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

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.

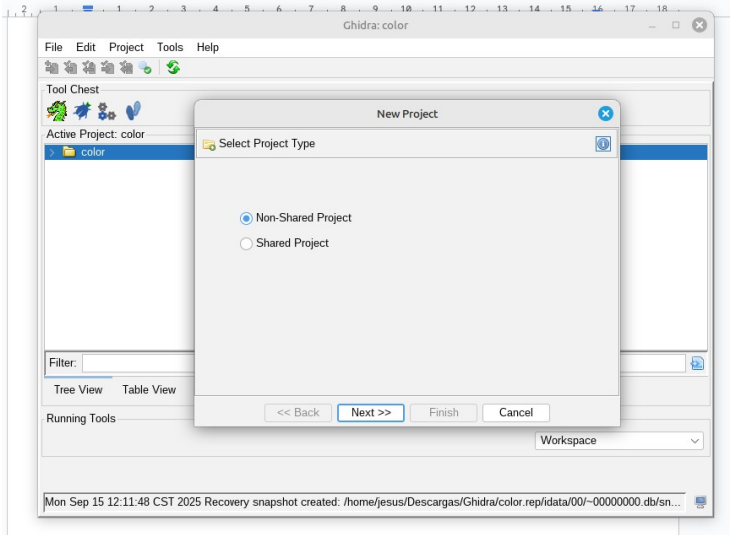
Notas: El programa ejecutable debe ser sencillo, debe generar algo como una imagen bmp o un documento json, se sugiere el uso del compilador de TCC pues es sencillo de reconstruir con Ghidra.

R003:
Descripción: Documentar el proceso de cómo obtuvieron el código fuente.

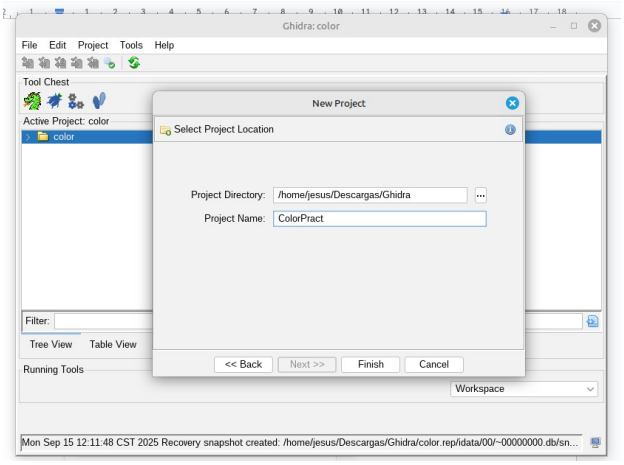
Obtención del código fuente:

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

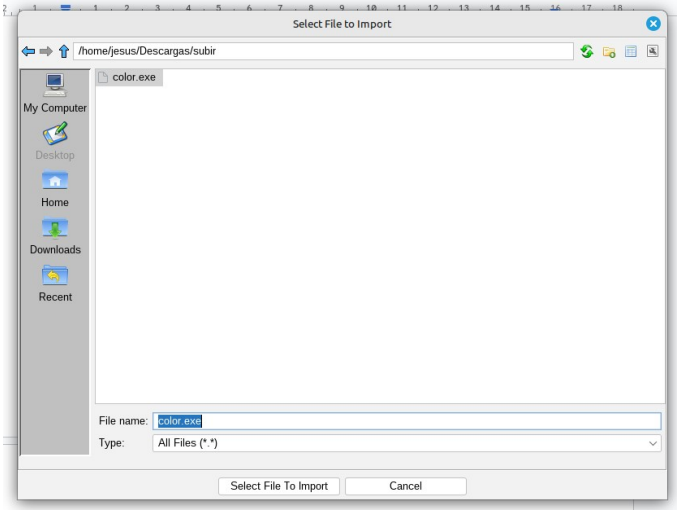


Definir la ruta y el nombre del proyecto:



Importar el archivo “color.exe”:

file → import file → color.exe

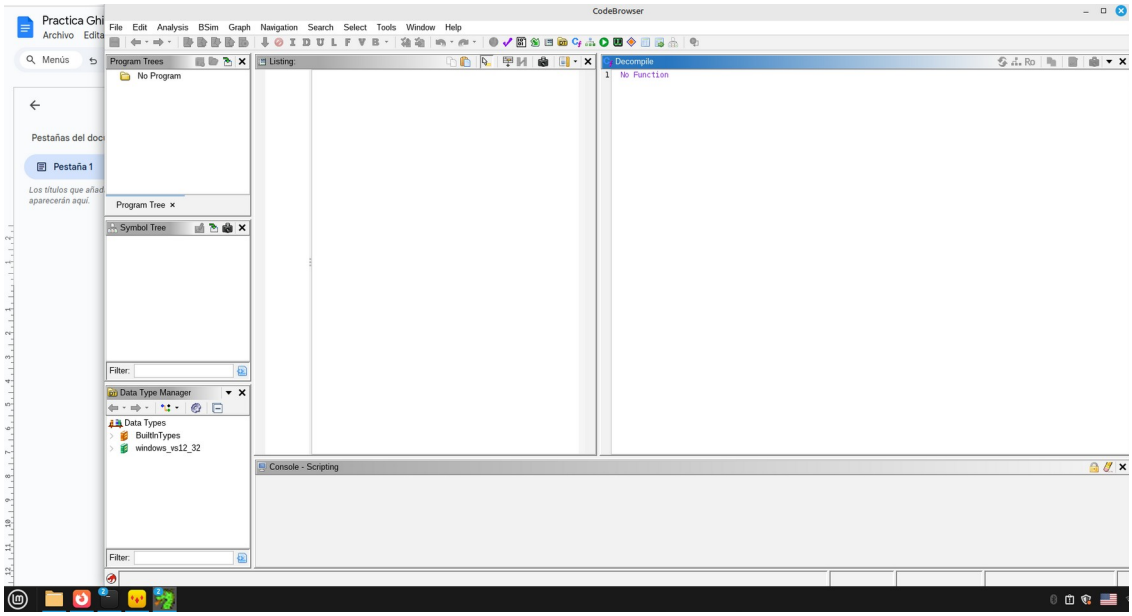


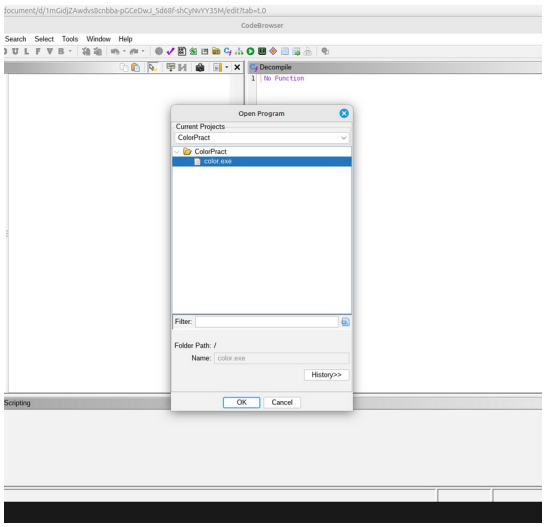
Propiedades del archivo:

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

Usar el CodeBrowser de Ghidra para analizar el archivo ejecutable:

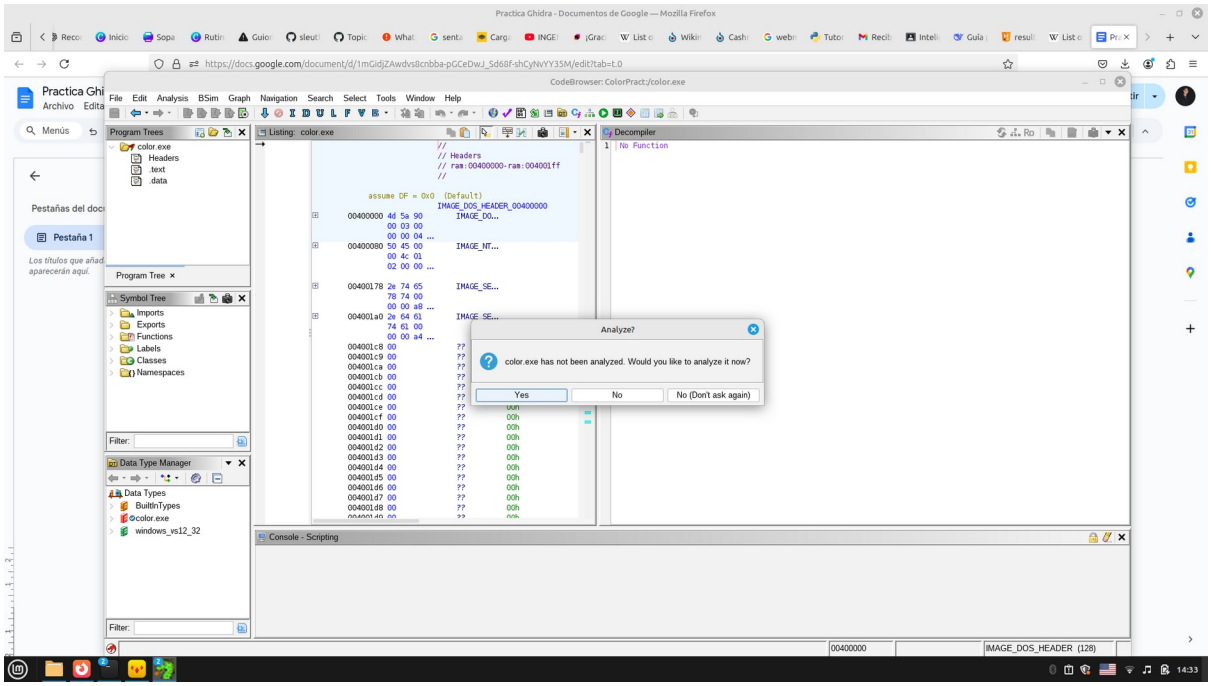
En esta parte se usa el CodeBrowser para elegir el archivo ejecutable para que lo analice y nos otorgue el código fuente o lo más parecido posible.

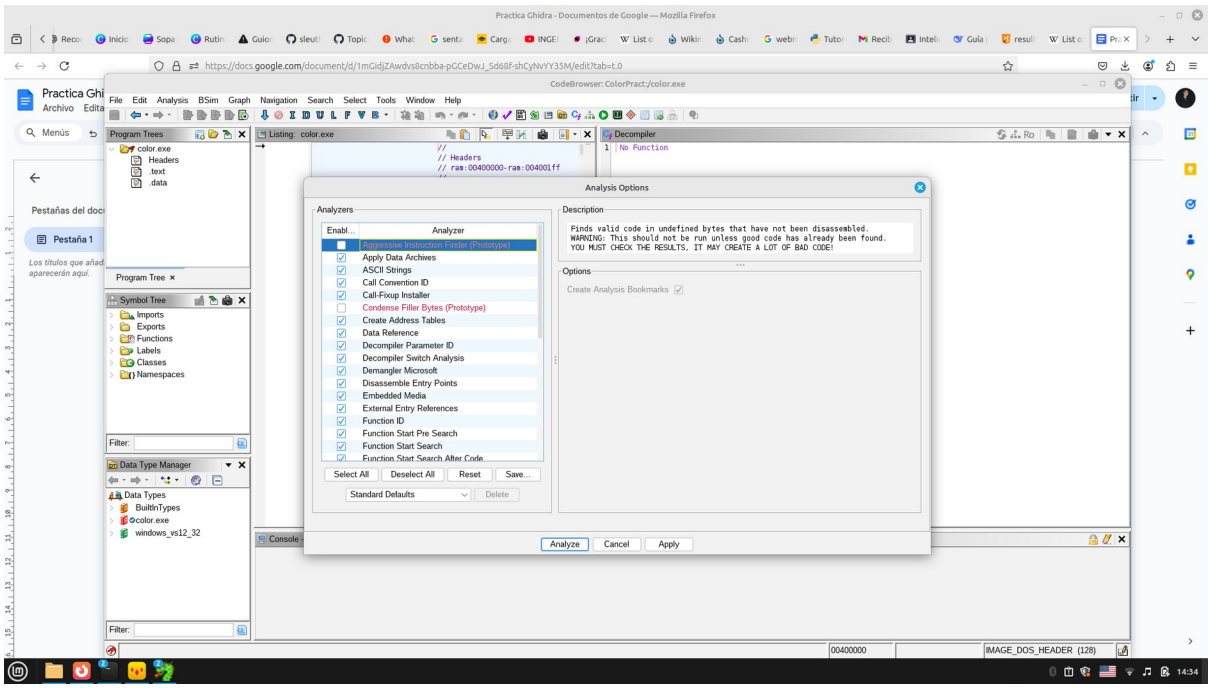




Analizar el ejecutable:

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.


Practica Ghidra - Documentos de Google — Mozilla Firefox
CodeBrowser: ColorPract:/color.exe
File Edit Analysis BSim Graph Navigation Search Select Tools Window Help
Program Trees
color.exe
Headers
.text
.data
Listing: color.exe
Decompiler
No Function
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Namespaces
Data Type Manager
Data Types
BuiltInTypes
color.exe
windows_vs12_32
Console - Scripting
Analyze?
color.exe has not been analyzed. Would you like to analyze it now?
Yes
No
No (Don't ask again)
00400000
IMAGE_DOS_HEADER (128)


Practica Ghidra - Documentos de Google — Mozilla Firefox
CodeBrowser: ColorPract:/color.exe
File Edit Analysis BSim Graph Navigation Search Select Tools Window Help
Program Trees
color.exe
Headers
.text
.data
Listing: color.exe
Decompiler
No Function
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Namespaces
Data Type Manager
Data Types
BuiltInTypes
color.exe
windows_vs12_32
Analysis Options
Analyzers
Enabl...
Analyzer
Aggressive Instruction Finder (Prototype)
Apply Data Archives
ASCII Strings
Call Convention ID
Call-Fixup Installer
Condense Filler Bytes (Prototype)
Create Address Tables
Data Reference
Decompiler Parameter ID
Decompiler Switch Analysis
Demangler Microsoft
Disassemble Entry Points
Embedded Media
External Entry References
Function ID
Function Start Pre Search
Function Start Search
Select All
Deselect All
Reset
Save...
Standard Defaults
Delete
Description
Finds valid code in undefined bytes that have not been disassembled.
WARNING: This should not be run unless good code has already been found.
YOU MUST CHECK THE RESULTS. IT MAY CREATE A LOT OF BAD CODE!
Options
Create Analysis Bookmarks
Analyze
Cancel
Apply
00400000
IMAGE_DOS_HEADER (128)

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:

```c
// color_recon.c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <time.h>

/*
 Reconstrucción del comportamiento visto en el decompiler:
 - Cabecera P6 (15 bytes): "P6\n640 480\n255\n"
 - Seeding: srand(time(NULL))
 - Loop: height = 0x1e0 (480), width = 0x280 (640)
 - Por cada píxel escribe 3 bytes generados con rand()%0x100, usando fwrite(&byte,1,1,f)
*/

int main(void) {
    /* Constantes encontradas en el decompilado */
    const int WIDTH  = 0x280; // 640
    const int HEIGHT = 0x1e0; // 480

    /* Cabecera exacta (15 bytes) encontrada en DAT_00402000 */
    const unsigned char header[] = "P6\n640 480\n255\n"; /* longitud 15 */


    const char fname = "s_sample.ppm"; / reemplaza por la cadena exacta si es otra */

    FILE f = fopen(fname, "wb"); / en decompiler aparece fopen(..., &DAT_00402023) */
    if (!f) {
        perror("fopen");
        return 1;
    }


    if (fwrite(header, 1, sizeof(header)-1, f) != sizeof(header)-1) {
        perror("fwrite header");
        fclose(f);
        return 1;
    }

    time_t t = time(NULL);
    srand((unsigned int)t);
```

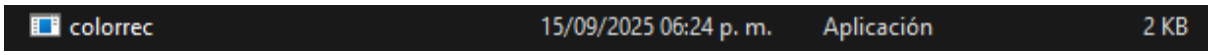
```c
    time_t t = time(NULL);
    srand((unsigned int)t);

    unsigned char tmp;
    for (int y = 0; y < HEIGHT; ++y) {
        for (int x = 0; x < WIDTH; ++x) {
            /* R */
            tmp = (unsigned char)(rand() % 0x100);
            fwrite(&tmp, 1, 1, f);
            /* G */
            tmp = (unsigned char)(rand() % 0x100);
            fwrite(&tmp, 1, 1, f);
            /* B */
            tmp = (unsigned char)(rand() % 0x100);
            fwrite(&tmp, 1, 1, f);
        }
    }

    fclose(f);
    return 0;
}
```

Ejecución del código:

Creación del .exe:



Resultados de comparación:

Tamaño de los archivos: 901 kb
Resolución: 640 x 480

Comparación visual:

Color.exe colorrec.exe