

Forum:[Hack Forums](#)

Thread: [PowerShell Pro Obfuscator & Virtualizer - Protect your PowerShell Scripts](#)

PELock - July 28, 2026, 5:34 pm

PowerShell Pro Obfuscator protects proprietary .ps1 scripts with renaming, polymorphic string and integer encryption, control-flow flattening, finite-state automata (FSA), a VM engine, self-defending integrity checks, a protection linker, and anti-debugging probes.

Available as

[Online PowerShell Obfuscator](#)

[Windows client with UI & separate CLI version](#) (both Linux/Mono compatible)

[API packages with usage examples for Python, JS, C#, Rust, PHP](#) (all sources on [GitHub](#))

PowerShell is a cross-platform shell and scripting language built on .NET. It is widely used for automation, configuration, DevOps pipelines, and endpoint management on Windows and Linux.

Why PowerShell scripts needs obfuscation?

Scripts are typically distributed as plain .ps1 files or embedded in modules. That convenience means anyone with file access can read the full logic, hunt for credentials or API keys in strings, and steal your algorithms unless you take extra steps to hide intent.

Obfuscation strategies

PowerShell Pro Obfuscator comes with many advanced obfuscation, virtualization & protection strategies. You can easily tune protection versus size and performance.

Powerful obfuscation

PowerShell Pro Obfuscator uses state-of-the-art obfuscation strategies such as polymorphic string encryption, integers & floats encryption, and decoy noise. The result conceals literals and structure while preserving tested runtime behaviour.

Code virtualization

Selected statements are lifted into a random generated VM engine opcodes with shuffled switch cases, decoy opcodes, and an obfuscated dispatcher loop. Analysts must interpret the virtual machine instead of reading plain PowerShell.

Finite-state automata (FSA)

Finite-state automata (FSA) obfuscation rewrites linear PowerShell statement blocks into dual-state automata with opaque schedulers and shuffled dispatch handlers. Instead of reading code top to bottom, analysts must follow numeric states, transition tables, and decoy paths to reconstruct the original order.

Anti-debugging

Anti-debugging protection inserts polymorphic probes that detect attached debuggers, PowerShell breakpoints, debug preference and tracing modes, and related host signals. When a check fires, the obfuscated script exits silently instead of revealing protected logic under interactive analysis.

Self-integrity checks

A bootstrap probe verifies on-disk script shape (function counts and integrity tokens) and sets a tamper key when the file no longer matches the obfuscated build. String decryptors consume that key, so patched scripts return garbage instead of plaintext. This self-defending layer raises the cost of casual deobfuscation and file edits.

Protection linker

A late pass wires honeypot resolvers, bogus helper calls with random arguments, and shallow-stack tripwires that stay active only before bootstrap completes. Extracted snippets keep noisy call surfaces that look real under static review. Normal execution after a successful integrity check remains unchanged.

PowerShell code after obfuscation

Command line version (CLI)

Obfuscator uses novel obfuscation techniques like FSA transformations and was heavily tested against all kinds of scripts and edge cases.

So, at least give it a try :)

PELock - July 30, 2026, 1:50 pm

Engine history

v1.0.2 - 30.07.2026 Self-defending rewritten — opaque body-digest probe (FNV over the normalized on-disk script with the probe span excluded) instead of source-shape / function-count checks

Integrity / string-decrypt keys are derived from the live body digest; content edits poison those keys

Detect-debugger no longer clears healthy self-defending keys when both options are enabled

v1.0.1 - 29.07.2026 Fixed try {} finally {} construct to be compatible with PS5 and PS7

Things moving fast - a native PowerShell module on PowerShell Gallery

<https://www.powershellgallery.com/packages/PowerShellProObfuscator/1.0.0>

It let you obfuscate PowerShell scripts directly from PowerShell code

Code

```
#####  
#  
# PowerShell Pro Obfuscator WebApi interface usage example.  
#  
# In this example we will obfuscate sample source with custom options.  
#  
# Version      : v1.0.0  
# Language     : PowerShell  
# Author       : Bartosz Wójcik  
# Web page     : https://www.pelock.com  
#  
#####  
  
# install once:  
# Install-Module -Name PowerShellProObfuscator -Scope CurrentUser  
  
Import-Module PowerShellProObfuscator  
  
#  
# source code in PowerShell format  
#  
$scriptSourceCode = @'  
function Get-Greeting {  
    param([string]$Name)  
    Write-Host "Hello World from $Name!"  
}  
Get-Greeting "PowerShell Pro Obfuscator"  
'@  
  
#  
# select obfuscation strategies explicitly (only listed switches are enabled)  
#  
try  
{  
    $output = Invoke-PowerShellProObfuscate `  
        -Source $scriptSourceCode `  
        -ActivationCode 'ABCD-ABCD-ABCD-ABCD' `  
        -EnableCompression:$false `  
        -SelfDefending `  
        -ProtectionLinker `  
        -RenameVariables `  
        -RenameParameters `  
        -RenameFunctions `  
        -ShuffleFunctions `  
        -ControlFlowFlatten `  
        -StateMachine `  
        -VmStrategy `  
        -EncryptIntegers `  
        -SplitStrings `  
        -EncryptStrings `  
        -IntegersToArrays `  
        -FloatsToArrays `
```

- InsertDeadCode `
- ComplexifyBooleans `
- IntegersToFloating `
- EncryptFloating `
- DecoyFunctions `
- DetectDebugger `
- FakeDotSourceMarkers `
- OpaqueBranches `
- ScriptblockDecoys `
- LiteralPadding `
- ReflectInvokeCommands `
- StringCharArrayVault `
- TryFinallyNoise `
- AffineIntegerMask `
- EventStub `
- RemoveComments

```
Write-Host $output
}
catch
{
    throw "An error occurred while trying to obfuscate the code: $_"
}
```

PELock - September 11, 2026, 12:43 pm

PowerShell Pro Obfuscator Extension for Visual Studio Code

Install for VS Code

<https://marketplace.visualstudio.com/items?itemName=PELock.powershell-pro-obfuscator>

Install for VS Code compatible editors (Cursor etc.)

<https://open-vsx.org/extension/PELock/powershell-pro-obfuscator>

Extension source code

<https://github.com/PELock/PowerShell-Pro-Obfuscator-VSCode-Extension>