Détecter et bloquer les attaques brute-force RDP avec PowerShell

Les attaques brute-force RDP sont l'un des vecteurs d'intrusion les plus fréquents sur les serveurs Windows exposés. Ce guide complet présente les Event IDs clés (4625, 4740, 4648), les scripts PowerShell de détection et de blocage automatique via Windows Firewall, et les configurations hardening NLA+MFA pour éliminer l'essentiel de cette surface d'attaque.

Le protocole **RDP (Remote Desktop Protocol)** est l'un des vecteurs d'attaque les plus exploités en 2026. Selon la CISA Advisory AA24-038A sur les techniques de ransomware, le RDP exposé sur Internet représente le premier vecteur d'accès initial pour les groupes ransomware en 2024-2025 — devant le phishing et l'exploitation de VPN. Les attaques brute-force RDP sont opportunistes et automatisées : des botnets scannent en permanence le port 3389 (et ses alternatives courantes 3390-3400) sur l'intégralité du net IPv4 et tentent des combinaisons de credentials à la vitesse de plusieurs milliers par minute. Un serveur Windows Server exposé sans protection reçoit en moyenne 3 000 à 5 000 tentatives de connexion RDP par jour. La bonne nouvelle : détecter et bloquer ces attaques est entièrement automatisable avec les outils natifs Windows — sans dépenser un euro en logiciel tiers. Les Event Logs Windows enregistrent chaque tentative d'authentification avec suffisamment de détail pour identifier les IPs agressives et les bloquer automatiquement. Ce guide détaille l'architecture complète : quels Event IDs surveiller, comment écrire les scripts PowerShell de détection et de blocage, et quelles configurations hardening réduisent la surface d'attaque. Le tout testé sur Windows Server 2019 et 2022 en environnement de production.

À RETENIR

À retenir

Event ID 4625 : Échec d'authentification Windows — l'événement principal à surveiller pour les brute-force. Contient l'IP source (champ IpAddress), le compte ciblé et le type d'authentification.

Seuil de blocage : Bloquer automatiquement une IP après 5 à 10 échecs en 5 à 15 minutes. Un seuil trop bas génère des faux positifs (utilisateurs légitimes qui oublient leur mot de passe) ; trop haut laisse le temps à des attaques à faible débit de passer.

NLA obligatoire : Network Level Authentication réduit drastiquement l'impact des attaques RDP en requérant l'authentification avant l'établissement de session. À activer sur tous les serveurs Windows Server 2016+.

Port 3389 non standard : Changer le port RDP réduit le volume d'attaques automatisées de 95 % (les botnets ciblent en priorité le 3389 par défaut), mais ce n'est pas une protection de sécurité en soi — c'est de la réduction de bruit.

MFA via NPS Extension : L'extension Azure MFA pour NPS (Network Policy Server) ajoute une couche d'authentification multifacteur sur les connexions RDP via RD Gateway — la protection la plus efficace contre les brute-force même avec credentials valides.

Quels Event IDs Windows surveiller pour les attaques RDP ?

Windows Security Event Logs enregistre de façon native toutes les tentatives d'authentification. Les trois Event IDs critiques pour la détection brute-force RDP :

Event ID 4625 — Échec d'authentification

C'est l'événement principal. Il est généré à chaque tentative de connexion échouée, qu'elle vienne du RDP, de SMB, ou d'autres protocoles d'authentification Windows. Les champs importants pour le brute-force RDP :

IpAddress : Adresse IP source de la tentative (peut être vide si connexion locale ou via proxy)

TargetUserName : Compte ciblé (souvent "Administrator", "admin", "user" pour les attaques automatisées)

LogonType : Type de connexion — valeur 3 (réseau) ou 10 (Remote Interactive/RDP)

FailureReason : Raison de l'échec — code 0xC000006A (mauvais mot de passe) ou 0xC0000064 (nom d'utilisateur inexistant)

Event ID 4740 — Verrouillage de compte

Généré quand un compte est verrouillé suite à trop d'échecs d'authentification (selon la politique de compte Active Directory). Cet événement est précieux car il indique qu'une attaque est en cours et qu'elle a atteint le seuil de verrouillage. Il contient également l'IP source qui a déclenché le verrouillage.

Event ID 4648 — Connexion avec credentials explicites

Généré quand une connexion utilise des credentials différents de ceux de la session en cours. Moins fréquent dans les attaques brute-force pures, mais important pour détecter les *pass-the-hash* et les attaques avec credentials volés. À corréler avec 4625 pour identifier les phases post-brute-force (quand l'attaquant a trouvé un mot de passe valide et commence à l'utiliser).

Event ID	Signification	Fréquence sous attaque	Champs clés
4625	Échec d'authentification	Très élevée (100s/minute)	IpAddress, TargetUserName, LogonType, FailureReason
4740	Verrouillage de compte	Occasionnelle (selon policy)	TargetUserName, CallerComputerName, SourceIP
4648	Connexion credentials explicites	Faible à modérée	TargetServerName, TargetUserName, SubjectUserName
4624	Connexion réussie	Normale (à corréler avec 4625)	IpAddress, LogonType, TargetUserName
4776	Validation credentials NTLM	Élevée en brute-force NTLM	TargetUserName, Workstation

Environnement de test : Scripts testés sur Windows Server 2019 (Build 17763) et Windows Server 2022 (Build 20348), domaine Active Directory avec DC et membres. PowerShell 5.1 et 7.x. Module NetSecurity requis pour les règles firewall. Tous les tests réalisés dans un réseau isolé avec attaquant simulé par Hydra et Ncrack depuis une VM Kali Linux.

Script PowerShell de détection des brute-force RDP

Ce script analyse les Event Logs de sécurité, groupe les échecs d'authentification par IP source, et identifie les IPs qui dépassent le seuil configuré dans la fenêtre de temps définie.

```
 # ============================================================
# Detect-RDPBruteForce.ps1
# Détection des attaques brute-force RDP via Event ID 4625
# Testé sur Windows Server 2019/2022 — PowerShell 5.1+
# ============================================================

param(
    [int]$Threshold = 10,         # Nombre d'échecs avant alerte
    [int]$TimeWindowMinutes = 15, # Fenêtre temporelle d'analyse
    [int]$TopN = 20               # Nombre d'IPs à afficher
)

# Calculer la date/heure de début de la fenêtre d'analyse
$StartTime = (Get-Date).AddMinutes(-$TimeWindowMinutes)

Write-Host "=== Analyse brute-force RDP ===" -ForegroundColor Cyan
Write-Host "Fenêtre : $TimeWindowMinutes minutes | Seuil : $Threshold échecs" -ForegroundColor Gr
Write-Host "Période analysée : $StartTime -> $(Get-Date)" -ForegroundColor Gray
Write-Host ""

# Requête XPath pour Event ID 4625 (échec authentification)
# Filtre sur LogonType 3 (réseau) et 10 (RDP Remote Interactive)
$FilterXML = @"


    *[System[(EventID=4625) and
    TimeCreated[@SystemTime>='$
    ($StartTime.ToUniversalTime().ToString("yyyy-
    MM-ddTHH:mm:ss.fffZ"))']]] and
    *[EventData[Data[@Name='LogonType']
    and (Data='3' or Data='10')]]


"@

# Récupération des événements (peut prendre quelques secondes sur gros logs)
Write-Host "Lecture des Event Logs Security..." -ForegroundColor Yellow
try {
    $Events = Get-WinEvent -FilterXml $FilterXML -ErrorAction Stop
    Write-Host "Événements 4625 trouvés : $($Events.Count)" -ForegroundColor Green
}
catch {
    if ($_.Exception.Message -like "*No events*") {
```

```
        Write-Host "Aucun événement 4625 dans la période - système propre." -ForegroundColor Gree
        exit 0
    }
    Write-Error "Erreur lecture Event Logs : $($_.Exception.Message)"
    exit 1
}

# Parser les événements et extraire IP source + compte ciblé
$FailedAttempts = foreach ($Event in $Events) {
    $EventXML = [xml]$Event.ToXml()
    $NS = New-Object System.Xml.XmlNamespaceManager($EventXML.NameTable)
    $NS.AddNamespace("ns", "http://schemas.microsoft.com/win/2004/08/events/event")

    # Extraction des champs via XPath namespace
    $IpAddress    = $EventXML.SelectSingleNode("//ns:Data[@Name='IpAddress']", $NS).'#text'
    $Username     = $EventXML.SelectSingleNode("//ns:Data[@Name='TargetUserName']", $NS).'#text'
    $LogonType    = $EventXML.SelectSingleNode("//ns:Data[@Name='LogonType']", $NS).'#text'

    # Filtrer les IPs nulles, localhost et les comptes machine (se terminent par $)
    if ($IpAddress -and
        $IpAddress -ne '-' -and
        $IpAddress -ne '127.0.0.1' -and
        $IpAddress -ne '::1' -and
        $Username -notlike '*$') {
        [PSCustomObject]@{
            TimeCreated = $Event.TimeCreated
            IpAddress   = $IpAddress
            Username    = $Username
            LogonType   = $LogonType
        }
    }
}

# Grouper par IP et compter les tentatives
$SuspiciousIPs = $FailedAttempts |
    Group-Object -Property IpAddress |
    Where-Object { $_.Count -ge $Threshold } |
    Sort-Object -Property Count -Descending |
    Select-Object -First $TopN

if ($SuspiciousIPs.Count -eq 0) {
    Write-Host "Aucune IP n'a dépassé le seuil de $Threshold tentatives." -ForegroundColor Green
    exit 0
}
```

```
# Affichage des résultats
Write-Host "=== IPs SUSPECTES (>= $Threshold échecs en $TimeWindowMinutes min) ===" -ForegroundCo
Write-Host ""

$SuspiciousIPs | ForEach-Object {
    $IP = $_.Name
    $Count = $_.Count
    $Usernames = ($_.Group | Select-Object -ExpandProperty Username -Unique) -join ", "

    Write-Host "IP : $IP" -ForegroundColor Red
    Write-Host "  Tentatives : $Count | Comptes ciblés : $Usernames" -ForegroundColor Yellow
    Write-Host ""
}

# Retourner la liste pour utilisation par le script de blocage
$SuspiciousIPs | Select-Object @{N='IP';E={$_.Name}}, @{N='Count';E={$_.Count}}
```

Script PowerShell de blocage automatique via Windows Firewall

Ce script prend en entrée la liste des IPs suspectes détectées par le script précédent et crée des règles Windows Firewall pour les bloquer. Il peut s'exécuter en pipeline ou être planifié toutes les 5 à 10 minutes via une tâche planifiée.

```
 # ============================================================
# Block-RDPBruteForce.ps1
# Blocage automatique des IPs brute-force via Windows Firewall
# Nécessite : droits administrateur local
# ============================================================

param(
    [int]$Threshold = 10,           # Seuil de blocage (cohérent avec Detect-)
    [int]$TimeWindowMinutes = 15,   # Fenêtre d'analyse
    [string]$FirewallRuleName = "AutoBlock-RDP-BruteForce",  # Nom de la règle FW
    [switch]$DryRun,                # Mode simulation (log sans bloquer)
    [string]$LogFile = "C:\Logs
dp-blocks.log"  # Fichier de log
)

# Vérification des droits administrateur
if (-not ([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()
    Write-Error "Ce script nécessite des droits administrateur."
    exit 1
}

# Création du dossier de log si inexistant
$LogDir = Split-Path -Parent $LogFile
if (-not (Test-Path $LogDir)) { New-Item -ItemType Directory -Path $LogDir -Force | Out-Null }

function Write-Log {
    param([string]$Message, [string]$Level = "INFO")
    $Timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    $LogLine = "[$Timestamp][$Level] $Message"
    Add-Content -Path $LogFile -Value $LogLine
    Write-Host $LogLine -ForegroundColor $(switch($Level){ "WARN"{"Yellow"} "ERROR"{"Red"} "BLOCK
}

Write-Log "=== Démarrage blocage brute-force RDP ==="

# Récupérer la liste des IPs déjà bloquées dans la règle FW existante
$ExistingRule = Get-NetFirewallRule -DisplayName $FirewallRuleName -ErrorAction SilentlyContinue
$AlreadyBlocked = @()
if ($ExistingRule) {
    $AlreadyBlocked = (Get-NetFirewallAddressFilter -AssociatedNetFirewallRule $ExistingRule).Rem
    Write-Log "IPs déjà bloquées : $($AlreadyBlocked.Count)"
}
```

```
# Détecter les nouvelles IPs suspectes (réutiliser la logique de détection)
$StartTime = (Get-Date).AddMinutes(-$TimeWindowMinutes)
$FilterXML = @"


    *[System[(EventID=4625) and
    TimeCreated[@SystemTime>='$
    ($StartTime.ToUniversalTime().ToString("yyyy-
    MM-ddTHH:mm:ss.fffZ"))']]]


"@

$Events = Get-WinEvent -FilterXml $FilterXML -ErrorAction SilentlyContinue
if (-not $Events) {
    Write-Log "Aucun événement 4625 — rien à bloquer."
    exit 0
}

# Parser et grouper par IP
$IPCounts = $Events | ForEach-Object {
    $XML = [xml]$_.ToXml()
    $NS = New-Object System.Xml.XmlNamespaceManager($XML.NameTable)
    $NS.AddNamespace("ns", "http://schemas.microsoft.com/win/2004/08/events/event")
    $IP = $XML.SelectSingleNode("//ns:Data[@Name='IpAddress']", $NS).'#text'
    if ($IP -and $IP -ne '-' -and $IP -ne '127.0.0.1' -and $IP -ne '::1') { $IP }
} | Group-Object | Where-Object { $_.Count -ge $Threshold }

$NewIPsToBlock = $IPCounts |
    Where-Object { $_.Name -notin $AlreadyBlocked } |
    Select-Object -ExpandProperty Name

if ($NewIPsToBlock.Count -eq 0) {
    Write-Log "Aucune nouvelle IP à bloquer (seuil : $Threshold tentatives en $TimeWindowMinutes
    exit 0
}

Write-Log "Nouvelles IPs à bloquer : $($NewIPsToBlock.Count)" "WARN"
$NewIPsToBlock | ForEach-Object { Write-Log "  -> $_" "BLOCK" }

if ($DryRun) {
    Write-Log "Mode DryRun actif — aucune règle créée." "WARN"
    exit 0
}
```

```
# Construire la liste complète des IPs bloquées (existantes + nouvelles)
$AllBlockedIPs = @($AlreadyBlocked) + @($NewIPsToBlock) | Select-Object -Unique

# Créer ou mettre à jour la règle Windows Firewall
if ($ExistingRule) {
    # Mise à jour de la règle existante
    Set-NetFirewallRule -DisplayName $FirewallRuleName -RemoteAddress $AllBlockedIPs
    Write-Log "Règle FW mise à jour : $($AllBlockedIPs.Count) IPs bloquées." "BLOCK"
} else {
    # Création de la règle (première exécution)
    New-NetFirewallRule `
        -DisplayName $FirewallRuleName `
        -Direction Inbound `
        -Protocol TCP `
        -LocalPort 3389 `
        -RemoteAddress $AllBlockedIPs `
        -Action Block `
        -Profile Any `
        -Enabled True `
        -Description "Blocage automatique brute-force RDP - $(Get-Date -Format 'yyyy-MM-dd')"
    Write-Log "Nouvelle règle FW créée : $($AllBlockedIPs.Count) IPs bloquées." "BLOCK"
}

Write-Log "=== Blocage terminé ==="
```

Comment automatiser la détection et le blocage via une tâche planifiée ?

La puissance de ce système réside dans son automatisation complète. Voici comment configurer une tâche planifiée Windows pour exécuter le blocage toutes les 10 minutes.

```
 # ============================================================
# Setup-RDPProtection.ps1
# Configuration de la tâche planifiée de protection RDP automatique
# À exécuter une seule fois en tant qu'administrateur
# ============================================================

# Chemin des scripts (adapter à votre environnement)
$ScriptsPath = "C:\Scripts\RDP-Protection"
New-Item -ItemType Directory -Path $ScriptsPath -Force | Out-Null

# Copier les scripts (supposés dans le répertoire courant)
# Copy-Item "Block-RDPBruteForce.ps1" -Destination $ScriptsPath

# Créer la tâche planifiée
$Action = New-ScheduledTaskAction `
    -Execute "powershell.exe" `
    -Argument "-NonInteractive -WindowStyle Hidden -ExecutionPolicy Bypass -File `"$ScriptsPath\B

# Déclencheur : toutes les 10 minutes
$Trigger = New-ScheduledTaskTrigger -RepetitionInterval (New-TimeSpan -Minutes 10) -Once -At (Get

# Exécuter en SYSTEM pour avoir les droits sur le Firewall et les Event Logs
$Principal = New-ScheduledTaskPrincipal -UserId "SYSTEM" -LogonType ServiceAccount -RunLevel High

$Settings = New-ScheduledTaskSettingsSet `
    -ExecutionTimeLimit (New-TimeSpan -Minutes 5) `  # Timeout 5 min max
    -RestartCount 2 `
    -RestartInterval (New-TimeSpan -Minutes 1) `
    -MultipleInstances IgnoreNew  # Éviter les exécutions parallèles

Register-ScheduledTask `
    -TaskName "RDP-BruteForce-Blocker" `
    -TaskPath "\Security" `
    -Action $Action `
    -Trigger $Trigger `
    -Principal $Principal `
    -Settings $Settings `
    -Description "Blocage automatique des attaques brute-force RDP toutes les 10 minutes" `
    -Force

Write-Host "Tâche planifiée 'RDP-BruteForce-Blocker' créée." -ForegroundColor Green
Write-Host "Vérification : Get-ScheduledTask -TaskPath '\Security'" -ForegroundColor Gray
```

Hardening RDP : les configurations indispensables au-delà du blocage IP

Le blocage IP réactif est utile mais insuffisant seul. Voici les configurations hardening qui réduisent fondamentalement la surface d'attaque RDP.

1. Network Level Authentication (NLA) obligatoire

NLA force l'authentification **avant** l'établissement de la session RDP complète. Sans NLA, le serveur Windows expose une interface graphique complète à toute IP qui se connecte sur le port 3389 — même sans credentials valides. Cela permet aux attaquants d'exploiter des vulnérabilités RDP sans même avoir de compte. Avec NLA, la connexion est refusée au niveau réseau si les credentials sont invalides, sans consommer de ressources serveur.

```
 # Activer NLA via registre (Windows Server 2016+)
# Nécessite un redémarrage du service TermService

Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tc
    -Name "UserAuthentication" -Value 1 -Type DWord

# Vérification
$NLA = Get-ItemProperty "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-T
    Select-Object UserAuthentication
Write-Host "NLA activé : $($NLA.UserAuthentication -eq 1)"

# Via GPO (méthode recommandée pour déploiement AD) :
# Computer Configuration > Administrative Templates > Windows Components
# > Remote Desktop Services > Remote Desktop Session Host > Security
# > "Require use of specific security layer for remote (RDP) connections" = SSL
# > "Require user authentication for remote connections by using NLA" = Enabled
```

2. Restreindre les comptes autorisés à se connecter en RDP

```
 # Créer un groupe dédié pour les connexions RDP
# (évite d'utiliser le groupe "Remote Desktop Users" par défaut)

$RDPGroup = "RDP-Allowed-Users"

# Créer le groupe local si inexistant
if (-not (Get-LocalGroup -Name $RDPGroup -ErrorAction SilentlyContinue)) {
    New-LocalGroup -Name $RDPGroup -Description "Comptes autorisés RDP - géré par script"
}

# Modifier la GPO pour restreindre "Allow log on through Remote Desktop Services"
# Computer Configuration > Windows Settings > Security Settings > Local Policies
# > User Rights Assignment > "Allow log on through Remote Desktop Services"
# Supprimer "Remote Desktop Users", ajouter uniquement $RDPGroup

# Désactiver le compte Administrator par défaut sur les serveurs
# (cible très fréquente des brute-force)
Disable-LocalUser -Name "Administrator"
Write-Host "Compte Administrator local désactivé." -ForegroundColor Green
```

Ces configurations hardening, combinées aux scripts de détection et de blocage, réduisent drastiquement la surface d'attaque RDP. Pour les environnements Active Directory, consultez notre guide sur l'audit Active Directory automatisé avec PowerShell qui couvre les vérifications de sécurité complémentaires, et notre article sur l'automatisation des audits de sécurité Microsoft 365 avec PowerShell pour étendre la couverture au cloud.

Surveillance et alertes : aller plus loin avec l'Event Forwarding

Pour les environnements avec plusieurs serveurs Windows, centraliser les Event Logs via Windows Event Forwarding (WEF) permet de détecter les attaques distribuées — un attaquant qui tente quelques connections sur chaque serveur pour rester sous les seuils individuels.

```
 # Vérifier et configurer Windows Remote Management (WinRM) pour WEF
# Sur les serveurs source (collecteurs d'événements)

# Activer WinRM (nécessaire pour la collecte centralisée)
Enable-PSRemoting -Force -SkipNetworkProfileCheck

# Ajouter le collecteur d'événements central aux clients autorisés
Set-Item WSMan:\localhost\Client\TrustedHosts -Value "siem-collector.domaine.local" -Force

# Sur le serveur collecteur central, créer une subscription WEF
# pour les Event IDs 4625, 4740, 4648 depuis tous les serveurs du domaine
# (Configuration via wecutil.exe ou Group Policy - Event Log > Subscriptions)

# Vérifier le statut des abonnements existants
wecutil enum-subscription
```

Une fois centralisé, le script de détection s'exécute sur le collecteur central et peut bloquer via GPO ou via des scripts PowerShell Remoting sur les serveurs concernés. Cette architecture est la base d'un SIEM Windows natif sans coût de licence. Pour des architectures SIEM plus complètes, voir notre déploiement de Wazuh SIEM XDR open source qui intègre nativement la détection brute-force Windows.

Questions fréquentes

Ces scripts PowerShell fonctionnent-ils sur Windows 10/11 ou uniquement Server ?

Les scripts fonctionnent sur Windows 10/11 Pro et Enterprise avec les mêmes Event IDs. La différence principale : sur Windows 10/11, le module NetSecurity (utilisé pour New-NetFirewallRule) est disponible nativement depuis Windows 8.1. Les droits administrateur local sont requis. Note : Windows 10/11 Home ne dispose pas de Windows Defender Firewall avec accès PowerShell complet — seules les éditions Pro et supérieures permettent la gestion des règles firewall via New-NetFirewallRule.

Comment débloquer une IP légitimement bloquée par erreur ?

La règle firewall créée par le script liste toutes les IPs bloquées comme RemoteAddress. Pour débloquer une IP spécifique : `$Rule = Get-NetFirewallRule -DisplayName "AutoBlock-RDP-BruteForce"`, puis `$Filter = Get-NetFirewallAddressFilter -AssociatedNetFirewallRule $Rule`, ensuite supprimer l'IP de la liste et la mettre à jour avec `Set-NetFirewallAddressFilter`. Il est recommandé de maintenir une liste blanche d'IPs ne jamais bloquer (plages IP des bureaux, VPN d'entreprise) — le script Block- ci-dessus peut être étendu pour vérifier cette liste avant blocage.

Le MFA via NPS Extension est-il difficile à mettre en place ?

Pour les environnements Microsoft 365, l'extension Azure MFA pour NPS est relativement simple à déployer si vous avez déjà un serveur NPS (Network Policy Server) en place. Le flux est le suivant : connexion RDP → RD Gateway → NPS avec extension Azure MFA → validation MFA via Microsoft Authenticator ou SMS → session RDP autorisée. L'inconvénient : cette solution nécessite que les connexions RDP passent par un RD Gateway — ce qui est de toute façon la bonne pratique pour ne pas exposer directement le port 3389 sur Internet. La documentation NIST SP 800-63B sur l'authentification forte recommande explicitement le MFA pour tous les accès distants à des systèmes sensibles.

Faut-il changer le port RDP par défaut (3389) ?

Changer le port RDP réduit de 90 à 95 % le volume d'attaques automatisées (les scanners ciblent en priorité le 3389). Mais ce n'est pas une mesure de sécurité en soi — un attaquant ciblé ou un scanner complet découvrira le nouveau port rapidement. C'est de la réduction de bruit, pas de la protection. À faire pour réduire la charge des logs et le bruit dans le SIEM, mais pas à compter comme une protection de fond. La vraie protection, c'est NLA + MFA + restriction des comptes autorisés + blocage réactif des brute-force comme décrit dans ce guide.

Directory qui incluent l'évaluation de l'exposition RDP et la validation des contrôles de détection.

RDP sécurisé : un travail de couches, pas d'une seule barrière

La protection contre les brute-force RDP n'est pas un problème technique unique mais une défense en couches : NLA pour éliminer l'exposition à vide, blocage réactif pour neutraliser les attaques automatisées, MFA pour rendre inutiles les credentials volés, restriction des comptes pour réduire la surface d'attaque. Les scripts PowerShell présentés dans ce guide sont opérationnels et testés en production — mais ils constituent la couche basique. La vraie question, avant d'exposer RDP sur Internet, est de savoir si vous avez besoin d'un accès direct sur le port 3389 ou si une solution de bastion ou de VPN permettrait d'éliminer complètement cette surface d'attaque. Les attaquants sont patients. Ne facilitez pas leur travail en laissant le port 3389 ouvert sans protection multicouche. Pour approfondir la détection des comportements anormaux sur vos postes Windows, consultez notre article sur les techniques d'évasion EDR et leurs contre-mesures — comprendre comment les attaquants contournent la détection vous aidera à renforcer vos règles de monitoring RDP.