



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO®



Instituto Tecnológico de San Juan del Río



Tópicos de Ciberseguridad

P R E S E N T A:

Maqueda Durazno Alexis

Mendoza Garcia Daniela

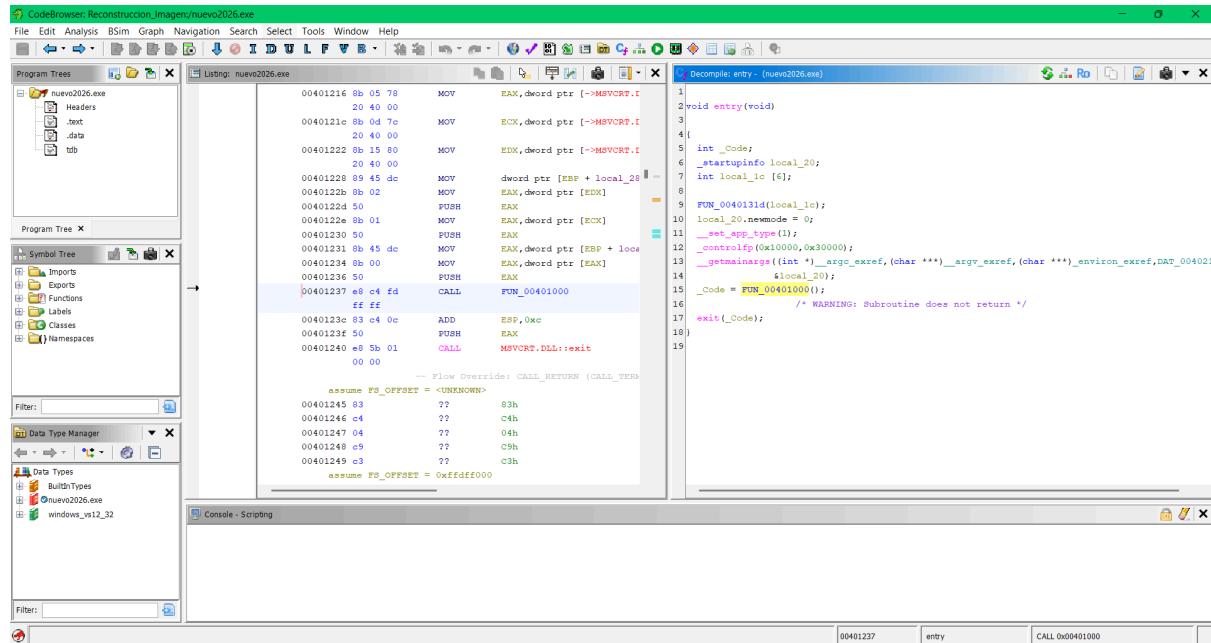
Rengel Olvera Paloma

Ing. Sistemas Computacionales

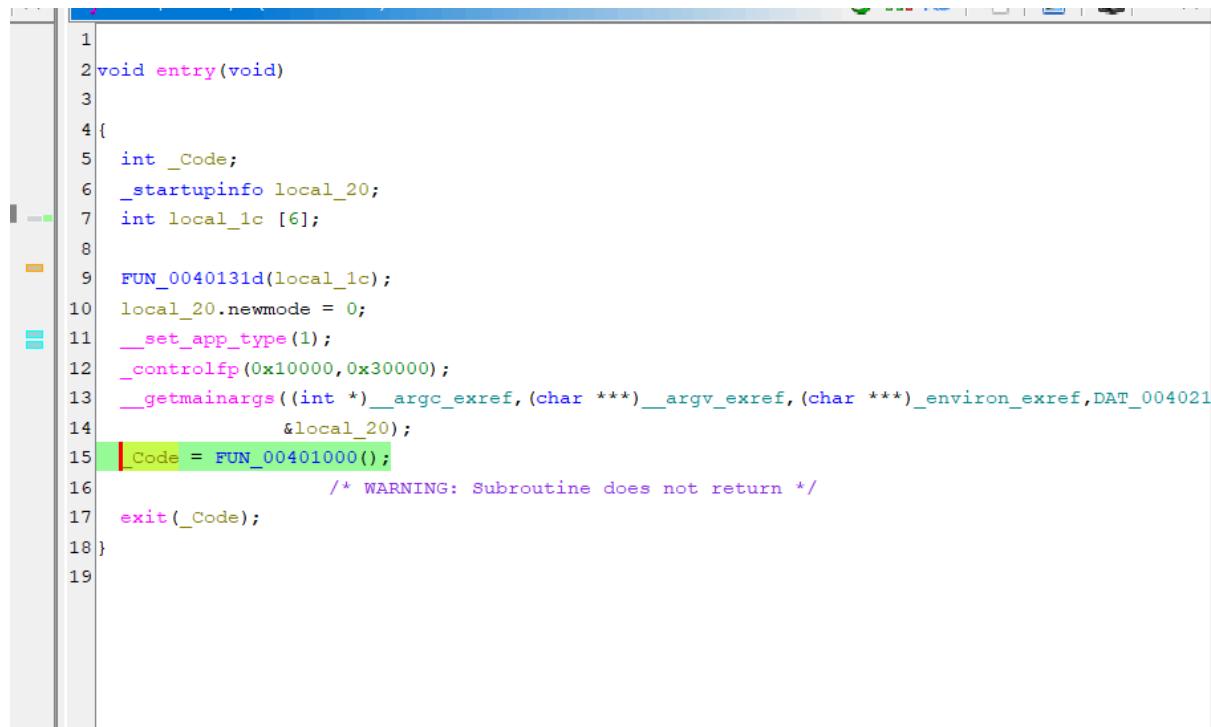




1- Primero creamos un nuevo proyecto en Ghidra, cargamos el ejecutable en el proyecto y para ver el código fuente



Ubicamos la parte del código que nos interesa



precionamos doble click y nos lleva a otra ventana



Decompile: FUN_00401000 - (nuevo2026.exe)

```
2 undefined4 FUN_00401000(void)
3
4 {
5     int iVar1;
6     time_t tVar2;
7
8     tVar2 = time((time_t *)0x0);
9     DAT_00402198 = (uint)tVar2;
10    srand(DAT_00402198);
11    DAT_0040219c = fopen(s_nuevo2026.ppm_0040201e,&DAT_0040202c);
12    fwrite(&DAT_00402000,1,0xf,DAT_0040219c);
13    for (DAT_00402010 = 0; DAT_00402010 < 0x1e0; DAT_00402010 = DAT_00402010 + 1) {
14        for (DAT_00402014 = 0; DAT_00402014 < 0x280; DAT_00402014 = DAT_00402014 + 1) {
15            iVar1 = rand();
16            DAT_0040200f = (undefined1)(iVar1 % 6);
17            fwrite(&DAT_00402018 + (iVar1 % 6 & 0xff),1,1,DAT_0040219c);
18            iVar1 = rand();
19            DAT_0040200f = (undefined1)(iVar1 % 6);
20            fwrite(&DAT_00402018 + (iVar1 % 6 & 0xff),1,1,DAT_0040219c);
21            iVar1 = rand();
22            DAT_0040200f = (undefined1)(iVar1 % 6);
23            fwrite(&DAT_00402018 + (iVar1 % 6 & 0xff),1,1,DAT_0040219c);
24        }
25    }
26    fclose(DAT_0040219c);
27    return 0;
28 }
```

esta parte es la que nos interesa
la analizamos:

1. Inicialización y Semilla (Generación Aleatoria)

El programa comienza configurando un generador de números aleatorios basado en el tiempo actual del sistema:

- **time(0)**: Obtiene la hora actual.
- **srand()**: Utiliza esa hora como "semilla". Esto asegura que cada vez que ejecutes el programa, la imagen generada sea distinta.

2. Creación del Archivo

- **fopen**: Abre un archivo llamado **nuevo2026.ppm** para escritura.
- **fwrite (Línea 12)**: Escribe el encabezado del archivo. En el formato PPM, esto define el tipo de imagen, el ancho y el alto.

3. El Bucle de Generación de Píxeles



El código utiliza dos bucles **for** anidados para llenar la imagen, lo cual es típico para manejar coordenadas (filas y columnas):

- **Dimensiones:**
 1. El bucle externo corre hasta **0x1e0** (480 en decimal).
 2. El bucle interno corre hasta **0x280** (640 en decimal).
 3. **Resultado:** Se está creando una imagen de **640 x 480 píxeles**.
- **Generación de Color:** Dentro del bucle interno, el programa genera tres valores aleatorios por cada píxel (uno para **Rojo**, uno para **Verde** y uno para **Azul**):
 1. Llama a **rand()**.
 2. Calcula **iVar1 % 6** (el resto de dividir por 6), lo que limita el valor a un rango muy pequeño (0 a 5).
 3. Usa ese resultado como un índice para buscar un color predefinido en una dirección de memoria (**DAT_00402018**).
 4. Escribe ese byte en el archivo mediante **fwrite**.

4. Finalización

- **fclose:** Cierra el archivo para asegurar que todos los datos se guarden correctamente en el disco.
- **return 0:** Finaliza la ejecución de la función.

Una vez analizado el código, escribimos el mismo código (la misma lógica pero ahora en python) quedando de esta manera

```
 1 > import ...
 3 def generar_imagen(): 1 usage
 4
 5     ancho = 640
 6     alto = 480
 7
 8     random.seed(int(time.time()))
 9
10     paleta = [0, 50, 100, 150, 200, 255]
11
12     try:
13         with open("nuevo2026.ppm", "wb") as f:
14             f.write(f"P6\n{ancho} {alto}\n255\n".encode())
15             for y in range(alto):
16                 for x in range(ancho): # DAT_00402014 < 0x280
17                     r = paleta[random.randint( a: 0, b: 5)]
18                     g = paleta[random.randint( a: 0, b: 5)]
19                     b = paleta[random.randint( a: 0, b: 5)]
20
21                     f.write(bytes([r, g, b]))
22
23             print("Imagen 'nuevo2026.ppm' generada con éxito.")
24
25     except Exception as e:
26         print(f"Error al crear la imagen: {e}")
27
28
29 D> if __name__ == "__main__":
30     generar_imagen()
```

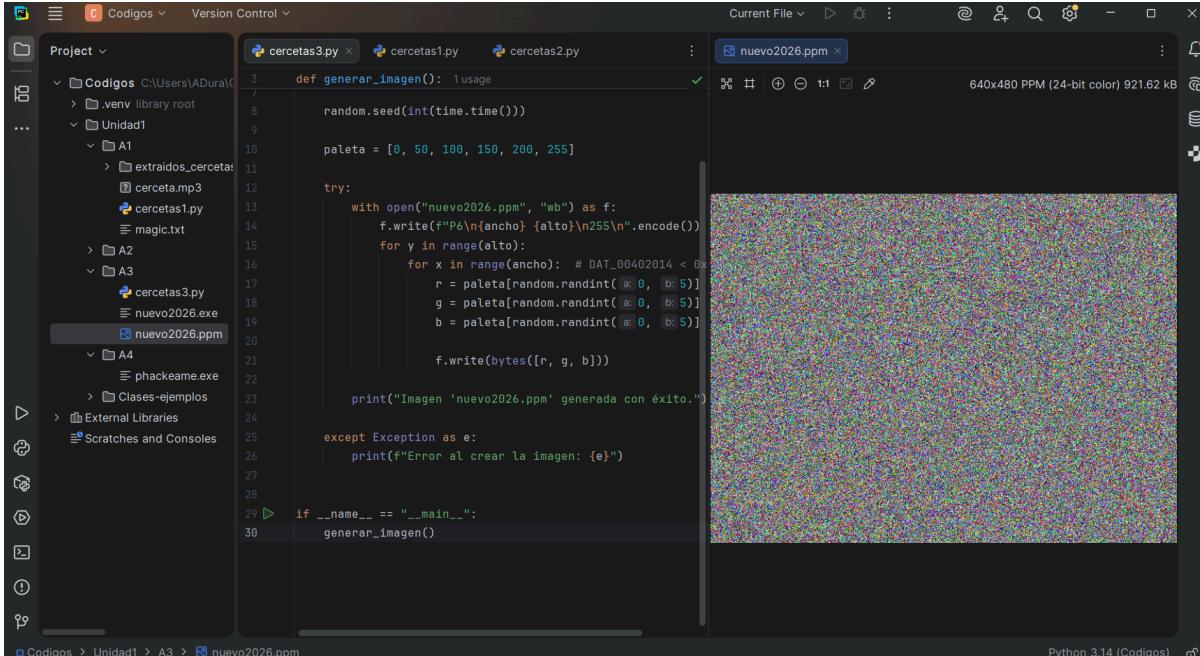


Primero, el script define las dimensiones de la imagen como **640x480 píxeles** y utiliza el reloj del sistema (`time.time()`) para que los números aleatorios cambien en cada ejecución. También establece una **paleta** con seis tonos específicos de intensidad (de 0 a 255), que son los únicos colores que el programa puede usar.

Al ejecutar la función, se crea un archivo binario llamado `nuevo2026.ppm`. Lo primero que se escribe es el encabezado **P6**, que le dice a cualquier visor de imágenes que el archivo es un **Portable Pixmap** en formato binario, seguido del ancho, el alto y el valor máximo de color.

Finalmente, el programa entra en un ciclo doble (uno para las filas y otro para las columnas) que recorre cada punto de la imagen. Por cada píxel, elige al azar un valor de la paleta para el canal rojo, otro para el verde y otro para el azul, escribiendo esos tres bytes directamente en el archivo. El resultado visual es una imagen de "ruido" o estática de colores, similar a la interferencia de una televisión antigua pero con puntos de colores muy marcados.

Dando el mismo resultado en la imagen generada



```
def generar_imagen():
    usage
    random.seed(int(time.time()))

    paleta = [0, 50, 100, 150, 200, 255]

    try:
        with open("nuevo2026.ppm", "wb") as f:
            f.write(f"P6\n{ancho} {alto}\n255\n".encode())
            for y in range(alto):
                for x in range(ancho): # DAT_00402014 < 0x
                    r = paleta[random.randint(0, 5)]
                    g = paleta[random.randint(0, 5)]
                    b = paleta[random.randint(0, 5)]

                    f.write(bytes([r, g, b]))

            print("Imagen 'nuevo2026.ppm' generada con éxito.")

    except Exception as e:
        print(f"Error al crear la imagen: {e}")

if __name__ == "__main__":
    generar_imagen()
```