



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de San Juan del Río



**[Tema 1
##R003## Gihra]**

[Tópicos de Ciberseguridad]

P R E S E N T A:

EQUIPO NARANJA
[Emma Belén Márquez García
María del Carmen Pérez Cruz
Karla Daniela Pérez González

[Ingería en Sistemas Computacionales]

PERIODO [Enero-Junio]

1. ANÁLISIS DEL EJECUTABLE EN GHIDRA

- Se cargó el ejecutable en Ghidra.
- Se identificó la función entry, la cual solo inicializa el runtime.
- Se localizó la función principal real: FUN_00401000.

The screenshot shows the Ghidra interface with the following details:

- Program Tree:** Shows the file structure of "nuevo2026.exe" with sections like Headers, .text, .data, and .rsrc.
- Symbol Tree:** Lists imports, exports, functions, labels, classes, and namespaces.
- Listing View:** Displays assembly code for the entry point. It includes instructions like `push ebp`, `mov esp,ebp`, and `call local_1c`.
- Decompiler View:** Shows the C-like pseudocode for the entry function, which initializes a `startuinfo` structure and calls `FUN_0040131d`.
- Console:** A scripting console at the bottom.

2. IDENTIFICACIÓN DE LA LÓGICA PRINCIPAL

- Dentro de FUN_00401000 se observó:
 - Llamada a: `time(NULL)` `srand(seed)`
 - Se identificaron los límites de los ciclos: `0x280 → 640` (ancho) `0x1e0 → 480` (alto)

The screenshot shows the Ghidra interface with the following details:

- Program Tree:** Shows the file structure of "nuevo2026.exe".
- Symbol Tree:** Lists imports, exports, functions, labels, classes, and namespaces.
- Listing View:** Displays assembly code for the main loop function, labeled FUN_00401000. It includes a loop from `0x00402010` to `0x00402014` with a condition `(DAT_00402010 >= 0x1e0 & DAT_00402014 < 0x280)`.
- Decompiler View:** Shows the C-like pseudocode for the main loop, which uses `time_t tVar2` and `srand(iVar1)` to generate random numbers and write them to memory.
- Console:** A scripting console at the bottom.

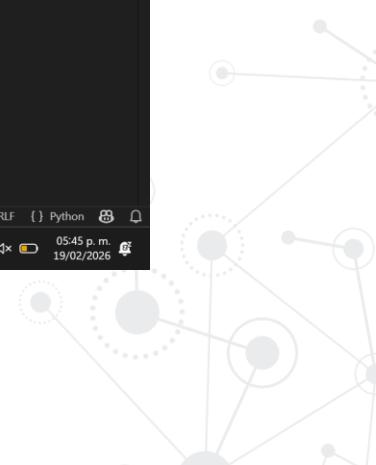
3. SE IDENTIFICO LA PALETA DE COLORES

DAT_00402018				
00402018	00	??	00h	
00402019	32	??	32h	2
0040201a	64	??	64h	d
0040201b	96	??	96h	
0040201c	c8	??	c8h	
0040201d	fa	??	FAh	

4. REPLICACIÓN EN PYTHON

Se implementó:

- Clase que replica exactamente el rand() de MSVC.
- Inicialización con time.time() para imitar srand(time(NULL)).
- Doble ciclo anidado para recorrer cada pixel.
- Escritura de 3 bytes por pixel (RGB).
- Generación del archivo nuevo2026.ppm.



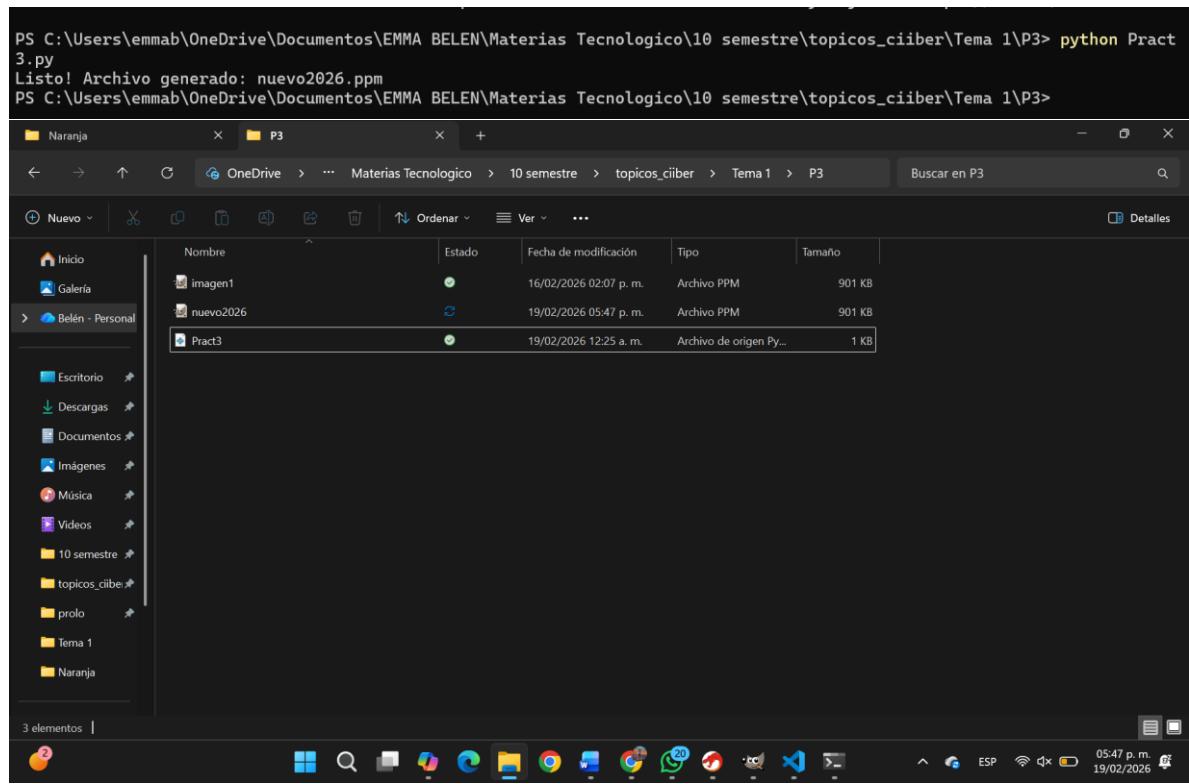
```

File Edit Selection View Go ... ← → ⌂ Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Pract3.py •
C: > Users > emmab > OneDrive > Documentos > EMMA BELEN > Materias Tecnologico > 10 semestre > temas_ciber > Tema 1 > Pract3.py > MSVRand > __init__.py
1 # Equipo Naranja
2 import time
3
4 # Paleta encontrada en DAT_00402018
5 # Son 6 valores que el programa usa para cada canal RGB
6 palette = [0x00, 0x32, 0x64, 0x96, 0xC8, 0xFA]
7
8 # Clase para replicar exactamente el rand() del compilador MSVC
9 # En el binario se usa srand(time(NULL)) y rand() % 6
10 # Aquí implemento el mismo generador lineal congruencial
11 class MSVRand:
12     def __init__(self, seed):
13         # Se guarda el estado inicial (equivalente a srand)
14         self.state = seed & 0xFFFFFFFF
15
16     def rand(self):
17         # Fórmula interna del rand() de MSVC
18         self.state = (self.state * 214013 + 2531011) & 0xFFFFFFFF
19         return (self.state >> 16) & 0xFFFF
20
21     # Semilla basada en tiempo, igual que time(NULL) en C
22     seed = int(time.time())
23     rng = MSVRand(seed)
24
25     # Dimensiones vistas en el desensamblado
26     WIDTH = 0x280 # 640 en decimal
27     HEIGHT = 0x1E0 # 480 en decimal
28

```

Ln 14, Col 39 Spaces: 4 UTF-8 CRLF { } Python 🐍 05:45 p.m. 19/02/2026

5. EJECUTA EL SCRIPT



6. SE ABRE EL ARCHIVO PARA VER LA IMAGEN ALEATORIO

