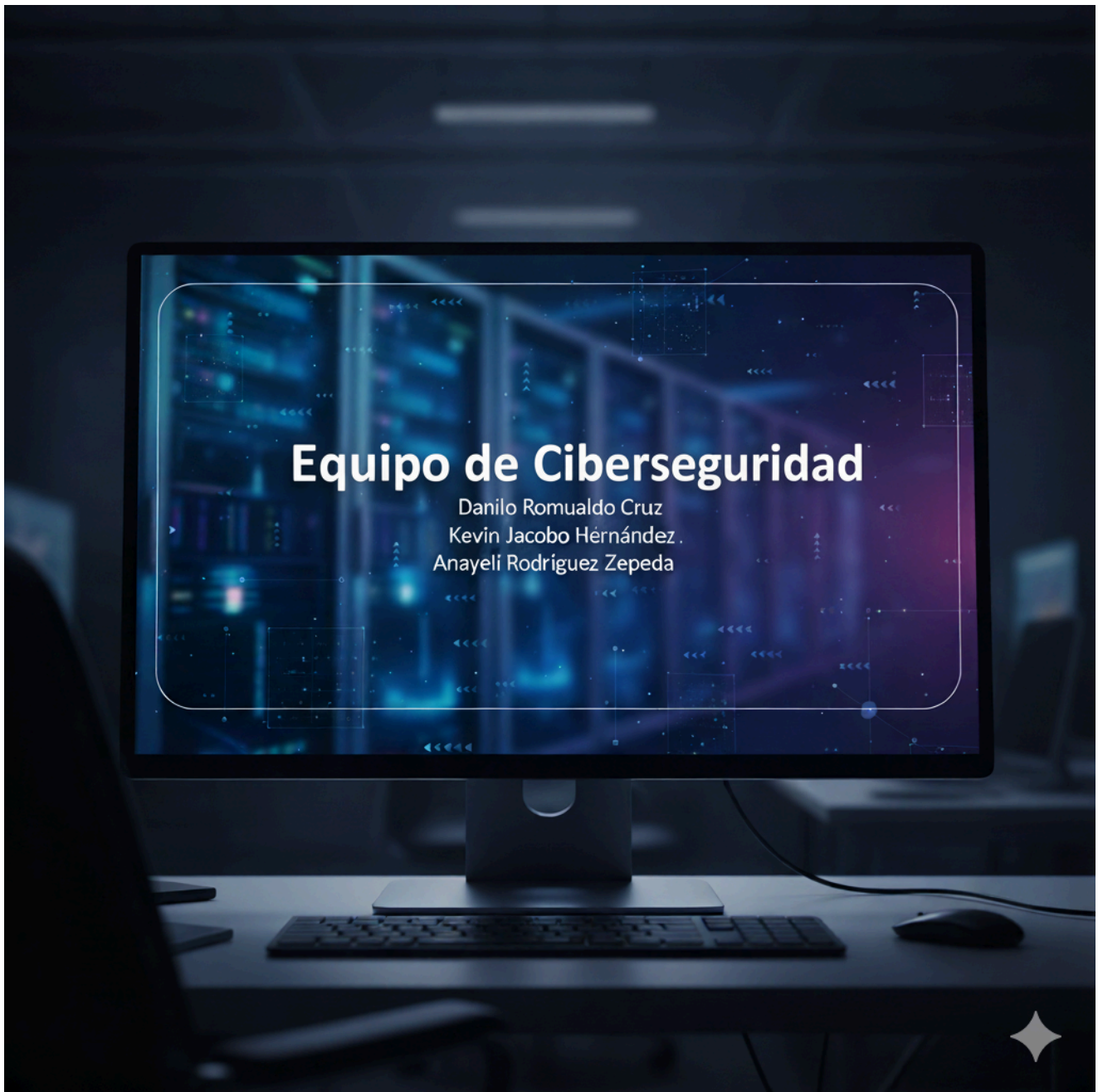
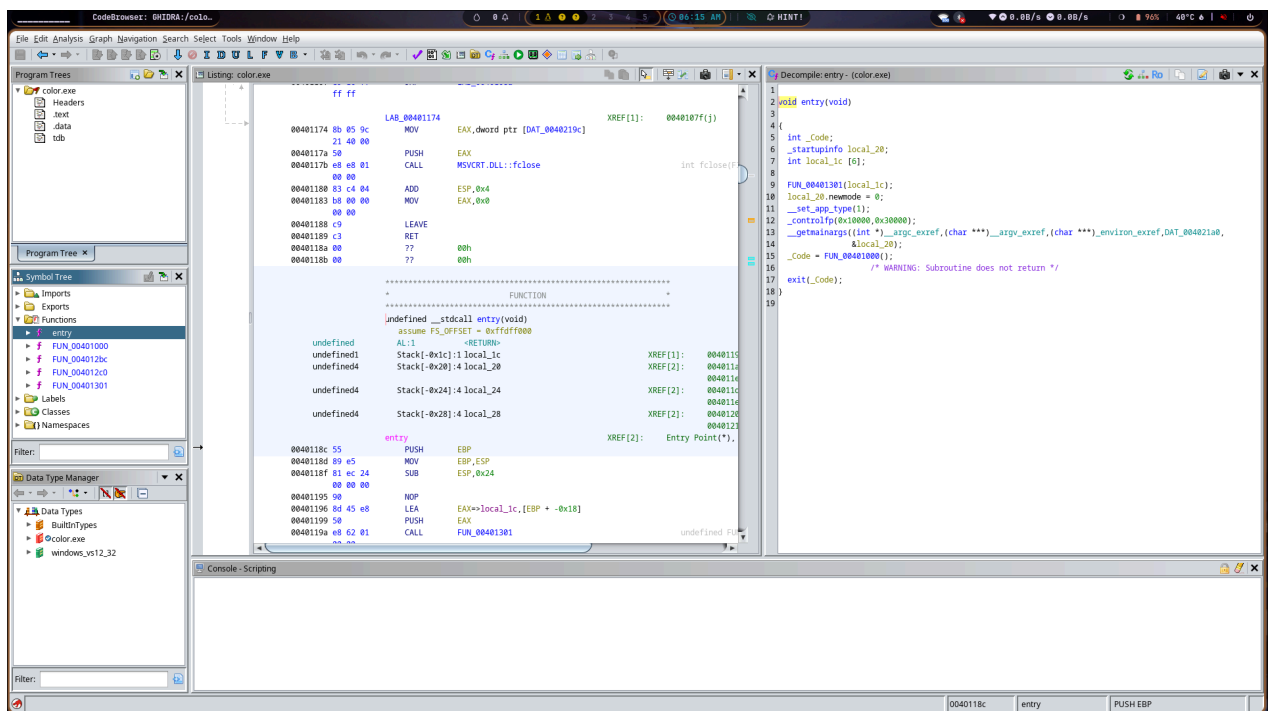


GHIDRA - PRACTICA 1



Apertura del GHIDRA

- En esta seccion primero ejecutamos nuestro GHIDRA con el exe proporcionario por el docente



2. - Buscar las funciones mas relevantes.


- Hay muchas funciones que realmente pueden servir o no ya que Ghidra solo nos da una interpretación del código en C, entonces después de haber visualizado las funciones que nos aparecen en su editor.
- Notamos que la función FUN_00401000 es la que tiene un contexto en código más explícito de lo que está haciendo el programa original.



Funciones Adicionales

- Esta seccion de la funcion no se ve tan relevante


Otras funciones adicionales

 Decompile: FUN_004012bc - (color.exe)

```

1
2 Undefined4 FUN_004012bc(void)
3
4 {
5     int unaff_EBP;
6
7     return *(undefined4 *) (unaff_EBP + -0x14);
8 }
9


```

 Decompile: FUN_004012c0 - (color.exe)

```

1
2 Undefined4 FUN_004012c0(void)
3
4 {
5     undefined4 *puVar1;
6
7     puVar1 = (undefined4 *) FUN_004012bc();
8     return *(undefined4 *) *puVar1;
9 }
10

```

 Decompile: FUN_00401301 - (color.exe)

```

1
2 void __cdecl FUN_00401301(int *param_1)
3
4 {
5     *param_1 = (int) &stack0x00000008;
6     param_1[1] = 0;
7     param_1[2] = (int) ExceptionList;
8     param_1[3] = (int) &LAB_004012fc;
9     param_1[4] = (int) &DAT_004012f0;
10    param_1[5] = 0;
11    ExceptionList = param_1 + 2;
12    return;
13 }
14

```

Conversion de Hexadecimales - Programa en C

Nota: Hacemos la transformacion de los valores hexadecimales que nos aparecen en el GHIDRA por valores decimales para nuestro codigo en C

Convertidor de hexadecimal a decimal

De A

Hexadecimal Decimal

Introduzca el número hexadecimal

0x100 16

= Convertir × Restablecer ↕ Intercambio

Número decimal (3 dígitos)

256 10

Decimal del complemento a 2 con signo

- Conversion de la siguiente seccion del modelo 0x1e0.

Nota: Estas convesiones las pase en una pagina web nomas para rectificar.

Hexadecimal to Decimal converter

From To

Hexadecimal Decimal

Enter hex number

0x1e0 16

= Convert × Reset ↕ Swap

Decimal number (3 digits)

480 10

Convertidor de hexadecimal a decimal

Desde Para

Hexadecimal ▼ Decimal ▼

Ingrese el número hexadecimal:

0x280 16

Convertir Reiniciar Intercambiar

Número decimal:

640 10

Decimal del complemento a 2 con signo:

OBSERVACIONES

Podemos notar que los valores Hexadecimales que nos esta arrojando parecidos a las dimensiones del archivo original asi que podemos deducir que esas son las que debemos de ocupar en nuestro archivo clonado

Investigacion de algunas estructuras de C

- En la FUN_00401000 se encuentra una seccion de codigo que decidimos investiga para entender que elementos recibiria del usuario o para que utilidad tiene en

especifico

Sintaxis

```
fwrite(const void * source, size_t size, size_t amount, FILE * fptr);
```

El `size_t` tipo de datos es un entero no negativo.

Valores de los parámetros

| Parameter | Description |
|---------------------|--|
| <code>source</code> | Required. A pointer to a block of memory where the data is copied from. |
| <code>size</code> | Required. The size of an element in the block of memory. |
| <code>amount</code> | Required. The number of elements to read from the block of memory and write into the file. |
| <code>fptr</code> | Required. A file pointer, usually created by the <code>fopen()</code> function. |

Estructura del Código en C

- Procedamos a la elaboración de la lógica del código de la creación de la imagen aleatoria.

```

27 // Doble bucle anidado para generar imagen 640x480
28 for (int y = 0; y < 480; y++) { // 0x1e0
29     for (int x = 0; x < 640; x++) { // 0x280
30         // Generar R
31         iVar1 = rand();
32         byte = (unsigned char)(iVar1 % 256);
33         fwrite(&byte, 1, 1, file);
34
35         // Generar G
36         iVar1 = rand();
37         byte = (unsigned char)(iVar1 % 256);
38         fwrite(&byte, 1, 1, file);
39
40         // Generar B
41         iVar1 = rand();
42         byte = (unsigned char)(iVar1 % 256);
43         fwrite(&byte, 1, 1, file);
44     }
45 }
46

```

Nota: En base a la transcripción del código que nos da ghidra y la conversión de los valores hexadecimales a decimal que decidimos transformarlos para una mejor legibilidad notamos.

- Los ciclos corresponden a la formación de una matriz por eso necesitamos los valores decimales para identificar las dimensiones que se toman para su creación.
 - se asignan valores aleatorios a las variables que corresponde a un valor sobre el módulo de 256 tal vez en esta parte se refiera a como los programas interpretan las imágenes con valores del 0 al 255
 - Estas características son las que le dan forma a la imagen aleatoria es la estructura principal del código
-

Código Final.

Nota: Esto ya es la recopilación de toda la transformación del código a C incluyendo dependencias así como la declaración de variables esto de manera general ya que lo

mas importante era la logica que hacia la imagen.

```

> .Vscode
> build
  a.out
  Com.c
  Final.c
  s_sample3.ppm
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 int FUN_00401000(void);
6
7 int main() {
8     return FUN_00401000();
9 }
10
11 int FUN_00401000(void) {
12     int iVar1;
13     time_t tVar2;
14     FILE *file;
15     unsigned char byte;
16
17     // Obtener tiempo actual y usarlo como semilla
18     tVar2 = time(NULL);
19     srand((unsigned int)tVar2);
20
21     // Abrir archivo en modo binario para escritura
22     file = fopen("s_sample4.ppm", "wb");
23
24     // Escribir cabecera PPM (formato P6)
25     fwrite("P6\n640 480\n255\n", 1, 15, file); // 0xF bytes
26
27     // Doble bucle anidado para generar imagen 640x480
28     for (int y = 0; y < 480; y++) { // 0x1e0
29         for (int x = 0; x < 640; x++) { // 0x280
30             // Generar R
31             iVar1 = rand();
32             byte = (unsigned char)(iVar1 % 256);
33             fwrite(&byte, 1, 1, file);
34
35             // Generar G
36             iVar1 = rand();
37             byte = (unsigned char)(iVar1 % 256);
38             fwrite(&byte, 1, 1, file);
39
40             // Generar B
41             iVar1 = rand();
42             byte = (unsigned char)(iVar1 % 256);
43             fwrite(&byte, 1, 1, file);
44         }
45     }
46
47     // Cerrar archivo
48     fclose(file);

```