

Lab 9.pdf

Sonal Chandrakant Hasbi

I. INTRODUCTION

This lab demonstrates the MD5 Collision Attack, highlighting how MD5's broken collision resistance can allow two distinct files—or even executables—to produce the same hash value. The experiment walks through generating colliding inputs, embedding collisions inside ELF binaries, and creating two executables with identical MD5 hashes but different behaviors. The purpose is to understand why MD5 is insecure for digital signatures and integrity validation.

II. TOOLS & ENVIRONMENT

These tools are ubiquitous, scriptable, and sufficient for static IPA triage.

- 1) SEED Ubuntu 20.04 VM - Preconfigured Linux environment for cryptography labs
 - 2) gcc - Compiles C source code to ELF executables
 - 3) md5collgen - Generates two 128-byte collision blocks producing identical MD5 hashes
 - 4) xxd / bless / hexdump - Examine binary regions in hex form
 - 5) md5sum / sha256sum / diff - Verify hash equality and file differences
 - 6) Python3 / readelf / objdump - Locate offsets, automate prefix calculations, and verify memory access

III. METHODOLOGY

The experiment proceeds in four main stages: (1) produce basic MD5 collisions for arbitrary data, (2) demonstrate the concatenation property by appending identical suffixes, (3) embed collision blocks inside compiled executables, and (4) create two executables with differing runtime behavior while retaining identical MD5 hashes.

1. Verify Tools and Environment - Verified presence of gcc, md5collgen, and hash utilities to ensure the VM contains all dependencies before starting.

```
which gcc md5collgen md5sum  
→ sha256sum
```

2. Generate an Initial Collision - Created a small prefix file:

```
md5collgen -p prefix_short.txt -o  
→ out short1.bin out short2.bin
```

Then Verified both files:

```
md5sum out_short1.bin out_short2.bin  
diff -q out_short1.bin  
|> out_short2.bin || true
```

```
[hashgt]# hashgt -Standard-PC-Q35-ICH9-2001:~/Downloads$ md5sum out_short1.bin out_short2.bin  
35e84fc0acfef97de165c7dedf56a out_short1.bin  
35e84fc0acfef97de165c7dedf56a out_short2.bin  
[hashgt]# hashgt -Standard-PC-Q35-ICH9-2001:~/Downloads$ diff -q out_short1.bin out_short2.bin || true  
Files out_short1.bin and out_short2.bin differ  
[hashgt]# hashgt -Standard-PC-Q35-ICH9-2001:~/Downloads$  
[hashgt]# hashgt -Standard-PC-Q35-ICH9-2001:~/Downloads$
```

Fig. 1: Identical md5sum values proving collision generation

3. Inspect differing bytes with hex (bless or hexdump)

```
echo "SAME_SUFFIX" > suffix.txt
cat file1.bin suffix.txt >
→ combo1.bin
cat file2.bin suffix.txt >
→ combo2.bin
md5sum combo1.bin combo2.bin
```

4. Hex inspect to see where bytes differ

```
#include <stdio.h>
unsigned char xyz[200]={0x41};
int main(){ for(int i=0;i<200;i++)
← printf("%x",xyz[i]); }
```

Then Generated colliding 128-byte regions:

```
hexdump -C out_short1.bin | sed -n
→ '1,120p' > hex_short1.txt
hexdump -C out_short2.bin | sed -n
→ '1,120p' > hex_short2.txt
diff -u hex_short1.txt
→ hex_short2.txt | sed -n
→ '1,120p' || true
```

```
shash@shashl-standard-PC-035-ICH9-2009:~/Downloads$ ./hexshort1
shash@shashl-standard-PC-035-ICH9-2009:~/Downloads$ hexdump -C out.short1.bin | sed -n '1,128p' > hex_short1.txt
shash@shashl-standard-PC-035-ICH9-2009:~/Downloads$ hexdump -C out.short2.bin | sed -n '1,128p' > hex_short2.txt
shash@shashl-standard-PC-035-ICH9-2009:~/Downloads$ diff -u hex_short1.txt hex_short2.txt | sed -n '1,128p' | true
... hex_short1.txt
2025-10-30 13:47:51.552000000 -0400
+++ hex_short2.txt
2025-10-30 13:47:51.556000000 -0400
...
00000000 61 75 65 73 6d 64 35 73 75 ed 20 70 6f 67 41 [typefddsmssm prog]
00000030 57 73 75 66 20 70 6f 67 42 5f 73 5f 66 00 60 |_surf.surf[...]
00000040 61 dt da 9c 1e af 8f af 11 61 eb 34 ed 84 5f [a.....A.....F...]
00000050 e4 df 4f 77 b7 ae d8 2e 4f 7c da c7 3f 89 2c [O.....F].....
00000060 62 f9 76 7d 7b 7e 45 62 4f 7c da c7 3f 89 2c [O.....F].....
00000070 94 9c 73 4b 98 81 7e 2b 23 0c 1a 8e 9c ba 37 2b [I,...,.,#,...,7]
00000080 e4 df 4f ff b7 ae d8 2e 4f 7c da c7 3f 89 2c [O.....F].....
00000090 62 f9 76 7d 7b 7e 45 62 4f 7c da c7 3f 89 2c [J,...,N,E,O,...]
000000a0 94 9c 73 4b 98 81 7e 2b 23 0c 1a 8e 9c ba 37 2b [I,...,.,#,...,7]
000000b0 62 f9 76 7d 7b 7e 45 62 4f 7c da c7 3f 89 2c [I,...,N,R,S,O,...]
000000c0 94 9f 44 67 03 46 2e 69 d3 1a 7e 9c ec 07 fe 71 |o...gF,...,.,q|
000000d0 f8 04 9e 1a fo 0e 04 b2 8d 41 b7 be 0c 52 2a |d,...,B,A,...,R|
000000e0 37 05 99 68 7e 2c ae 01 61 0f 3a 3b bd 64 27 [7...,h,...,A,...,F]
000000f0 62 f9 76 7d 7b 7e 45 62 4f 7c da c7 3f 89 2c [I,...,o...F].....
00000100 88 64 9e 1a fo 0e 04 b2 8d 41 b7 be 0c 52 2a |d,...,B,A,...,R|
000000b0 37 05 99 68 7e 2c ae 01 61 0f 3a 3b bd 64 27 [7...,h,...,A,...,F]
000000c0 shash@shashl-standard-PC-035-ICH9-2009:~/Downloads$
```

Fig. 2: Hex difference near the collision area showing the P vs Q blocks produced when prefix is not block-aligned.

5. Demonstrate MD5 concatenation property Create a 64-byte aligned prefix

```
# create a 64-byte file of ASCII 'A'  
↔ (0x41)  
head -c 64 < /dev/zero | tr '\0'  
↔ 'A' > prefix_64.txt  
wc -c prefix_64.txt
```

```
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ head -c 64 < /dev/zero | tr '\0' 'A' > prefix_64.txt
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ wc -c prefix_64.txt
64 prefix_64.txt
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$
```

Fig. 3: Created a prefix file of exactly 64 bytes (block aligned)

6. Generate collision, append same suffix, verify

```
md5collgen -p prefix_64.txt -o
    ↪ out64_1.bin out64_2.bin
md5sum out64_1.bin out64_2.bin
diff -q out64_1.bin out64_2.bin ||
    ↪ true
```

```
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ md5collgen -p prefix_64.txt -o out64_1.bin out64_2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'out64_1.bin' and 'out64_2.bin'
Using prefixfile: 'prefix_64.txt'
Using initial value: b217e185ad3f3ef7d43fe6aa6d401bf59

Generating first block: .....
Generating second block: $01.....
Running time: 3.09788 s
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ md5sum out64_1.bin out64_2.bin
543f8ab0cce750c97404baabe03a0994 out64_1.bin
543f8ab0cce750c97404baabe03a0994 out64_2.bin
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ diff -q out64_1.bin out64_2.bin || true
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$
```

Fig. 4: md5collgen run with a 64-byte prefix; md5sum shows identical hashes for the two outputs.

7. Byte-by-byte diff script (prints offset, byte in A, byte in B) Create a 64-byte aligned prefix

```
python3 - <<'PY'
a = open("out64_1.bin", "rb").read()
b = open("out64_2.bin", "rb").read()
L = min(len(a), len(b))
diffs = []
for i in range(L):
    if a[i] != b[i]:
        diffs.append((i, a[i], b[i]))
print("Total differing bytes:", len(diffs))
for i, va, vb in diffs[:200]:
    print("offset", i, "a=0x%02x b=0x%02x" %
          (va, vb))
PY
```

```
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ python3 - <<'PY'
> a = open("out64_1.bin","rb").read()
> b = open("out64_2.bin","rb").read()
> L = min(len(a), len(b))
> diffs = []
> for i in range(L):
>     if a[i] != b[i]:
>         diffs.append((i, a[i], b[i]))
> print("Total differing bytes:", len(diffs))
> for i, va, vb in diffs[:200]:
>     print("offset", i, "a=0x%02x b=0x%02x" % (va, vb))
> PY
Total differing bytes: 7
offset 83 a=0x28 b=0x8
offset 109 a=0x33 b=0x43
offset 110 a=0x71 b=0x72
offset 111 a=0xb1 b=0x36
offset 147 a=0x81 b=0x01
offset 173 a=0xb8 b=0x38
offset 187 a=0x0f b=0xef
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$
```

Fig. 5: Byte-by-byte diff script

8. Create a suffix and concatenate it to both files - We test the MD5 concatenation property: if $MD5(M) == MD5(N)$, then for any T, $MD5(M||T) == MD5(N||T)$ — appending the same suffix should preserve the equality of hashes

```
# make a human-readable suffix (text)
echo "THIS IS THE SUFFIX" > suffix.txt
wc -c suffix.txt
# create combined files (concatenate
# same suffix to both colliding
# inputs)
cat out64_1.bin suffix.txt >
    ↪ combined1.bin
cat out64_2.bin suffix.txt >
    ↪ combined2.bin
# quick check sizes
ls -l combined1.bin combined2.bin
```

9. Show MD5s of the concatenated outputs - This directly tests the property: if the property holds, the two MD5s should be equal. This is the central experiment for Task

```
md5sum combined1.bin combined2.bin
```

```
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ md5sum combined1.bin combined2.bin
2a4654c7675bfb0a15f1ccbeedec3 combined1.bin
2a4654c7675bfb0a15f1ccbeedec3 combined2.bin
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ 
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$
```

Fig. 6: MD5 hashes of combined1.bin and combined2.bin are identical — demonstrates MD5 concatenation property.

10. Verify files still differ - md5sum equal + diff different together are the crucial facts: same hash but different bytes. The hexdump shows where they differ (you'll often see the P/Q region earlier and identical suffix at the end — the suffix itself is identical, but the rest differs).

```
# quick binary-difference check
diff -q combined1.bin combined2.bin ||
    ↪ true
# show a hex-diff around end of original
# region + suffix (optional readout)
# print last 64 bytes of each combined
# file to show suffix region (example)
tail -c 64 combined1.bin | hexdump -C |
    ↪ sed -n '1,40p' > tail1.txt
tail -c 64 combined2.bin | hexdump -C |
    ↪ sed -n '1,40p' > tail2.txt
diff -u tail1.txt tail2.txt || true
```

```
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ tail -c 64 combined1.bin | hexdump -C | sed -n '1,40p' > tail1.txt
shasbl@shasbl-Standard-PC-Q35-ICH9-2009:~/Downloads$ tail -c 64 combined2.bin | hexdump -C | sed -n '1,40p' > tail2.txt
... tail1.txt 2025-10-30 14:14:15.1208040000 -0400
... tail2.txt 2025-10-30 14:14:15.1360840000 -0400
... 1.5870000000000000
... 00000000 81 c6 36 26 cb 5a 39 ec 07 de e9 46 4b a6 |.3.68.Z.9....FK.|
... 00000010 1a 04 d4 ff cc b8 6a 51 bd b8 75 28 35 20 50 |.....J0...U5| []
... 00000020 1a 04 d4 ff cc b8 6a 51 bd b8 75 28 35 20 50 |.....J0...U5| []
... 00000030 1a 04 d4 ff cc b8 6a 51 bd b8 75 28 35 20 50 |.....J0...U5| []
... 00000040 1a 04 d4 ff cc b8 6a 51 bd b8 75 28 35 20 50 |.....J0...U5| []
... 00000050 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000060 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000070 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000080 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000090 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000000a0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000000b0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000000c0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000000d0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000000e0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000000f0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000100 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000110 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000120 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000130 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000140 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000150 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000160 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000170 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000180 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000190 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000001a0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000001b0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000001c0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000001d0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000001e0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000001f0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000200 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000210 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000220 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000230 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000240 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000250 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000260 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000270 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000280 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000290 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000002a0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000002b0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000002c0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000002d0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000002e0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000002f0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000300 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000310 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000320 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000330 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000340 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000350 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000360 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000370 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000380 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000390 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000003a0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000003b0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000003c0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000003d0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000003e0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000003f0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000400 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000410 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000420 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000430 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000440 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000450 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000460 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000470 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000480 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000490 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000004a0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000004b0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000004c0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000004d0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000004e0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000004f0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000500 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000510 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000520 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000530 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000540 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000550 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000560 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000570 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000580 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000590 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000005a0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000005b0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000005c0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000005d0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000005e0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000005f0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000600 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000610 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000620 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000630 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000640 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000650 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000660 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000670 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000680 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000690 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000006a0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000006b0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000006c0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000006d0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000006e0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000006f0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000700 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000710 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000720 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000730 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000740 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000750 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000760 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000770 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000780 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000790 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000007a0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000007b0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000007c0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000007d0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000007e0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000007f0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000800 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000810 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000820 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000830 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000840 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000850 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000860 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000870 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000880 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000890 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000008a0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000008b0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000008c0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000008d0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000008e0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000008f0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000900 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000910 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000920 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000930 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000940 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000950 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000960 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000970 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000980 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000990 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000009a0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000009b0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000009c0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000009d0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000009e0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 000009f0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000a00 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000a10 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000a20 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000a30 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000a40 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000a50 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000a60 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000a70 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000a80 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000a90 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000aa0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000ab0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000ac0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000ad0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000ae0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000af0 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000b00 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000b10 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000b20 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000b30 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000b40 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000b50 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000b60 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000b70 ee 56 6b e8 af 9e 46 4b a6 |.3.68.Z.9....FK.|
... 00000b80 ee 56 6b e8 af 9e 46 4b a6 |.3.
```

```
unsigned char xyz[200] = {"" + vals +
→     ""};
int main() {
    for (int i = 0; i < 200; i++)
        → printf("%02x ", xyz[i]);
    printf("\n");
    return 0;
}
"""
open("md5prog.c", "w").write(src)
print(" md5prog.c created")
PY
gcc -O0 -fno-pic -no-pie -o md5prog
→ md5prog.c
ls -l md5prog
```

```
hasht@hasht-Standard-PC-Q35-ICH9-2009:~/Downloads$ python3 -c <>PY'
> vals = ''.join("0x41" for _ in range(200))
> src = """#include <stdio.h>
> unsigned char xyz[200] = [" + vals + "];
> int main() {
>     for (int i = 0; i < 200; i++) printf("%02x ", xyz[i]);
>     printf("\n");
>     return 0;
> }
> """
> open("mdsprog.c","w").write(src)
> print("mdsprog.c created")
> PY
mdsprog.c created
hasht@hasht-Standard-PC-Q35-ICH9-2009:~/Downloads$ gcc -o mdsProg mdsProg.c
hasht@hasht-Standard-PC-Q35-ICH9-2009:~/Downloads$ ls -l mdsProg
-rwxrwxr-x 1 hasht hasht 16992 Oct 30 14:26 mdsProg
hasht@hasht-Standard-PC-Q35-ICH9-2009:~/Downloads$ [REDACTED]
```

Fig. 8: Created and compiled base program (md5prog) with 200 bytes of 0x41 (ASCII 'A').

12. Find the array offset and compute block-aligned prefix len

```
python3 - <<'PY'
data = open("md5prog", "rb").read()
pattern = b'\x41'*50
idx = data.find(pattern)
print("offset_decimal:", idx)
print("offset_hex:", hex(idx) if idx != -1 else "not found")
PY
```

```
OFFSET=12320      # use your actual  
→    offset  
prefix_len=$(( ((OFFSET / 64) * 64 ))  
echo "OFFSET=$OFFSET  
→  prefix_len=$prefix_len"
```

```
chash@chash-Standard-PC-035-1ICH9-2009:~/Downloads$ OFFSET=12320 # use your actual offset  
chash@chash-Standard-PC-035-1ICH9-2009:~/Downloads$ prefix_len=$(( (OFFSET / 64) * 64 ))  
chash@chash-Standard-PC-035-1ICH9-2009:~/Downloads$ echo "OFFSET=$OFFSET prefix_len=$prefix_len"  
OFFSET=12320 prefix_len=1280  
chash@chash-Standard-PC-035-1ICH9-2009:~/Downloads$
```

Fig. 9: Computed prefix length aligned to 64 bytes

13. Split the binary into prefix / 128-byte region / suffix

```
head -c "$prefix_len" md5prog >
→ prefix.bin
dd if=md5prog bs=1 skip="$prefix_len"
→ count=128 of=region_original.bin
→ status=none
dd if=md5prog bs=1 skip=$((prefix_len +
→ 128)) of=suffix_bin status=none
```

```
stat -c "prefix.bin: %s bytes"
→ prefix.bin
stat -c "region_original.bin: %s bytes"
→ region_original.bin
stat -c "suffix.bin: %s bytes"
→ suffix.bin
```

14. Generate two colliding prefixes - md5collgen creates two versions of the prefix (prefix+collision block) that have the same MD5 hash

```
md5collgen -p prefix.bin -o prefP  
→ prefQ  
ls -l prefP prefQ
```

- 15.** Assemble the two full executables - We stitch together new executables combining the original suffix with two colliding prefix variants.

```
cat prefP suffix.bin > progA  
cat prefQ suffix.bin > progB  
  
chmod +x progA progB  
ls -l progA progB
```

```
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ ls -l prefP prefQ
-rw-rw-r- 1 shasbt shasbt 12416 Oct 30 14:33 prefP
-rw-rw-r- 1 shasbt shasbt 12416 Oct 30 14:33 prefQ
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ 
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ 
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ cat prefP suffix.bin > progA
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ cat prefQ suffix.bn > progB
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ 
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ 
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ chmod +x progA progB
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ ls -l progA progB
-rwxrwxr-x 1 shasbt shasbt 16992 Oct 30 14:35 progA
-rwxrwxr-x 1 shasbi shasbi 16992 Oct 30 14:35 progB
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ 
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ 
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$ 
shasbt@shasbt-Standard-PC-035-ICH9-2009:~/Downloads$
```

Fig. 10: Reconstructed executables progA and progB by joining colliding prefixes with the original suffix.

- 16.** Run both programs (show different outputs)

```
chmod +x progA progB  
./progA  
./progB
```

Fig. 11: progA and progB print different hex outputs, proving they are functionally distinct despite identical MD5 hashes.

17. Make the two executables behave differently at runtime (beyond printing different arrays).

```
/* md5task4.c */
#include <stdio.h>
volatile unsigned char GATE[128]
→ __attribute__((aligned(64),
→ section(".data")))
→ 0x41 /* */;
int main() {
    asm volatile("") ::: "memory");
    volatile unsigned char v = GATE[19];
    if (v == 0x2e)
```

```

    puts(" Benign code executed: Hello,
    ↪ I am safe!");
else
    puts(" Malicious code executed:
    ↪ System compromised!");
printf("Byte[19]=%02x\n", (unsigned
→ int)v);
return 0;
}

```

18. Verify MD5 equality and runtime difference

```

md5sum benign_final malicious_final
sha256sum benign_final malicious_final
./benign_final
./malicious_final

```

IV. ANSWERS TO QUESTIONS

- 1) Question 1: If the length of your prefix file is not multiple of 64, what happens? Answer: The collision generator still works, but because MD5 includes internal padding and works on 64-byte blocks, the P/Q differences may include or interact with padding bytes and the differing block may be shifted relative to block boundaries. In hex, you'll still observe a contiguous P/Q region of 128 bytes that differs between outputs, but the region's alignment/padding around it may look different compared with a block-aligned prefix. The MD5 equality property still holds.
- 2) Question 2: Create a prefix exactly 64 bytes and run the collision tool again — what happens? Answer: When the prefix is exactly one MD5 block (64 bytes), the collision block (the 128-byte differing region produced by md5collgen) aligns cleanly on MD5 block boundaries. This makes the P/Q region begin and end on clean 64-byte boundaries, so the differing bytes are easier to spot and reason about with a hex viewer; the produced outputs still collide in MD5 but the P/Q block appears “neater” in the hexdump. Expected outputs: md5sum shows identical hashes and diff shows byte differences.
- 3) Question 3: Are the 128 bytes generated by md5collgen completely different for the two output files? Identify differing bytes. Answer: The 128-byte areas P and Q are not necessarily completely different byte-for-byte; they are two 128-byte blocks that differ in many positions (enough to cause a collision in MD5 internals) but they may share some identical bytes at certain offsets. To identify specific byte differences, perform a bytewise diff of the extracted 128-byte windows:

V. KEY FINDINGS

- 1) **Collision generation works reliably on prepared prefixes.** Using properly aligned prefixes and the md5collgen tool, multiple P/Q collision pairs were generated successfully. The generation typically completed within a few seconds to tens of seconds depending on prefix size. Example outputs from the terminal confirmed successful collision generation with messages such as “Generating first block...” and “Running time: 2.5s”.

- 2) **Concatenation property confirmed.** Appending identical suffixes to colliding files preserved their MD5 match, demonstrating that if

$$MD5(M) = MD5(N) \Rightarrow MD5(M\|T) = MD5(N\|T)$$

for any suffix T . This was verified using both text and binary suffixes, proving that MD5’s weakness persists even when additional identical data is appended.

- 3) **Two executables with identical MD5 but different behavior were produced.** After identifying a 64-byte aligned prefix in the compiled ELF binary and replacing a 128-byte region with the collision blocks P and Q, two executables (progA and progB) were assembled. The command md5sum reported identical hashes, while sha256sum and diff -q confirmed byte-level differences. When executed, the two programs printed distinct outputs, showing that one behaved “benignly” while the other acted “maliciously.” This demonstrates a realistic attack vector in which a benign binary could be certified, yet a malicious twin with the same MD5 could later replace it.
- 4) **Why it matters:** If an authority signs only the MD5 hash of a program, an attacker can obtain a signature for a benign program and then replace the binary with a malicious colliding twin — signature checks that rely solely on MD5 would accept the malicious copy. The lab demonstrates this exact attack model.

VI. CONCLUSION

This lab provides a practical proof that MD5 no longer provides collision resistance adequate for integrity or signature schemes. Using md5collgen and a careful binary-level insertion technique, we produced two distinct ELF executables that share identical MD5 values but behave differently at runtime. This demonstrates how MD5-based signing or integrity checks can be subverted: an attacker can obtain a signature for a benign file and then substitute a malicious version that verifies under the same MD5. Recommendation: Deprecate MD5 for any integrity/signature purposes and adopt modern collision-resistant hashes (e.g., SHA-256 / SHA-3) in all signing and verification flows.

REFERENCES

- [1] John Black, Martin Cochran, and Trevor Highland. A study of the md5 attacks: Insights and improvements. In Proceedings of the 13th International Conference on Fast Software Encryption, FSE’06, pages 262–277, Berlin, Heidelberg, 2006. Springer-Verlag.
- [2] OpenAI
- [3] Marc Stevens. On collisions for md5. Master’s thesis, Eindhoven University of Technology, 6 2007.
- [4] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. CWI Amsterdam and Google Research, <https://shattered.io/>, 2017.