

M4 - P1: Evasión con ofuscadores de Python (*PyArmor, Nuitka*)

Objetivo

- Comprender cómo las técnicas de ofuscación y compilación dificultan el análisis de código Python.
- Analizar en laboratorio cómo un software benigno ofuscado puede comportarse de manera distinta frente a soluciones de seguridad (AV/EDR).
- Reflexionar sobre el **uso ético** de estas técnicas en **ciberseguridad defensiva, auditorías y análisis forense**.

Entorno recomendado

- Máquina atacante: **Kali** para el desarrollo/ofuscación remota.
- Máquina víctima: **Windows 10/11 (VM)** para ejecutar/analizar.
 - Tener Defender activado para observación y trabajar en red interna aislada.
 - Ejecutar todo como usuario no-administrador salvo para instalar herramientas.

A. Preparación en Windows (*instalar Python, compilador y herramientas*)

1. Instalar Python 3.8–3.11 desde python.org (marca *Add Python to PATH*).
2. Instalar Visual Studio Build Tools (C++ workload) - recomendado para que Nuitka use MSVC. (*Descarga desde Microsoft Visual Studio Build Tools*).
3. Abrir PowerShell (no necesariamente como *admin*) y actualizar *pip*:

```
python -m pip install --upgrade pip setuptools Wheel
```

```
PS C:\Users\Raúl> python -m pip install --upgrade pip setuptools wheel
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pip in c:\users\raúl\appdata\roaming\python\python38\site-packages (25.0.1)
Requirement already satisfied: setuptools in c:\users\raúl\appdata\roaming\python\python38\site-packages (75.3.2)
Requirement already satisfied: wheel in c:\users\raúl\appdata\roaming\python\python38\site-packages (0.45.1)
PS C:\Users\Raúl>
```

4. Instalar *Nuitka* y *PyArmor* (en *user mode* o en un *virtualenv* si prefieres):

<https://pyarmor.readthedocs.io/en/stable/tutorial/getting-started.html>

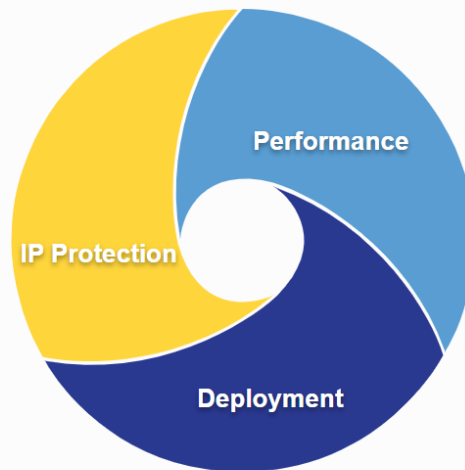
```
> $ cat dist/foo.py

from pytransform import pyarmor
pyarmor(__name__, __file__, b'\x50\x59\x41\x52\x4d\x44\x52\x00'
\x43\x49\x03\x00\x03\x55\x00\x05\x2c\x27\x00\x40\x00', 2)
```

<https://nuitka.net/>

Nuitka the Python Compiler

- A IP Protection**
Compiling your source code for security. If you're a commercial user, ensure that all of your data and files are secured and that there are no readable strings.
- B Performance**
Boost your program runtime and launch performance.
- C Deployment**
Enjoy hassle-free Python deployment with standalone distributions, onefile, PyPI wheels, and more.



```
python -m pip install --user nuitka pyarmor
```

o (recomendado) crear venv:

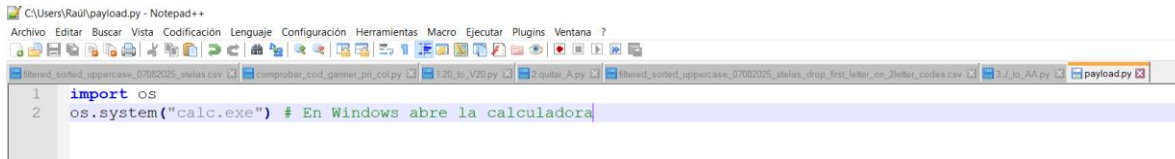
```
python -m venv .venv
.\.venv\Scripts\Activate.ps1
python -m pip install nuitka pyarmor
```

```
PS C:\Users\Raúl> python -m pip install --user nuitka pyarmor
Requirement already satisfied: nuitka in c:\users\raul\appdata\roaming\python\python38\site-packages (2.7.16)
Collecting pyarmor
  Downloading pyarmor-9.1.9-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: ordered-set>=4.1.0 in c:\users\raul\appdata\roaming\python\python38\site-packages (from nuitka) (4.1.0)
Requirement already satisfied: zstandard>=0.15 in c:\users\raul\appdata\roaming\python\python38\site-packages (from nuitka) (0.23.0)
Collecting pyarmor.cli.core~7.6.8 (from pyarmor)
  Downloading pyarmor_cli_core-7.6.8-cp38-none-win_amd64.whl.metadata (3.0 kB)
  Downloading pyarmor-9.1.9-py3-none-any.whl (2.9 MB)
----- 2.9/2.9 MB 8.5 MB/s eta 0:00:00
  Downloading pyarmor_cli_core-7.6.8-cp38-none-win_amd64.whl (413 kB)
Installing collected packages: pyarmor.cli.core, pyarmor
  WARNING: The scripts pyarmor-7.exe, pyarmor-8.exe, pyarmor-auth.exe and pyarmor.exe are installed in 'C:\Users\Raúl\AppData\Roaming\Python\Python38\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed pyarmor-9.1.9 pyarmor.cli.core-7.6.8
PS C:\Users\Raúl>
```

B. Script ejemplo seguro (Windows)

Crear el fichero inofensivo payload.py en C:\Users\Raúl\ofusca\payload.py:

```
import os
os.system("calc.exe") # abre la calculadora
```



C. Ofuscación con PyArmor en Windows

Según la versión de PyArmor que se utilice es posible que los comandos varíen. Pyarmor 8/9 cambió algunos comandos y en este caso se utiliza `pyarmor gen` y ajustes útiles para que luego funcione con Nuitka.

```
PS C:\Users\Raúl> C:\Users\Raúl\AppData\Roaming\Python\Python38\Scripts\pyarmor
INFO Python 3.8.10
INFO Pyarmor 9.1.9 (trial), 000000, non-profits
INFO Platform windows.x86_64
usage: pyarmor [-h] [-v] [-q] [-d] {gen,generate,g,reg,register,r,env,enviren,e,init,i,build,b,cfg,man} ...

optional arguments:
  -h, --help            show this help message and exit
  -v, --version          show version information and exit
  -q, --silent           suppress all normal output
  -d, --debug           generate debug file "pyarmor.debug.log"

The most commonly used pyarmor commands are:
{gen,generate,g,reg,register,r,env,enviren,e,init,i,build,b,cfg,man}
gen (generate, g)      generate obfuscated scripts and required runtime files
reg (register, r)       register Pyarmor or upgrade old Pyarmor license
env (enviren, e)       check and set Pyarmor environments
init (i)               init project in current path
build (b)              obfuscate all the scripts in the project
cfg                    show and config Pyarmor environments
man                    start Pyarmor Man shell
PS C:\Users\Raúl>
```

1. Configuración para evitar protecciones que rompan la compilación, si pyarmor está en PATH:

```
pyarmor cfg restrict_module=0
pyarmor cfg no_cross_protection=1
```

```
PS C:\Users\Raúl> C:\Users\Raúl\AppData\Roaming\Python\Python38\Scripts\pyarmor cfg restrict_module=0
INFO Python 3.8.10
INFO Pyarmor 9.1.9 (trial), 000000, non-profits
INFO Platform windows.x86_64
INFO change option "restrict_module" to new value "0"

-----
Section: builder

Current settings
  restrict_module = 1

Global settings

Local settings
  restrict_module = 0
PS C:\Users\Raúl> C:\Users\Raúl\AppData\Roaming\Python\Python38\Scripts\pyarmor cfg no_cross_protection=1
INFO Python 3.8.10
INFO Pyarmor 9.1.9 (trial), 000000, non-profits
INFO Platform windows.x86_64
INFO change option "no_cross_protection" to new value "1"

-----
Section: builder

Current settings
no option "no_cross_protection"

Global settings

Local settings
  no_cross_protection = 1
PS C:\Users\Raúl>
```

2. Generar la versión ofuscada en carpeta obf/:

```
pyarmor gen -O obf payload.py
```

si tu versión no acepta *gen*, usar la sintaxis equivalente de tu versión:



```
pyarmor obfuscate --restrict 0 --no-bootstrap --package-runtime 0 -O obf payload.py
```

```
PS C:\Users\Raúl> C:\Users\Raúl\AppData\Roaming\Python\Python38\Scripts\pyarmor gen -O obf payload.py
INFO Python 3.8.10
INFO Pyarmor 9.1.9 (trial), 000000, non-profits
INFO Platform windows.x86_64
INFO search inputs ...
INFO find script payload.py
INFO find 1 top resources
INFO start to generate runtime files
INFO target platforms {'windows.amd64'}
INFO write obf\pyarmor_runtime_000000\pyarmor_runtime.pyd
INFO generate runtime files OK
INFO start to obfuscate scripts
INFO process resource "payload"
INFO obfuscating file payload.py
INFO write obf\payload.py
INFO obfuscate scripts OK
PS C:\Users\Raúl>
```

3. Verificar que obf contiene el payload.py ofuscado y la carpeta pytransform (runtime).

```
1 # Pyarmor 9.1.9 (trial), 000000, non-profits, 2025-09-22T22:47:15.295614
2 from pyarmor_runtime_000000 import __pyarmor__
3 __pyarmor__(_name_, _file_, b'PY000000\x00\x03\x08\x00U\r\n\x80\x00\x01\x00\x08\x00\x00\x00\x04\x00\x00\x00@\x00\x00f\x00
4
```

Este equipo > Disco local (C:) > Usuarios > Raúl > obf > pyarmor_runtime_000000

Nombre	Fecha de modificación	Tipo	Tamaño
 __init__.py	22/09/2025 22:47	Python File	1 KB
 pyarmor_runtime.pyd	22/09/2025 22:47	Python Extension ...	619 KB

PyArmor a veces requiere incluir explícitamente su runtime (pytransform) cuando se empaqueta con Nuitka para que funcione correctamente.

D. Combinar PyArmor + Nuitka (compilar el ofuscado a .exe)

Compilar en **Windows** es la forma más fiable. Si se usa `python -m nuitka` se puede evitar problemas de PATH.

1. En PowerShell ir a la carpeta con el código ofuscado

```
cd C:\Users\Raúl\ofusca\obf
```

2. Compilar con Nuitka invocando las opciones para mayor compatibilidad

```
python -m nuitka --standalone --onefile --windows-console-mode=disable payload.py
```

```
PS C:\Users\Raúl> cd obf
PS C:\Users\Raúl\obf> dir

Directorio: C:\Users\Raúl\obf

Mode                LastWriteTime         Length Name
----                -
d-----          22/09/2025   22:47             pyarmor_runtime_000000
-a----          22/09/2025   22:47         1365 payload.py

PS C:\Users\Raúl\obf> python -m nuitka --standalone --onefile --windows-console-mode=disable payload.py
Nuitka-Options: Used command line options:
Nuitka-Options:  --standalone --onefile --windows-console-mode=disable payload.py
Nuitka: Starting Python compilation with:
Nuitka:  Version '2.7.16' on Python 3.8 (flavor 'CPython Official') commercial grade 'not installed'.
Nuitka: Completed Python level compilation and optimization.
Nuitka: Generating source code for C backend compiler.
Nuitka: Running data composer tool for optimal constant value handling.
Nuitka: Running C compilation via Scons.
Nuitka-Scons: Backend C compiler: cl (cl 14.2).
Nuitka-Scons: Backend C linking with 7 files (no progress information available for this stage).
Nuitka-Scons: Compiled 7 C files using clcache with 4 cache hits and 3 cache misses.
Nuitka-Postprocessing: Creating single file from dist folder, this may take a while.
Nuitka-Onefile: Running bootstrap binary compilation via Scons.
Nuitka-Scons: Onefile C compiler: cl (cl 14.2).
Nuitka-Scons: Onefile C linking.
Nuitka-Scons: Compiled 1 C files using clcache with 1 cache hits and 0 cache misses.
Nuitka-Onefile: Using compression for onefile payload.
Nuitka-Onefile: Onefile payload compression ratio (33.40%) size 9184410 to 3067535.
Nuitka-Onefile: Keeping onefile build directory 'payload.onefile-build'.
Nuitka: Keeping dist folder 'payload.dist' for inspection, no need to use it.
Nuitka: Keeping build directory 'payload.build'.
Nuitka: Successfully created '~\obf\payload.exe'.
PS C:\Users\Raúl\obf>
```

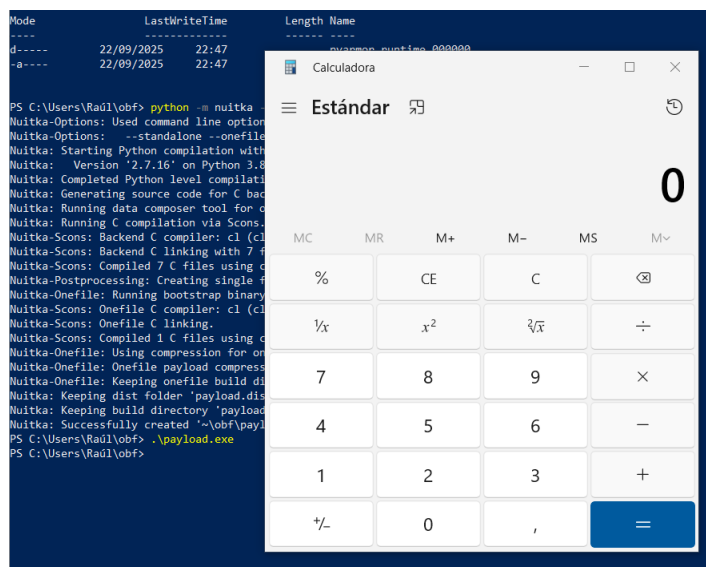
- `--standalone`
Indica a Nuitka que empaquete **todas las dependencias necesarias** (módulos Python, librerías compartidas, DLLs) para que el binario funcione en un equipo que no tenga Python instalado. El resultado habitual sin `--onefile` sería una **carpeta** con el ejecutable + dependencias.
- `--onefile`
Sobre la base de `--standalone`, empaqueta **todo en un único ejecutable** (normalmente comprimido). Al ejecutarlo, ese .exe extrae temporalmente las dependencias en tiempo de ejecución y arranca desde ahí. Esto facilita distribución, pero puede aumentar tiempo de arranque y tamaño.
- `--windows-console-mode=disable`
(opciones similares aparecen como `--windows-disable-console/--windows-console-mode hide/attach/etc.` en docs) indica que el programa **no debe mostrar una ventana de consola** en Windows. Es decir, el ejecutable se comportará como una aplicación “sin consola” (útil para GUIs o para evitar la ventana negra). Puede que la consola aparezca brevemente según la versión de Windows o de Nuitka.
- `payload.py`
Es el script Python de entrada que se quiere compilar.

2. El resultado sería el fichero `payload.exe` (y `payload.dist/` para inspección).

3. Ejecutar:

`.\payload.exe`

Debería abrir la calculadora (*calc.exe*).



Si hay errores, revisar `payload.dist` y buscar `pytransform` y DLLs incluidas; puede que se necesite copiar manualmente `pytransform` (y su `.dll`) dentro de la estructura que `Nuitka` incluye o indicarla con `--include-data-dir`.

Consejos prácticos para errores conocidos

- Si al ejecutar el `.exe` se obtiene `RuntimeError: unauthorized use of script`, probar desactivar restricción y regenerar (`pyarmor cfg restrict_module=0` y `pyarmor gen ...`).
- Si faltan dependencias del runtime `PyArmor`, copiar la carpeta `pytransform` dentro de la salida de `Nuitka` o usar `--include-data-dir` para forzar su inclusión.

E. Pruebas de evasión / evaluación

Objetivo: Es comparar el comportamiento de detección antes y después de ofuscar mediante `VirusTotal`.

1. Prueba base: ejecutar `payload.exe` **sin ofuscar** (compila sin `PyArmor`)
2. Desde la carpeta con `payload.py` original

```
python -m nuitka --standalone --onefile --windows-console-mode=disable ..\payload.py
```

3. Renombrar el fichero para no sobrescribir a *original_payload.exe*

```
PS C:\Users\Raúl\obf> python -m nuitka --standalone --onefile --windows-console-mode-disable ..\payload.py
Nuitka-Options: Used command line options:
Nuitka-Options: --standalone --onefile --windows-console-mode-disable ~\payload.py
Nuitka: Starting Python compilation with:
Nuitka: Version '2.7.16' on Python 3.8 (flavor 'CPython Official') commercial grade 'not installed'.
Nuitka: Completed Python level compilation and optimization.
Nuitka: Generating source code for C backend compiler.
Nuitka: Running data composer tool for optimal constant value handling.
Nuitka: Running C compilation via Scons.
Nuitka-Scons: Backend C compiler: cl (cl 14.2).
Nuitka-Scons: Backend C linking with 6 files (no progress information available for this stage).
Nuitka-Scons: Compiled 6 C files using clcache with 6 cache hits and 0 cache misses.
Nuitka-Postprocessing: Creating single file from dist folder, this may take a while.
Nuitka-Onefile: Running bootstrap binary compilation via Scons.
Nuitka-Scons: Onefile C compiler: cl (cl 14.2).
Nuitka-Scons: Onefile C linking.
Nuitka-Scons: Compiled 1 C files using clcache with 1 cache hits and 0 cache misses.
Nuitka-Onefile: Using compression for onefile payload.
Nuitka-Onefile: Onefile payload compression ratio (33.51%) size 8542780 to 2863053.
Nuitka-Onefile: Keeping onefile build directory 'payload.onefile-build'.
Nuitka: Keeping dist folder 'payload.dist' for inspection, no need to use it.
Nuitka: Keeping build directory 'payload.build'.
Nuitka: Successfully created '~\obf\payload.exe'.
PS C:\Users\Raúl\obf>
```

- Anotar el Hash SHA256 de cada binario, para su posterior comprobación:

Get-FileHash .\payload.exe -Algorithm SHA256

```
PS C:\Users\Raúl\obf> Get-FileHash .\payload.exe -Algorithm SHA256

Algorithm      Hash
-----
SHA256         78CE5A30D9C4574839F52441852137023BB0D815058CFC1821316AA6485253C3
Path
-----
C:\Users\Raúl\obf\payload.exe
```

2. Ejecutar ambos en la VM con **Windows Defender activado** y registrar:
 - ¿Se bloquea/ ejecuta?
 - Revisar Eventos en Event Viewer comprobando los eventos Windows Defender logs.
3. **VirusTotal**: si vas a subir archivos, solo usa archivos benignos y toma en cuenta que los ficheros subidos son accesibles a *vendors*; evitar subir archivos con datos sensibles. VirusTotal explica cómo funcionan los motores y los riesgos operativos.

The screenshot shows the VirusTotal web interface. At the top, a red banner indicates that 13/72 security vendors flagged the file as malicious. Below this, a circular progress indicator shows 13/72. The file details section shows the file name 'payload.exe', size '2.92 MB', and last analysis date 'a moment ago'. The 'DETECTION' tab is selected, showing a table of security vendors and their results. The table has columns for the vendor name, the detection result (with a red circle icon for detections and a green circle icon for undetections), and the specific detection name.

Security vendors' analysis	Do you want to automate checks?		
Bkav Pro	Win64.AIDetect.Malware	CrowdStrike Falcon	Win/malicious_confidence_30% (D)
Cynet	Malicious (score: 100)	Elastic	Malicious (High Confidence)
Google	Detected	Ikarus	Trojan.Python.Agent
Malwarebytes	Malware.AI.1899022665	McAfee Scanner	Trojan.CDD96282B
Microsoft	Program.Win32/Wacacem.Cml	SecureAge	Malicious
Skyhigh (SWG)	BehavesLike.Win64.Emotet.vc	Symantec	ML.Attribute.HighConfidence
Trapsine	Malicious.moderate.ml.score	Acronis (Static ML)	Undetected
AhnLab-V3	Undetected	Alibaba	Undetected
AliCloud	Undetected	ALYac	Undetected
Antiy-AVL	Undetected	Arcabit	Undetected

4. Observar los resultados obtenidos con VirusTotal, la ofuscación suele **reducir la detección estática** pero no elimina la detección por **comportamiento/EDR** (Defender analiza acciones del proceso: creación de procesos, sockets, etc.).

G. Notas finales y advertencias (ética + seguridad)

- **Solo** usa muestras benignas en tests y en entornos aislados; nunca pruebes payloads maliciosos fuera de laboratorios controlados con autorización.
- Subir archivos a **VirusTotal** comparte muestras con terceros; usar con precaución.

H. Fuentes / referencias

1. **Nuitka** - User manual / tutorial (uso `--windows-console-mode`, invocación `python -m nuitka`). <https://nuitka.net/user-documentation/user-manual.html>
2. **PyArmor** — documentación sobre opciones, manejo de runtimes y compatibilidad con terceros. <https://pyarmor.readthedocs.io/en/latest/how-to/third-party.html>
3. Integración PyArmor + Nuitka – ver discusiones en stack overflow sobre copiar `pytransform` y desactivar `restrict`.
4. **Microsoft Defender** - behavior monitoring / EDR (*por qué el comportamiento puede ser detectado aunque el binario esté ofuscado*). <https://learn.microsoft.com/en-us/defender-endpoint/behavior-monitor>
5. **VirusTotal** - cómo funciona y riesgos de subir muestras. <https://docs.virustotal.com/docs/how-it-works>

9. Mitigaciones reales

Los defensores pueden usar:

- **EDR con análisis de comportamiento** (detecta qué hace el programa, no cómo está escrito).
- **Sandboxing dinámico** para ejecutar y observar acciones reales.
- **Monitorización de tráfico saliente** y procesos sospechosos.
- **Reglas YARA** para detectar patrones en binarios ofuscados.