Instituto Tecnológico de San Juan del Río

Tópicos de Ciberseguridad

Actividad 3: Reconstrucción del código fuente (Ghidra)

P R E S E N T A:

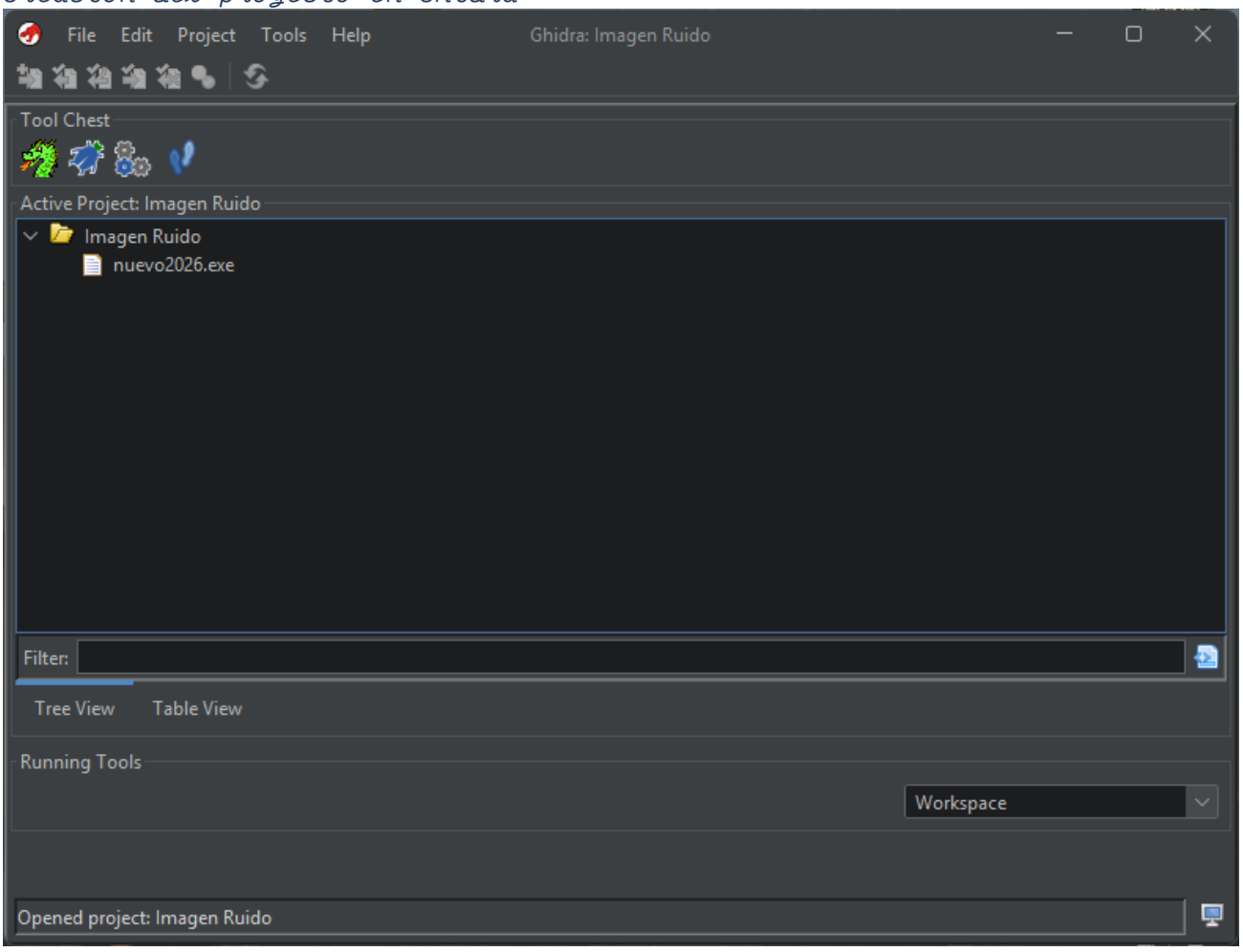
Ingeniería Sistemas Computacionales
Hernández Lucio Isaac
21590386

Martínez Yáñez Alexis Jonathan
21590291

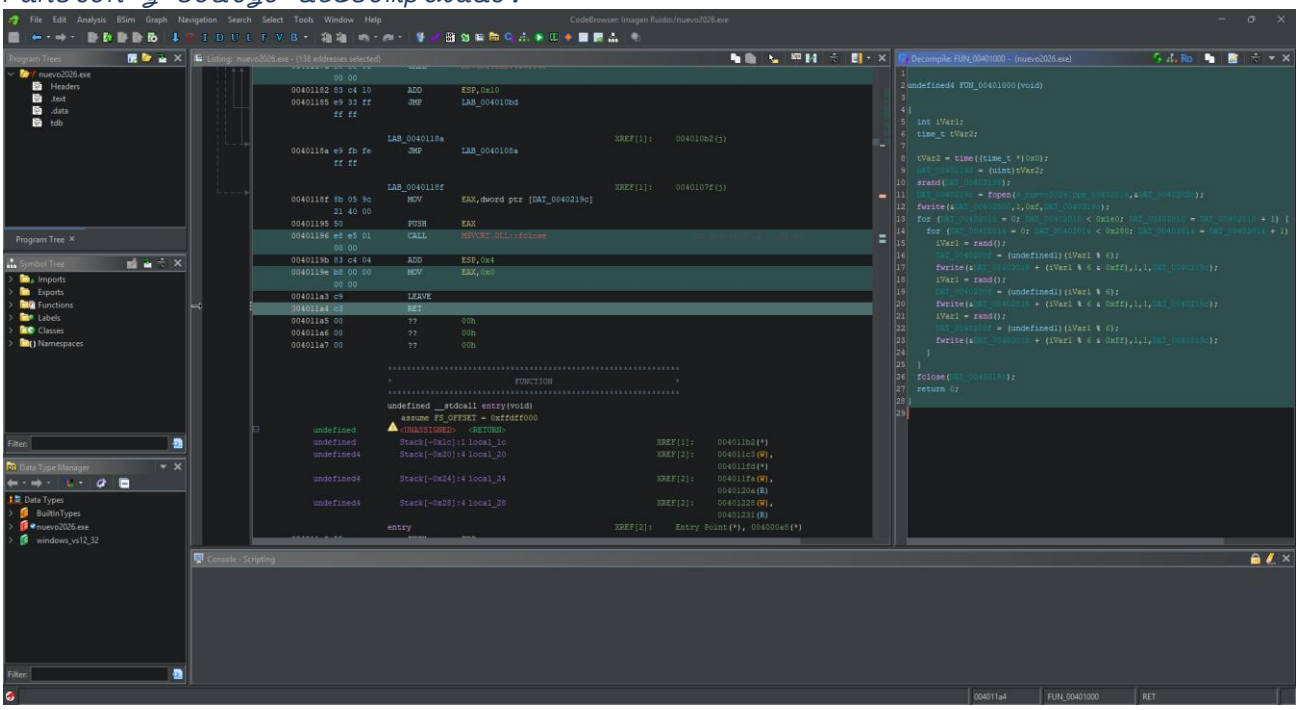
Uribe Callejas José Uriel
21021001

ENE-JUL 2026

Creación del proyecto en Ghidra



Función y código descompilado.



Una vez que el ejecutable esté cargado se muestra la interfaz, la pantalla principal es una ventana con el código desensamblado mientras que la pantalla de la izquierda muestra una de las funciones descompilado que en este caso es la función principal.

Análisis del código descompilado (Función: FUN_00401000).

Esta función genera un archivo PPM (Portable Pixmap) llamado "nuevo2026.ppm" con datos aleatorios.

Datos del archivo generado:

- Ancho: 0x280 = 640 píxeles
- Alto: 0x1E0 = 480 píxeles
- Formato: PPM binario P6

Reinterpretación de código.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
    FILE *archivo;
    time_t semilla;
    int i, j;
    int color;
    unsigned char paleta[6] = {0, 50, 100, 150, 200, 250}; // Colores de la paleta

    // Inicializar generador de números aleatorios
    semilla = time(NULL);
    srand((unsigned int)semilla);

    // Abrir archivo PPM
    archivo = fopen("nuevo2026.ppm", "+wb");
    if (archivo == NULL)
        return 1;

    // Escribir encabezado PPM (P6 640 480 255)
    fprintf(archivo, "P6\n640 480\n255\n");

    // Generar imagen de 480 filas x 640 columnas
    for (i = 0; i < 480; i++)
    { // 0x1e0 = 480
        for (j = 0; j < 640; j++)
        { // 0x280 = 640
            // Escribir 3 componentes RGB aleatorios
            color = rand() % 6;
            fwrite(&paleta[color], 1, 1, archivo); // R

            color = rand() % 6;
            fwrite(&paleta[color], 1, 1, archivo); // G

            color = rand() % 6;
            fwrite(&paleta[color], 1, 1, archivo); // B
        }
    }

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

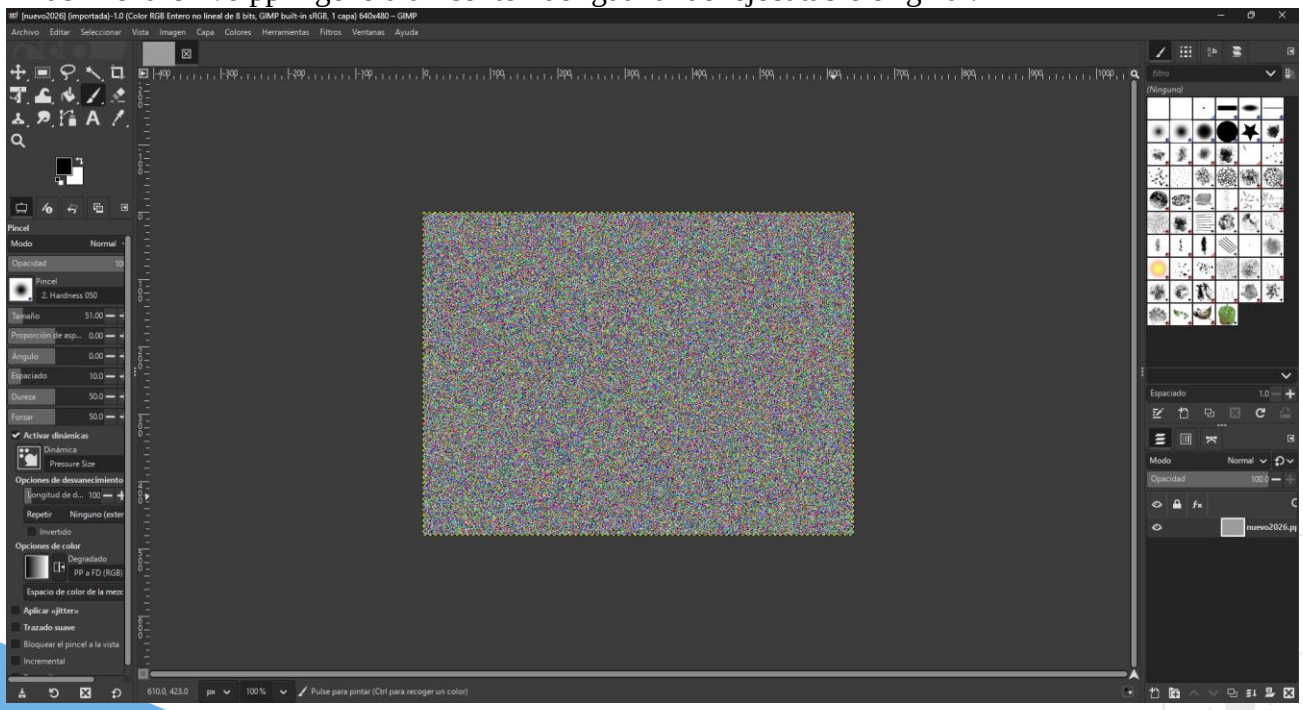
Compilación

```
PROBLEMAS  SALIDA  TERMINAL  DEVDB  GITLENS  SPELL CHECKER  PUERTOS  CONSOLA DE DEPURACIÓN
PS C:\Users\gameg\OneDrive\Documents\Escuela\Topicos de Ciberseguridad> cd "Actividades/Tema 1/Actividad 3"
PS C:\Users\gameg\OneDrive\Documents\Escuela\Topicos de Ciberseguridad\Actividades\Tema 1\Actividad 3> tcc CodigoNuevo2026.c
PS C:\Users\gameg\OneDrive\Documents\Escuela\Topicos de Ciberseguridad\Actividades\Tema 1\Actividad 3>
```

Una vez compilado el código, genera este ejecutable y al ejecutarlo crea el archivo ppm:

```
Length Name
------ ----
       Imagen Ruido.rep
       src
  1122 CodigoNuevo2026.c
  2560 CodigoNuevo2026.exe
1592826 Documentación Actividad 3.docx
     0 Imagen Ruido.gpr
921615 nuevo2026.ppm
   603 nuevo2026.py
```

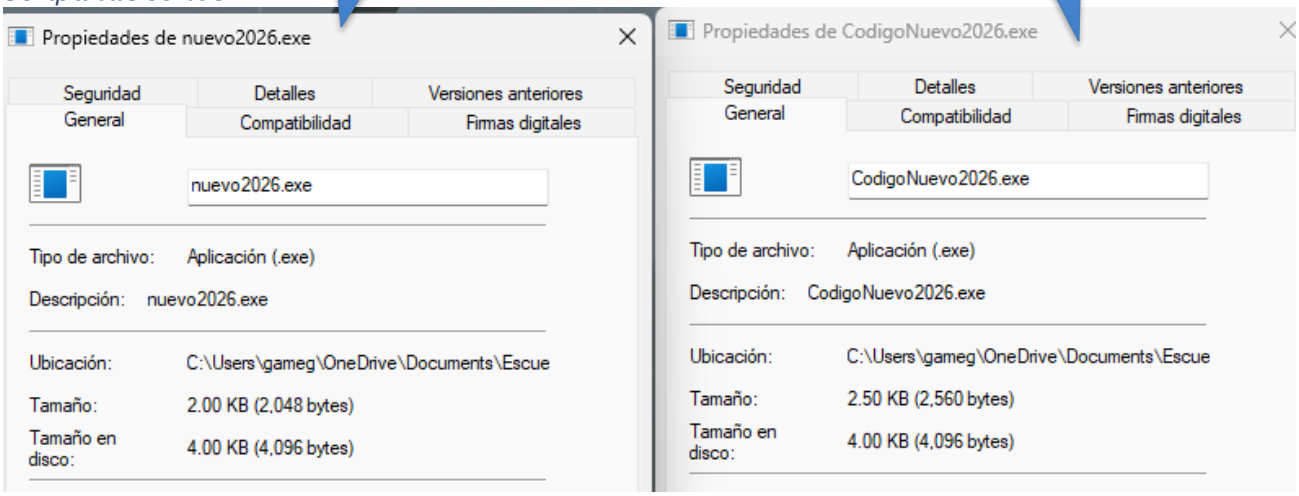
Al abrir el archivo ppm genera un contenido igual al del ejecutable original:



Ejecutable Original

Ejecutable Reconstruido

Comparaciones



Como se puede apreciar los ejecutables tienes un tamaño similar, aunque tengan esa ligera diferencia, el código nuevo hace exactamente el mismo proceso.

Conclusiones

La reconstrucción del código fuente a partir del ejecutable mediante el uso de Ghidra, permitió recuperar la funcionalidad del programa original. El código obtenido fue capaz de generar archivos PPM idénticos, lo que quiere decir que la lógica se mantuvo.

Pero este proceso de ingeniería inversa tiene algunas limitaciones. Durante la compilación se pierde información relevante como los nombres originales de variables y detalles del código. Aunque la funcionalidad sea la misma, los ejecutables recompilados no coinciden byte a byte con el binario original.

A pesar de esto, la ingeniería inversa es una herramienta muy importante, como la auditoría de seguridad, el análisis de malware y la recuperación o mantenimiento de software legado.