github/truerandom

1. Obtén el valor del hash MD5 de los ejecutables "hello" y "erase"
que se encuentran en el directorio prog_linux

md5sum hello
md5sum erase

```
user@debian32b-anvuln:~/p13/prog_linux$ md5sum hello erase
da5c61e1edc0f18337e46418e48c1290  hello
da5c61e1edc0f18337e46418e48c1290  erase
```

2. Ejecuta los programas y compara sus salidas.

./hello
./erase

```
user@debian32b-anvuln:~/p13/prog_linux$ md5sum hello erase
da5c61e1edc0f18337e46418e48c1290  hello
da5c61e1edc0f18337e46418e48c1290  erase
```

3. Obtén el valor del hash sha256sum de los ejecutables "hello" y "erase".

sha256sum hello
sha256sum erase

```
user@debian32b-anvuln:~/p13/prog_linux$ ./erase
This program is evil!!!
Erasing hard drive...1Gb...2Gb... just kidding!
Nothing was erased.

(press enter to quit)
user@debian32b-anvuln:~/p13/prog_linux$ ./hello
Hello, world!

(press enter to quit)
user@debian32b-anvuln:~/p13/prog_linux$
```

github/truerandom

4. Compila la biblioteca "evilize" que se usará para crear la colisión MD5 de ejecutables, está basada en la implementación de Patrick Stach del algoritmo de Wang y Yu.

tar -xvf evilize-0.2.tar.gz
cd evilize-0.2
make

```
user@debian32b-anvuln:~/p13/evilize-0.2$ make
gcc -O3 -Wall -DSTDC_HEADERS -g -DMD5COLL_VERSION=\"0.1s\" -DVERSION=\"0.2\"   -c -o evilize.o evilize.c
gcc -O3 -Wall -DSTDC_HEADERS -g -DMD5COLL_VERSION=\"0.1s\" -DVERSION=\"0.2\"   -c -o md5.o md5.c
md5.c: In function ‘md5_finish_ctx’:
md5.c:125:3: warning: dereferencing type-punned pointer will break strict-aliasing rules [-Wstrict-aliasing]
   *(md5_uint32 *) &ctx->buffer[bytes + pad] = SWAP (ctx->total[0] << 3);
   ^
md5.c:126:3: warning: dereferencing type-punned pointer will break strict-aliasing rules [-Wstrict-aliasing]
   *(md5_uint32 *) &ctx->buffer[bytes + pad + 4] = SWAP ((ctx->total[1] << 3) |
   ^
gcc -O3 -Wall -DSTDC_HEADERS -g -DMD5COLL_VERSION=\"0.1s\" -DVERSION=\"0.2\"   -c -o md5coll_lib.o md5coll_lib.c
gcc   evilize.o md5.o md5coll_lib.o   -o evilize
gcc -O3 -Wall -DSTDC_HEADERS -g -DMD5COLL_VERSION=\"0.1s\" -DVERSION=\"0.2\"   -c -o md5coll.o md5coll.c
gcc   md5coll.o md5coll_lib.o   -o md5coll
gcc -O3 -Wall -DSTDC_HEADERS -g -DMD5COLL_VERSION=\"0.1s\" -DVERSION=\"0.2\"   -c -o goodevil.o goodevil.c
```

5. Crea un programa en C con dos funciones que impriman dos de tus frases favoritas, una por cada función. En vez de usar la función main() de forma tradicional, escribe dos funciones diferentes, una main_good() y otra main_evil(). Toma como ejemplo el programa hello-erase.c que está en la carpeta evilize-0.2/

```c
#include <stdio.h>
#include <unistd.h>

/* do something innocent */
int main_good(int ac, char *av[]) {
  char buf[10];
  fprintf(stdout, "No gods or kings, only man\n");
  fgets(buf, 10, stdin);
  return 0;
}

/* do something evil */
int main_evil(int ac, char *av[]) {
  char buf[10];
  fprintf(stdout, "Using no way as a way, having no limitation as limitation\n");
  fgets(buf, 10, stdin);
  return 0;
}
```

6. Compila tu programa y enlázalo con goodevil.o, por ejemplo si tu programa se llama hello-erase.c

gcc hello-erase.c goodevil.o -o hello-erase

```
user@debian32b-anvuln:~/p13/evilize-0.2$ gcc programa.c goodevil.o -o programa
```

7. Ejecuta el siguiente comando para crear un vector de inicialización (IV)

./evilize hello-erase -i

```
user@debian32b-anvuln:~/p13/evilize-0.2$ ./evilize programa -i
Initial vector: 0x9cc35269 0xb67d0e4a 0xcd17096a 0xdf9d4dab
```

github/truerandom

8. Crea una colisión ejecutando el siguiente comando, pero reemplaza el vector de inicialización del ejemplo con el resultado que te haya arrojado a tí del paso anterior.

```
user@debian32b-anvuln:~/p13/evilize-0.2$ ./md5coll 0x9cc35269 0xb67d0e4a 0xcd17096a 0xdf9d4dab > init.txt
Progress: 14.92.99.22 (done)
user@debian32b-anvuln:~/p13/evilize-0.2$ cat init.txt
Random seed: 1601140700
unsigned int m0[32] = {
0xebdfd945, 0x981c8ddc, 0x0f64516b, 0x0ddacc48,
0x613a130a, 0x69c1c173, 0xf22ca3db, 0xd2159bc1,
0x0533ecd1, 0x22334604, 0x85b4b2df, 0xe74a1b78,
0xdf9b6ed0, 0x9a2cd5df, 0xa84fd02a, 0x87c9de52,
0x01f18f86, 0x037fab1f, 0x250acc35, 0xb4c858ae,
0x1656d294, 0x1c7d7752, 0x06a8a942, 0xd04d46d5,
0x767a6f51, 0xab60957d, 0x778918be, 0x6fceb08c,
0x7dcd64c8, 0x87e80661, 0x3be26988, 0x8941819c,
};

unsigned int m1[32] = {
0xebdfd945, 0x981c8ddc, 0x0f64516b, 0x0ddacc48,
0xe13a130a, 0x69c1c173, 0xf22ca3db, 0xd2159bc1,
0x0533ecd1, 0x22334604, 0x85b4b2df, 0xe74a9b78,
0xdf9b6ed0, 0x9a2cd5df, 0x284fd02a, 0x87c9de52,
0x01f18f86, 0x037fab1f, 0x250acc35, 0xb4c858ae,
0x9656d294, 0x1c7d7752, 0x06a8a942, 0xd04d46d5,
0x767a6f51, 0xab60957d, 0x778918be, 0x6fce308c,
0x7dcd64c8, 0x87e80661, 0xbbe26988, 0x8941819c,
};
```

9. Crea un par de programas, uno bueno y otro malicioso, con el comando
./evilize hello-erase -c init.txt -g good -e evil

```
user@debian32b-anvuln:~/p13/evilize-0.2$ ./evilize programa -c init.txt -g good -e evil
Writing 'good' file good.
Writing 'evil' file evil.
```

10. Obtén los valores MD5 de los programas "good" y "evil", deben ser iguales.

```
user@debian32b-anvuln:~/p13/evilize-0.2$ md5sum good evil
9d773744061c55a58a1b54a4f5c9680e  good
9d773744061c55a58a1b54a4f5c9680e  evil
```

11. Ejecuta los programas "good" y "evil"

```
user@debian32b-anvuln:~/p13/evilize-0.2$ ./good
No gods or kings, only man

user@debian32b-anvuln:~/p13/evilize-0.2$ ./evil
Using no way as a way, having no limitation as limitation
```

12. Obtén los valores SHA256 de los programas "good" y "evil".

```
user@debian32b-anvuln:~/p13/evilize-0.2$ sha256sum good evil
56bc9df46ce1e883ac2d4cd5b82ba12248e6074a6bb5346197866df2e4bab6e0  good
a59447c156bbbab7ee62853c827a881a8fcda3c2014d3a36b031d5a0f06ade6c  evil
```