

INSTITUTO TECNOLÓGICO DE SAN JUAN DEL RÍO



Tópicos de ciberseguridad.

Práctica - Reconstrucción del código fuente.

P R E S E N T A:

CRISTIAN MARTINEZ MARTINEZ 21590392

JESÚS CRUZ CRUZ 21590382

SALVADOR RAMIREZ HERNANDEZ 20590284

INGENIERÍA EN SISTEMAS COMPUTACIONALES

PERIODO AGOSTO-DICIEMBRE 2025

Descripción: Usando una herramienta deben generar el código fuente de un ejecutable, luego debe volverse a compilar y se compara el resultado con el original.

Notas: El programa ejecutable debe ser sencillo, debe generar algo como una imagen bmp o un documento json, se sugiere el uso del compilador de TCC pues es sencillo de reconstruir con Ghidra.

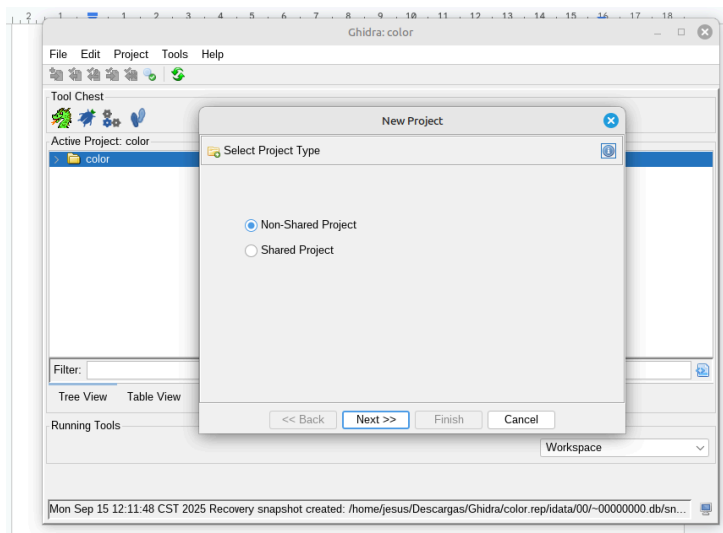
R003:

Descripción: Documentar el proceso de cómo obtuvieron el código fuente.

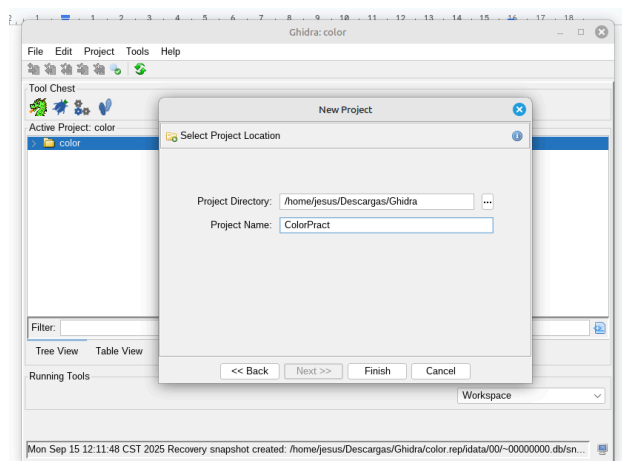
Obtención del código fuente:

Creación del proyecto: Una vez abierto Ghidra se crea un proyecto donde se analizará el código llamado “color.exe”.

New project → Non Shared Project → Next

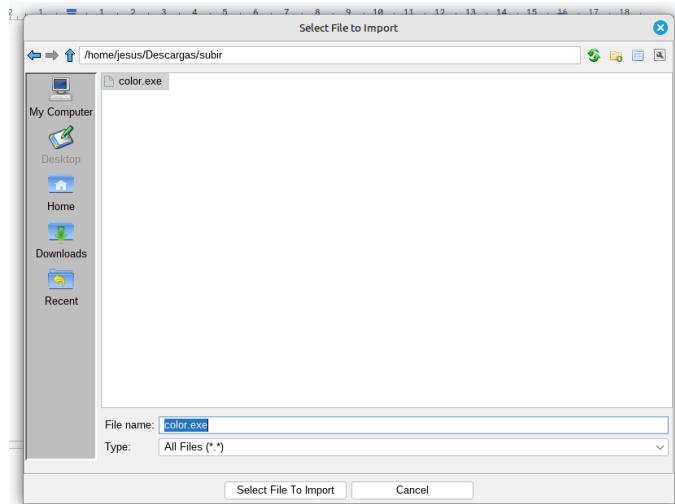


Definir la ruta y el nombre del proyecto:



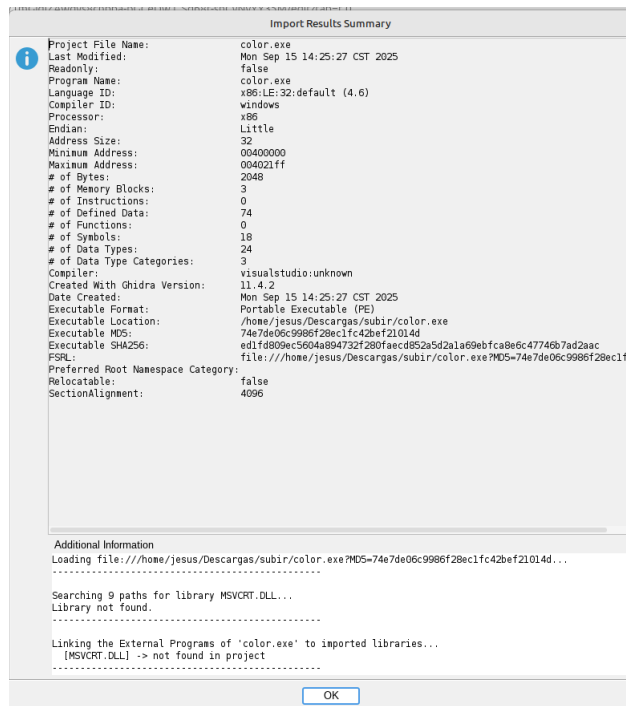
Importar el archivo “color.exe”:

file → import file → color.exe



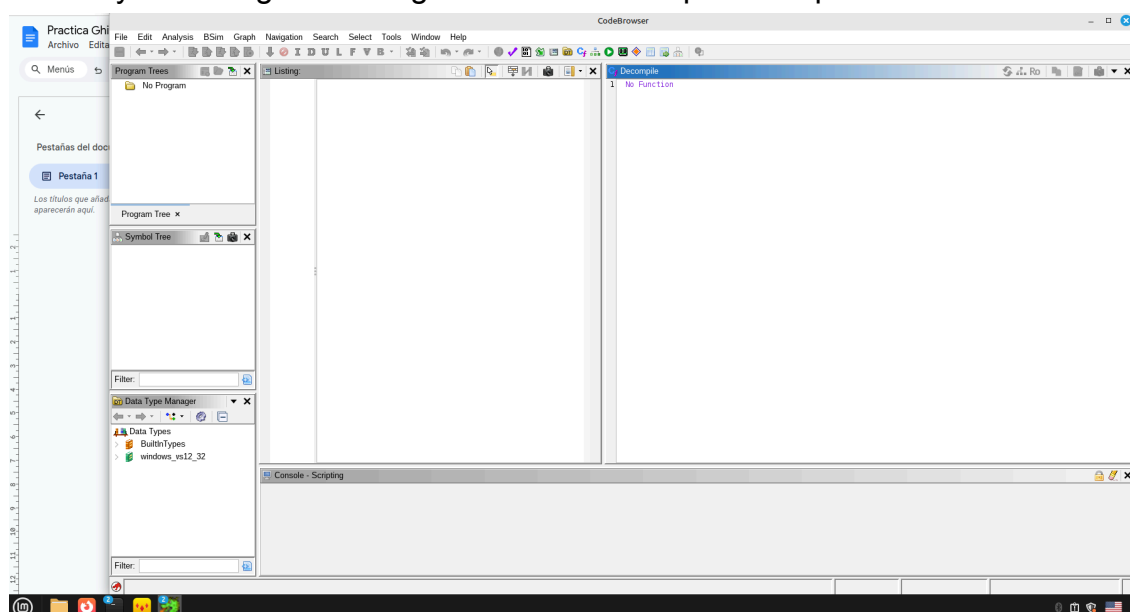
Propiedades del archivo:

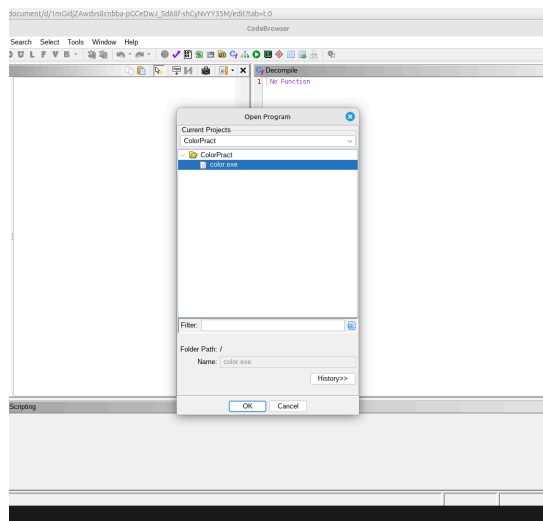
Una vez importado el ejecutable el mismo Ghidra muestra algunas propiedades del ejecutable:



Usar el CodeBrowser de Ghidra para analizar el archivo ejecutable:

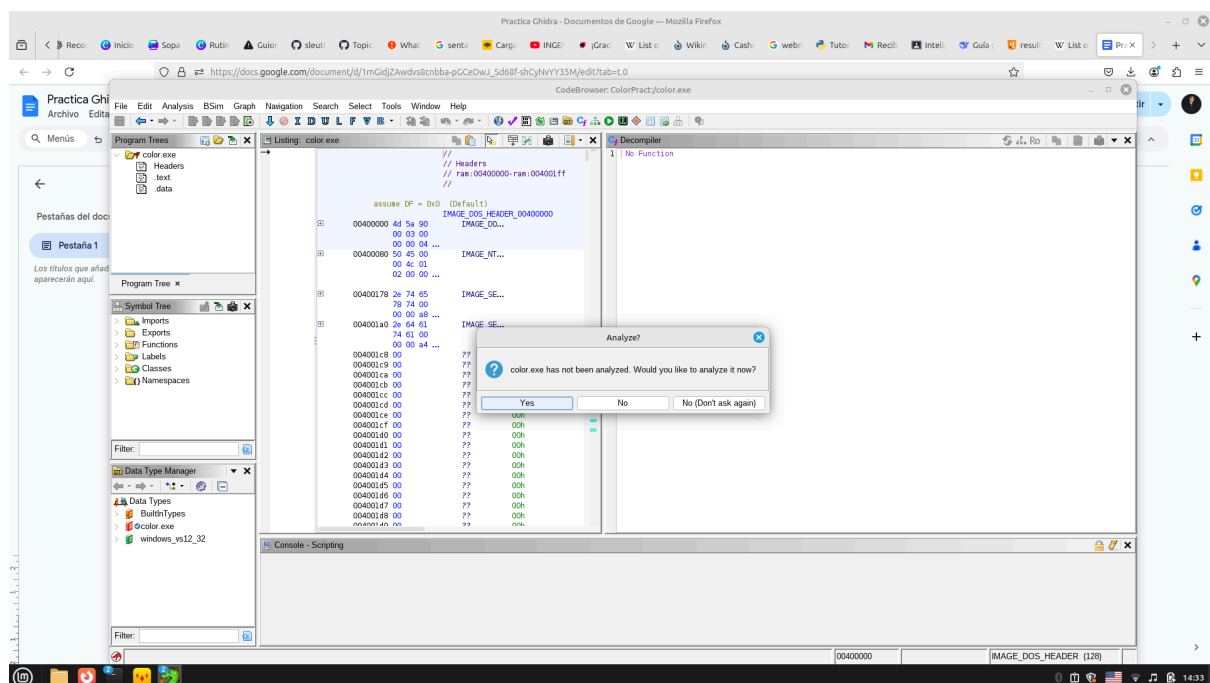
En esta parte se usa el CodeBrowser para elegir el archivo ejecutable para que lo analice y nos otorgue el código fuente o lo más parecido posible.

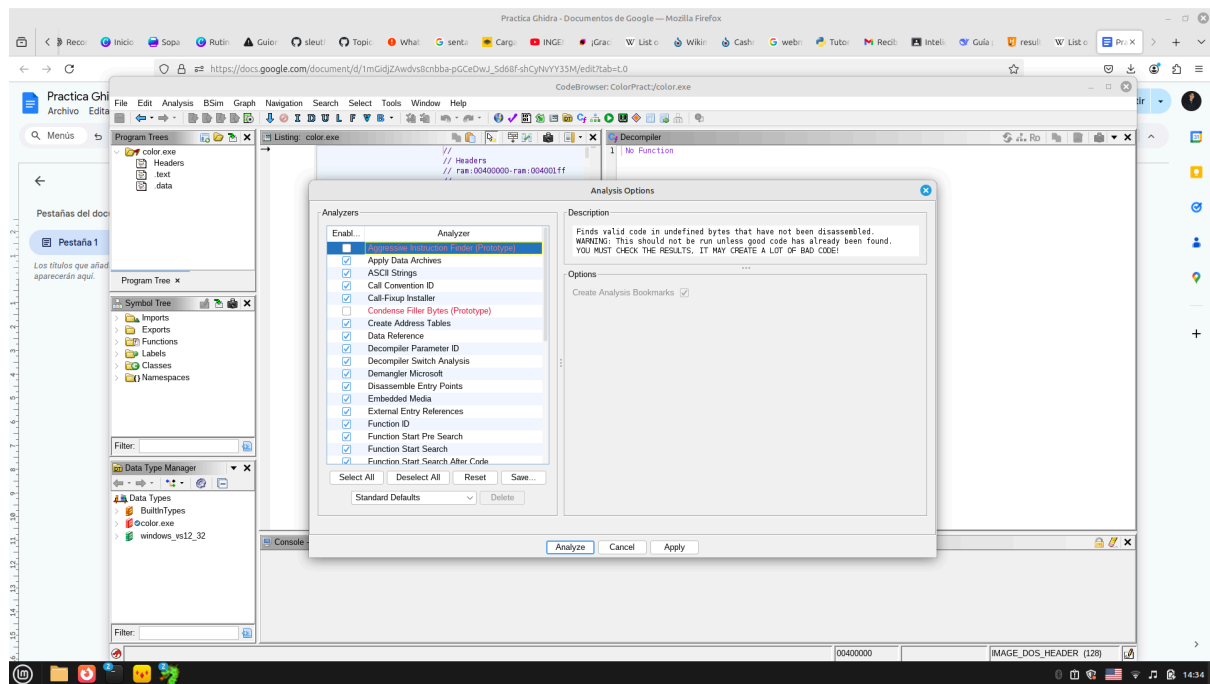




Analizar el ejecutable:

Al seleccionar el archivo a analizar nos pide confirmar de que ese archivo es el que se va a analizar y que todo es lo que se va a analizar, por lo que solo se confirma lo que se va a analizar.





Análisis y reconstrucción del código:

Identificación de cadenas de texto

- En el menú **Window** → **Defined Strings** se buscaron cadenas legibles.
- Se localizaron elementos clave:
 - "P6" (formato PPM binario).
 - "640 480" y "255", que corresponden a dimensiones y rango de color.
 - El nombre del archivo de salida (sample.ppm).
 - El modo de apertura (wb).

- Esto confirmó que el programa genera un archivo PPM.

Análisis de la función entry

- En **Symbol Tree** → **Functions** se abrió la función entry.
- El pseudocódigo mostró que solo se inicializa el entorno de C y finalmente se llama a la función FUN_00401000.
- Se concluyó que la lógica principal estaba en FUN_00401000.

Análisis de la función principal (FUN_00401000)

- El decompilador mostró:
 - Una llamada a time(NULL) y srand(), lo que indica inicialización de números aleatorios con la hora del sistema.
 - Una llamada a fopen("sample.ppm","wb") para crear el archivo de salida.
 - Una escritura inicial de 15 bytes con fwrite, que corresponde a la cabecera "P6\n640 480\n255\n".
 - Dos bucles anidados con límites 0x1e0 (480) y 0x280 (640), que representan las dimensiones de la imagen.
 - En cada iteración se generan 3 valores con rand()%256 y se escriben uno por uno con fwrite, representando los canales R, G y B de cada píxel.
 - Finalmente, fclose() cierra el archivo.
- De esta manera se dedujo la estructura del programa en C.

Traducción a código C estándar

- Se tomó el pseudocódigo de Ghidra y se reemplazaron nombres automáticos como DAT_00402000 o undefined1 por variables claras (header, unsigned char tmp).

Y se dio como resultado el siguiente código:

```

1 // color_recon.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <stdint.h>
5 #include <time.h>
6
7 /*
8  * Reconstrucción del comportamiento visto en el decompiler:
9  * - Cabecera P6 (15 bytes): "P6\n640 480\n255\n"
10  * - Seeding: srand(time(NULL))
11  * - Loop: height = 0x1e0 (480), width = 0x280 (640)
12  * - Por cada píxel escribe 3 bytes generados con rand()%0x100, usando fwrite(&byte,1,1,f)
13  */
14
15 int main(void) {
16     /* Constantes encontradas en el decompilado */
17     const int WIDTH = 0x280; // 640
18     const int HEIGHT = 0x1e0; // 480
19
20     /* Cabecera exacta (15 bytes) encontrada en DAT_00402000 */
21     const unsigned char header[] = "P6\n640 480\n255\n"; /* longitud 15 */
22
23     const char fname = "s_sample.ppm"; /* reemplaza por la cadena exacta si es otra */
24
25     FILE f = fopen(fname, "wb"); /* en decompiler aparece fopen(..., &DAT_00402023) */
26     if (!f) {
27         perror("fopen");
28         return 1;
29     }
30
31     if (fwrite(header, 1, sizeof(header)-1, f) != sizeof(header)-1) {
32         perror("fwrite header");
33         fclose(f);
34         return 1;
35     }
36
37     time_t t = time(NULL);
38     srand((unsigned int)t);

```



```


38
39     time_t t = time(NULL);
40     srand((unsigned int)t);
41
42     unsigned char tmp;
43     for (int y = 0; y < HEIGHT; ++y) {
44         for (int x = 0; x < WIDTH; ++x) {
45             /* R */
46             tmp = (unsigned char)(rand() % 0x100);
47             fwrite(&tmp, 1, 1, f);
48             /* G */
49             tmp = (unsigned char)(rand() % 0x100);
50             fwrite(&tmp, 1, 1, f);
51             /* B */
52             tmp = (unsigned char)(rand() % 0x100);
53             fwrite(&tmp, 1, 1, f);
54         }
55     }
56
57     fclose(f);
58     return 0;
59 }
60

```

Ejecución del código:

```
Windows PowerShell
PS C:\Users\jexxbl\Downloads\tcc-0.9.27-win32-bin\tcc> .\tcc reccolor.c -o colorrec.exe
PS C:\Users\jexxbl\Downloads\tcc-0.9.27-win32-bin\tcc> |
```

Creación del .exe:

 colorrec	15/09/2025 06:24 p. m.	Aplicación	2 KB
--	------------------------	------------	------

Resultados de comparación:

Tamaño de los archivos: 901 kb

Resolución: 640 x 480

Comparación visual:

Color.exe



colorrec.exe

