out
[10/10/22]seed@VM:~/.../share$ md5sum a2.out
f73a82ecb9467b90e4cdebb87df2a1e3 a2.out
[10/10/22]seed@VM:~/.../share$
```

由图可得两个文件执行的函数不同但是md5值相同。

d)对c任务的解释

- 不同行为:在最后生成的可执行程序中,第二个数组与源程序无关,完全来自于第一个数组,因为middle取自out1.bin;因此经过填充后,两个文件中第二个数组的与其中一个相同而与(md5collgen产生的)另一个不相同导致if判断产生不一样的结果,最后执行不一样的函数。
- 相同md5: out1.bin与out2.bin是由md5collgen产生的具有相同md5值的不同 prefix文件,而两个文件后面的填充+middle+suffix完全相同,因此在迭代运算中 保持着相同的md5导致最后计算结果一样。

参考

MD5 Collision Attack Lab seed solution - SKPrimin - 博客园 (cnblogs.com)

从入门到入土: [SEED-Lab]MD5碰撞试验|MD5collgen实验|linux|Ubuntu|MD5 Collision Attack Lab|详细讲解 桃地睡不着的博客-CSDN博客

网络攻防技术——MD5碰撞试验啦啦啦啦啦啦啦噜噜的博客-CSDN博客md5碰撞