CHECKLIST SÉCURITÉ ANC

Checklist Durcissement DNSSEC 2026

Ayi NEDJIMI Consultants — ayinedjimi-consultants.fr

Version 1.0 · Mars 2026 · 36097 mots · Format ANC

CHECKLIST DURCISSEMENT DNSSEC 2026

🛡️ AYI NEDJIMI CONSULTANTS (ANC)

Version : 1.0
Date : 2026-06-03
Classification : CONFIDENTIEL
Auteur : ANC Security Team
Scope : Zones DNS autoritatives, résolveurs récursifs internes, registrars, HSM

📋 Légende d'Évaluation

Symbole	Statut	Description
✅	Conforme	Le contrôle est correctement implémenté
❌	Non-Conforme	Le contrôle n'est pas implémenté ou défaillant
⚠️	Partiel	Le contrôle est partiellement implémenté
N/A	Non Applicable	Le contrôle ne s'applique pas à cet environnement

Niveaux de Criticité

Niveau	Icône	Description	Priorité
Critique	🔴	Zone bogué ou compromission clé — résolution DNS bloquée pour tous les resolvers validants	P0 — Immédiat
Élevé	🟠	Risque élevé d'usurpation DNS, zone walking ou rollover défaillant	P1 — < 7 jours
Moyen	🟡	Risque modéré, non-conformité réglementaire ou lacune monitoring	P2 — < 30 jours
Faible	🟢	Bonnes pratiques, optimisation et préparation future	P3 — < 90 jours

Pré-requis outillage

Outil	Version mini	Installation
`dig`	9.18+	`apt install dnsutils`
`delv`	9.18+	`apt install bind9-dnsutils`
`kdig`	3.x	`apt install knot-dnsutils`
`ldns-verify-zone`	1.8+	`apt install ldnsutils`
`openssl`	3.0+	`apt install openssl`
`dnssec-keygen`	9.18+	`apt install bind9utils`
`python3-dnspython`	2.4+	`pip install dnspython`

Outil	Version mini	Installation
`zonemaster-cli`	6.x	Voir zonemaster.net

Résolveurs de référence pour tests externes

IP	Opérateur	Validation DNSSEC
9.9.9.9	Quad9	Oui + threat intel
1.1.1.1	Cloudflare	Oui
8.8.8.8	Google	Oui
194.0.2.100	ANSSI/AFNIC	Oui

⚡ Mode Découverte Rapide — 15 Questions DNSSEC Clés

Évaluation rapide des risques DNSSEC critiques en 20 minutes

Q1. La zone est-elle signée DNSSEC ?

```
dig <domaine> DNSKEY +short
dig <domaine> RRSIG SOA +short
```

Critères de validation : - ✅ DNSKEY présent, RRSIG SOA présent et non expiré - ❌ Aucun DNSKEY = zone non signée, RRSIG absente = signing défaillant

Seuil critique : Absence de DNSKEY sur domaine OIV/OSE = non-conformité NIS 2 Art.21
Évaluation : ✅ ❌ ⚠️ N/A

Q2. L'algorithme utilisé est-il conforme RFC 8624 ?

```
dig <domaine> DNSKEY +short | awk '{print $3}' # champ algo
# Algo 13 = ECDSA P-256 (MUST), 15 = Ed25519 (RECOMMENDED), 8 = RSA/SHA-256 (OK si ≥2048 bits)
```

Critères de validation : - ✅ Algorithme 13 ou 15, pas d'algo 5 (RSA/SHA-1) ou 7 - ❌ Algo 5 ou 7 = déprécié, vulnérable, non conforme RFC 8624

Évaluation : ✅ ❌ ⚠️ N/A

Q3. Le DS est-il présent et cohérent chez le parent ?

```
dig <domaine> DS +short  # depuis resolver validant
dig @<parent-ns> <domaine> DS +short  # depuis NS parent
```

Critères de validation : - ✅ DS présent, digest type 2 (SHA-256) ou 4 (SHA-384), cohérent avec DNSKEY active - ❌ DS absent = chaîne de confiance brisée, zone non validable

Évaluation : ✅ ❌ ⚠️ N/A

Q4. Les signatures RRSIG sont-elles valides et non expirées ?

```
dig <domaine> RRSIG A +dnssec +short
# Vérifier Expiration > aujourd'hui et Inception < aujourd'hui
delv <domaine> A +rtrace
```

Critères de validation : - ✅ RRSIG non expiré, statut "fully validated" avec delv - ❌ RRSIG expiré = SERVFAIL pour tous les resolvers validants

Évaluation : ✅ ❌ ⚠️ N/A

Q5. NSEC3 est-il configuré avec iterations=0 (RFC 9276) ?

```
dig <domaine> NSEC3PARAM +short
# Format: algo iterations flags salt
# iterations DOIT être 0 depuis RFC 9276 (oct 2022)
```

Critères de validation : - ✅ Deuxième champ (iterations) = 0, sel ≥ 128 bits - ❌ Iterations > 0 = vulnérable aux attaques hashcat offline

Évaluation : ✅ ❌ ⚠️ N/A

Q6. Le résolveur interne effectue-t-il la validation DNSSEC ?

```
dig @<resolver-interne> dnssec-failed.org A
# Doit retourner SERVFAIL si validation active
dig @<resolver-interne> dnssec-failed.org A +dnssec | grep "ad"
```

Critères de validation : - ✅ SERVFAIL sur dnssec-failed.org, bit AD présent sur domaines signés valides - ❌ NOERROR sur dnssec-failed.org = validation non activée

Évaluation : ✅ ❌ ⚠️ N/A

Q7. Y a-t-il des enregistrements TLSA pour les services critiques ?

```
dig _25._tcp.<domaine-mx> TLSA +short  # SMTP DANE
dig _443._tcp.<domaine> TLSA +short    # HTTPS DANE
```

Critères de validation : - ✅ TLSA 3 1 1 présent pour SMTP sur port 25, cohérent avec certificat actuel - ❌ Absent = protection DANE inactive malgré DNSSEC

Évaluation : ✅ ❌ ⚠️ N/A

Q8. La procédure de rollover KSK est-elle documentée et testée ?

- Vérifier l'existence d'un DPS (DNSSEC Practice Statement)
- Vérifier la date du dernier rollover KSK en production
- Vérifier l'existence d'un environnement de staging DNSSEC

Critères de validation : - ✅ DPS à jour, rollover < 2 ans, staging opérationnel - ❌ Aucun DPS, jamais de rollover = risque catastrophique en cas de compromission KSK

Évaluation : ✅ ❌ ⚠️ N/A

Q9. Y a-t-il un monitoring RRSIG avec alertes d'expiration ?

```
# Vérifier l'expiration des RRSIG critiques
dig <domaine> RRSIG SOA +dnssec | grep "Expir"
# Alerte J-14 minimum recommandée
```

Critères de validation : - ✅ Monitoring automatique actif, alertes J-14, J-7, J-1 - ❌ Aucun monitoring = risque de zone bogué silencieux

Évaluation : ✅ ❌ ⚠️ N/A

Q10. Les clés KSK sont-elles stockées dans un HSM ?

- Vérifier l'inventaire des clés DNSSEC et leur lieu de stockage
- HSM minimum FIPS 140-2 Level 3 pour KSK en production

Critères de validation : - ✅ KSK dans HSM certifié, ZSK peut être online avec protection logicielle - ❌ KSK stockée en fichier sur serveur exposé = risque de vol

Évaluation : ✅ ❌ ⚠️ N/A

Q11. La zone est-elle testée via Zonemaster ou DNSViz ?

```
zonemaster-cli <domaine> --test dnssec
# Ou via https://dnsviz.net et https://zonemaster.net
```

Critères de validation : - ✅ Aucune erreur DNSSEC critique, avertissements documentés - ❌ Erreurs non résolues = risque de SERVFAIL chez resolver validant

Évaluation : ✅ ❌ ⚠️ N/A

Q12. Les DS Digest Type SHA-1 (type 1) sont-ils absents ?

```
dig <domaine> DS +short | awk '{if($3==1) print "SHA-1 DS FOUND — UNSAFE"}'
```

Critères de validation : - ✅ Aucun DS avec digest type 1 (SHA-1) = obsolète RFC 8624 - ❌ DS type 1 présent = non-conforme, à retirer immédiatement

Évaluation : ✅ ❌ ⚠️ N/A

Q13. Le registrar supporte-t-il CDS/CDNSKEY pour l'automatisation ?

```
dig <domaine> CDS +short    # Si publié
dig <domaine> CDNSKEY +short
# Consulter la liste IANA des registrars supportant CDS
```

Critères de validation : - ✅ Registrar supporte CDS/CDNSKEY, workflow automatisé documenté - ⚠️ Registrar ne supporte pas CDS = rollover KSK manuel, risque d'erreur

Évaluation : ✅ ❌ ⚠️ N/A

Q14. Les RPKI ROA couvrent-ils tous les préfixes des NS autoritatifs ?

```
whois -h whois.radb.net <IP-NS> | grep -i route
# Vérifier via https://rpki.cloudflare.com ou https://bgp.tools
```

Critères de validation : - ✅ ROA valide pour tous les préfixes IP hébergeant les NS - ❌ ROA absent ou invalide = risque de BGP hijacking vers NS DNSSEC

Évaluation : ✅ ❌ ⚠️ N/A

Q15. L'organisation dispose-t-elle d'un plan d'urgence KSK compromis ?

- Procédure écrite de rollover KSK d'urgence (< 4h)
- Contact registrar/registry en cas d'urgence documenté
- RTO défini pour résolution zone bogué (P0)

Critères de validation : - ✅ Procédure testée, contacts validés, RTO ≤ 4h - ❌ Aucune procédure = catastrophe opérationnelle si compromission KSK

Évaluation : ✅ ❌ ⚠️ N/A

S01 — Pré-requis et Inventaire DNSSEC

Contexte

Avant tout déploiement ou audit DNSSEC, l'organisation doit disposer d'un inventaire exhaustif de ses zones DNS, de leurs statuts de signature, et de la chaîne de délégation jusqu'à la racine. Cette étape est fondamentale : une zone oubliée ou un TTL mal calibré peut provoquer des outages silencieux lors des rollovers de clés. La norme RFC 6781 (DNSSEC Operational Practices) insiste sur la nécessité de documenter la politique de gestion des clés (DPS — DNSSEC Practice Statement) avant toute opération.

Les TTL jouent un rôle critique dans DNSSEC : un TTL DNSKEY trop bas augmente la charge des resolvers (re-fetch fréquent), tandis qu'un TTL DS trop élevé ralentit la propagation lors d'un rollover KSK. Le TTL DS chez le parent détermine directement la durée minimale d'une phase de double-signature lors d'un rollover.

Contrôles S01

S01-C001 — Inventaire complet des zones DNS de l'organisation

Criticité : 🟠 Élevé
Sources : RFC 6781 §2.2, ANSSI REC-DNSSEC-2021 §2

```
# Lister toutes les zones pour BIND
named-checkconf -p | grep 'zone "'
# Pour PowerDNS
pdns_control list-zones
# Vérifier statut de signature pour chaque zone
for zone in <liste-zones>; do
  echo -n "$zone : "
  dig @<ns-autoritatif> $zone DNSKEY +short | wc -l | xargs -I{} bash -c 'if [ {} -gt 0 ]; then echo "SIGNÉE";
else echo "NON SIGNÉE ❌"; fi'
done
```

Résultat attendu : Toutes les zones de production signées, registre documenté avec date de mise en service DNSSEC, algorithme utilisé, identifiants des clés actives.

Effet de bord : Une zone non inventoriée est un angle mort pour le monitoring d'expiration RRSIG. Si la zone existe dans le DNS public mais n'est pas dans l'inventaire, le renouvellement de signature sera oublié, conduisant à une zone bogué silencieuse.

Remédiation : Créer un tableur ou CMDB avec colonnes : nom de zone, NS autoritatif, registrar, algorithme DNSSEC, date activation, date dernier rollover KSK, date dernier rollover ZSK, emplacement des clés.

Évaluation : ✅ ❌ ⚠️ N/A

S01-C002 — Vérification du support DNSSEC chez le registrar et l'hébergeur DNS

Criticité : 🔴 Critique
Sources : RFC 4034 §5, RFC 7344

```
# Vérifier si le registrar accepte les enregistrements DS
# Test : soumettre un DS fictif via l'interface et observer
# Vérifier la liste IANA des registrars supportant CDS/CDNSKEY :
# https://www.iana.org/assignments/cds-cdnskey-registrars/
whois <domaine> | grep -i "registrar"
```

Résultat attendu : Le registrar supporte la publication manuelle de DS, idéalement CDS/CDNSKEY automatisé. L'hébergeur DNS (autoritatif) génère et signe les zones DNSSEC.

Effet de bord : Si le registrar ne supporte pas DNSSEC, aucune chaîne de confiance ne peut être établie vers la racine DNS, rendant DNSSEC inopérant même si la zone est techniquement signée. Certains hébergeurs mutualisés ne proposent pas DNSSEC.

Remédiation : Migrer vers un registrar DNSSEC-compatible (Gandi, OVH, AFNIC directement pour .fr). Pour les zones critiques, gérer les NS autoritatifs en propre (BIND, Knot DNS, PowerDNS).

Évaluation : ✅ ❌ ⚠️ N/A

S01-C003 — Audit et documentation des TTL DNSSEC

Criticité : 🟡 Moyen
Sources : RFC 6781 §3.3, RFC 4034 §2.4

```
# TTL du SOA (propagation des changements NS)
dig <domaine> SOA +short | awk '{print "TTL SOA:", $7}'
# TTL des DNSKEY (durée cache resolver pour les clés)
dig <domaine> DNSKEY +ttl +short | head -1 | awk '{print "TTL DNSKEY:", $1}'
# TTL du DS chez le parent (détermine durée minimum rollover KSK)
dig <domaine> DS +ttl | grep "^<domaine>" | awk '{print "TTL DS:", $2}'
# TTL des RRSIG (durée de validité de la signature)
dig <domaine> RRSIG SOA +short | awk '{print "Sig expire:", $5, "Sig inception:", $6}'
```

Résultat attendu : - TTL DNSKEY : 300-3600s (recommandé 3600s) - TTL DS chez parent : 86400s maximum (24h) - TTL SOA : ≥ 3600s - Durée RRSIG : 7-30 jours

Effet de bord : TTL DNSKEY trop bas (< 300s) = re-fetch constant par les resolvers → surcharge des NS autoritatifs lors d'un trafic intense. TTL DS trop élevé (> 86400s) = rollover KSK impossible en moins de 24h sans risque de zone bogué.

Remédiation : Ajuster les TTL dans le fichier de zone avant de signer. Pour modifier le TTL DS, contacter le registrar qui gère ses propres TTL sur les enregistrements DS délégués.

Évaluation : ✅ ❌ ⚠️ N/A

S01-C004 — Documentation de la chaîne de délégation complète

Criticité : 🟠 Élevé
Sources : RFC 4033 §3.1, RFC 6840 §5

```
# Tracer la chaîne complète depuis la racine
delv <domaine> A +rtrace +vtrace 2>&1 | head -50
# Vérifier chaque maillon : root → TLD → domaine
dig . DNSKEY +short | head -3     # Ancre de confiance racine
dig <tld> DS +short               # DS du TLD chez root
dig <domaine> DS +short           # DS du domaine chez TLD
dig <domaine> DNSKEY +short       # Clés de la zone
```

Résultat attendu : Chaîne continue root → TLD → domaine → sous-domaines, chaque maillon vérifié. Aucun "island of security" non connecté à la racine.

Effet de bord : Un DS manquant à n'importe quel niveau casse la validation pour la totalité des sous-domaines. Un "island of security" (zone signée sans DS parent) est valide techniquement mais ne bénéficie d'aucune protection pour les resolvers qui n'ont pas configuré d'ancre de confiance locale.

Remédiation : Soumettre le DS au registrar pour chaque zone signée. Documenter les îlots de sécurité intentionnels avec leur justification dans le DPS.

Évaluation : ✅ ❌ ⚠️ N/A

S01-C005 — Existence et conformité du DPS (DNSSEC Practice Statement)

Criticité : 🟡 Moyen
Sources : RFC 6781 §3.1, ANSSI REC-DNSSEC-2021 §4, AFNIC DPS

```
# Vérifier existence du document DPS dans la base documentaire
# Le DPS doit couvrir :
# - Politique de génération et stockage des clés
# - Procédures de rollover KSK et ZSK
# - Gestion des incidents
# - Contacts d'urgence
# - Durées de vie des clés et signatures
```

Résultat attendu : DPS rédigé, approuvé par la direction, versionné, et révisé au moins semestriellement. Pour les OIV/OSE, le DPS peut être requis par l'ANSSI.

Effet de bord : Absence de DPS = opérations DNSSEC ad hoc, risque d'incohérence entre opérateurs, impossibilité de démontrer la conformité lors d'un audit NIS 2.

Remédiation : S'inspirer du modèle AFNIC (disponible publiquement) ou du template ICANN pour rédiger le DPS. Le document doit être accessible à l'équipe opérationnelle 24/7.

Évaluation : ✅ ❌ ⚠️ N/A

S01-C006 — Vérification de la cohérence entre tous les NS autoritatifs

Criticité : 🟠 Élevé
Sources : RFC 4035 §4.1, RFC 6781 §3.3.4

```
# Obtenir la liste des NS autoritatifs
NS_LIST=$(dig <domaine> NS +short)
# Comparer les DNSKEY sur chaque NS
for ns in $NS_LIST; do
  echo "=== $ns ==="
  dig @$ns <domaine> DNSKEY +short | sort
done
# Les sorties doivent être identiques
```

Résultat attendu : DNSKEY identiques sur tous les NS autoritatifs, RRSIG valides sur chaque NS, numéros de série SOA synchronisés.

Effet de bord : Désynchronisation des DNSKEY entre NS = certains resolvers obtiennent une réponse valide, d'autres SERVFAIL selon le NS qu'ils interrogent. Incident de type "flapping" très difficile à diagnostiquer.

Remédiation : Vérifier la configuration de transfert de zone (AXFR/IXFR) entre NS primaire et secondaires. Forcer une synchronisation manuelle si nécessaire.

Évaluation : ✅ ❌ ⚠️ N/A

S01-C007 — Inventaire des sous-zones déléguées et statut DNSSEC

Criticité : 🟡 Moyen
Sources : RFC 4035 §2.3, RFC 6781 §2.3

```
# Identifier les sous-zones déléguées
dig <domaine> NS +short  # NS de la zone principale
# Pour chaque sous-zone connue :
dig <sous-zone>.<domaine> NS +short
dig <sous-zone>.<domaine> DS +short  # DS présent = délégation sécurisée
dig <sous-zone>.<domaine> DNSKEY +short  # Zone signée ?
```

Résultat attendu : Chaque sous-zone déléguée a son DS publié dans la zone parente, assurant la continuité de la chaîne de confiance. Les sous-zones non critiques peuvent être exemptées avec justification documentée.

Effet de bord : Une sous-zone déléguée sans DS est un point de rupture potentiel. Un attaquant peut substituer des réponses pour cette sous-zone sans être détecté par les resolvers validants (pas d'authenticated denial).

Remédiation : Publier les DS de toutes les sous-zones critiques dans la zone parente. Utiliser opt-out NSEC3 si de nombreuses sous-zones non signées existent (RFC 5155).

Évaluation : ✅ ❌ ⚠️ N/A

S01-C008 — Test de validation de bout en bout depuis une sonde externe

Criticité : 🟠 Élevé
Sources : RFC 4035, RIPE Atlas, ICANN DNSSEC Monitoring

```
# Test depuis resolver Quad9 (validation DNSSEC stricte)
dig @9.9.9.9 <domaine> A +dnssec | grep -E "SERVFAIL|NOERROR|ad"
# Test avec delv (validation locale complète)
delv <domaine> A @9.9.9.9 +vtrace 2>&1 | grep -E "fully validated|BOGUS|negative"
# Test zonemaster (nécessite installation ou accès web)
zonemaster-cli <domaine> --test dnssec --level INFO
```

Résultat attendu : `NOERROR` avec bit `ad` (Authenticated Data) sur les réponses DNSSEC valides. `SERVFAIL` uniquement sur les zones intentionnellement invalides. delv retourne `fully validated`.

Effet de bord : Un test uniquement depuis le réseau interne peut masquer des problèmes visibles depuis Internet (problèmes de propagation DS, différences entre NS). Toujours tester depuis au moins deux resolvers externes indépendants.

Remédiation : Mettre en place des sondes RIPE Atlas permanentes (5 minimum, géodistribuées) pour le monitoring continu de la validation DNSSEC des zones critiques.

Évaluation : ✅ ❌ ⚠️ N/A

S02 — Sélection des Algorithmes Cryptographiques (RFC 8624)

Contexte

Le choix de l'algorithme de signature DNSSEC est la décision cryptographique la plus importante du déploiement. La RFC 8624 (Algorithm Implementation Requirements and Usage Guidance for DNSSEC, 2019) définit les statuts MUST/SHOULD/MAY/NOT RECOMMENDED/MUST NOT pour chaque algorithme. En 2026, les algorithmes 13 (ECDSA P-256/SHA-256) et 15 (Ed25519) sont les référentiels de l'état de l'art. Les algorithmes RSA/SHA-1 (5, 7) sont formellement proscrits.

Ed25519 (algorithme 15) offre des signatures plus petites (64 octets vs 256 octets pour RSA-2048), une génération de clés plus rapide, et une résistance aux timing attacks par construction. ECDSA P-256 (algorithme 13) est le choix de compatibilité maximale pour les environnements avec des resolvers anciens. L'algorithme 14 (ECDSA P-384) est réservé aux usages nécessitant une sécurité renforcée (durée de vie > 10 ans, conformité gouvernementale).

Contrôles S02

S02-C001 — Vérification que l'algorithme actif est conforme RFC 8624

Criticité : 🔴 Critique
Sources : RFC 8624 §3, ANSSI REC-DNSSEC-2021 §3.1, NIST SP 800-81r2

```
# Extraire les algorithmes de toutes les DNSKEY de la zone
dig <domaine> DNSKEY +short | awk '{print "Flags:", $1, "Proto:", $2, "Algo:", $3}'
# Décoder l'identifiant d'algorithme :
# 5 = RSASHA1 (MUST NOT signer, MUST NOT valider)
# 7 = RSASHA1-NSEC3-SHA1 (NOT RECOMMENDED)
# 8 = RSA/SHA-256 (MAY — si clé ≥ 2048 bits)
# 10 = RSA/SHA-512 (NOT RECOMMENDED)
# 13 = ECDSAP256SHA256 (MUST)
# 14 = ECDSAP384SHA384 (MAY)
# 15 = ED25519 (RECOMMENDED)
# 16 = ED448 (MAY)
```

Résultat attendu : Algorithme 13 ou 15 pour toutes les clés actives. Aucun algorithme 5 ou 7 en utilisation active.

Effet de bord : La migration d'algorithme (algorithm rollover) est une opération délicate décrite dans la RFC 6781 §4.1.4. Elle nécessite une phase de double-signature avec l'ancien et le nouvel algorithme. Un rollover d'algorithme bâclé peut rendre la zone bogué pour les resolvers qui ne supportent pas encore le nouvel algorithme.

Remédiation : Effectuer un algorithm rollover planifié selon RFC 6781 §4.1.4. Phase 1 : créer nouvelles clés avec algo cible + signer avec les deux algos. Phase 2 : soumettre DS du nouvel algo. Phase 3 : retirer ancien DS. Phase 4 : retirer anciennes clés.

Évaluation : ✅ ❌ ⚠️ N/A

S02-C002 — Interdiction des algorithmes dépréciés (5, 7) en signature active

Criticité : 🔴 Critique
Sources : RFC 8624 §3.1 (MUST NOT sign, MUST NOT validate)

```
# Détecter si une RRSIG utilise un algorithme déprécié
dig <domaine> RRSIG SOA +short | awk '{print "Algo RRSIG:", $2}'
# Chercher algo 5 ou 7 dans les RRSIG de toute la zone
for rr in SOA A MX NS TXT; do
  dig <domaine> RRSIG $rr +short | awk -v rr=$rr '{if($2==5||$2==7) print "DÉPRÉCIÉ algo",$2,"sur"rr}'
done
```

Résultat attendu : Aucune RRSIG signée avec algorithme 5 (RSA/SHA-1) ou 7 (RSASHA1-NSEC3). La présence de l'ancienne DNSKEY pendant un algorithm rollover est acceptable, mais elle ne doit plus signer.

Effet de bord : Certains registries asiatiques et certains TLD anciens n'acceptent que les DS correspondant à l'algorithme 5. Dans ce cas exceptionnel, maintenir un DS SHA-1 uniquement pour la délégation parente, et utiliser algo 13/15 pour la signature effective de la zone. Documenter cette exception dans le DPS.

Remédiation : Initier immédiatement un algorithm rollover. En cas de blocage registrar (non-support algo 13/15), escalader auprès du registrar ou envisager une migration.

Évaluation : ✅ ❌ ⚠️ N/A

S02-C003 — Taille de clé RSA conforme si algorithme RSA (algo 8) utilisé

Criticité : 🟠 Élevé
Sources : NIST SP 800-81r2 §4.1, RFC 8624 §3

```
# Vérifier la taille des clés RSA si algo 8 est utilisé
dig <domaine> DNSKEY +short | while read flags proto algo key; do
  if [ "$algo" = "8" ]; then
    # Décoder la clé base64 et vérifier la taille
    echo "$key" | base64 -d | wc -c | awk '{print "Taille clé RSA bytes:", $1, "(min 256 bytes = 2048 bits)"}'
  fi
done
```

Résultat attendu : Pour RSA (algo 8) : clé ≥ 2048 bits (256 bytes). Pour KSK RSA : recommandé 4096 bits. Pour ZSK RSA : 2048 bits minimum.

Effet de bord : RSA-1024 bits (128 bytes) est cassable avec du matériel moderne. Une ZSK RSA-1024 peut être factorisée en quelques jours/semaines selon les ressources. RSA-4096 bits en ZSK génère des réponses DNS de grande taille (> 1232 bytes) → fragmentation UDP systématique → TCP fallback obligatoire.

Remédiation : Si RSA-1024 détecté : rollover ZSK d'urgence vers RSA-2048 ou migration vers ECDSA P-256 (algo 13).

Évaluation : ✅ ❌ ⚠️ N/A

S02-C004 — Évaluation de la migration vers Ed25519 (algorithme 15)

Criticité : 🟢 Faible
Sources : RFC 8624 §3.2 (RECOMMENDED), RFC 8080

```
# Vérifier le support Ed25519 sur le serveur DNS autoritatif
named -v | grep "BIND"
# BIND ≥ 9.12 supporte Ed25519 (algo 15)
# Knot DNS ≥ 2.4, PowerDNS ≥ 4.1
dig <domaine> DNSKEY +short | grep "^257\|^256" | awk '{print "Algo:", $3}'
# Algo 15 = Ed25519 présent ?
```

Résultat attendu : Migration vers Ed25519 planifiée ou déjà effectuée. Avantages : signatures 64 bytes (vs 256 pour RSA-2048), performances supérieures, résistance aux timing attacks.

Effet de bord : Certains vieux resolvers (BIND < 9.11, Unbound < 1.5) ne supportent pas Ed25519 et retourneront SERVFAIL pour les zones Ed25519-only. En pratique en 2026, la couverture est > 99% mais vérifier les résolveurs spécifiques de l'environnement cible (appareils IoT, anciens routeurs).

Remédiation : Tester la compatibilité Ed25519 avec l'outil `check.sidnlabs.nl`. Déployer en double-signing (algo 13 + 15) si besoin de compatibilité maximale. Migration complète possible quand la compatibilité est validée.

Évaluation : ✅ ❌ ⚠️ N/A

S02-C005 — Vérification du support algorithmique par l'ensemble des NS secondaires

Criticité : 🟠 Élevé
Sources : RFC 6781 §3.4.2

```
# Tester que chaque NS secondaire accepte et transmet correctement les DNSKEY
NS_LIST=$(dig <domaine> NS +short)
for ns in $NS_LIST; do
  echo "=== Test DNSSEC depuis $ns ==="
  dig @$ns <domaine> DNSKEY +dnssec +short | head -3
  dig @$ns <domaine> RRSIG DNSKEY +short | awk '{print "Algo RRSIG:", $2}'
done
```

Résultat attendu : Tous les NS secondaires transmettent les DNSKEY et RRSIG correctement. Aucun NS ne strip les enregistrements DNSSEC.

Effet de bord : Un NS secondaire qui strip les enregistrements DNSSEC (bug logiciel, configuration de filtrage, anycast via provider tiers) rend la zone bogué pour les resolvers qui interrogent ce NS. Difficile à diagnostiquer car l'incident est intermittent (round-robin DNS).

Remédiation : Vérifier la configuration du serveur DNS secondaire. Mettre à jour si version ancienne. Contacter le provider anycast si le NS est géré par un tiers.

Évaluation : ✅ ❌ ⚠️ N/A

S02-C006 — Vérification de l'absence d'algorithme 10 (RSA/SHA-512)

Criticité : 🟡 Moyen
Sources : RFC 8624 §3 (NOT RECOMMENDED)

```
dig <domaine> DNSKEY +short | awk '{if($3==10) print "Algo 10 (RSA/SHA-512) détecté — NOT RECOMMENDED"}'
dig <domaine> RRSIG SOA +short | awk '{if($2==10) print "RRSIG avec algo 10 détectée"}'
```

Résultat attendu : Aucun algorithme 10 utilisé. RSA/SHA-512 génère des signatures plus grandes que RSA/SHA-256 sans apport de sécurité significatif en DNSSEC, car la sécurité est limitée par la taille de la clé RSA, pas du hash.

Effet de bord : Les grandes tailles de signature RSA/SHA-512 augmentent la taille des réponses DNS, augmentant le risque de fragmentation UDP et de paquets tronqués pour les clients avec MTU réduit.

Remédiation : Migrer vers algo 13 (ECDSA P-256) ou 15 (Ed25519) via un algorithm rollover planifié.

Évaluation : ✅ ❌ ⚠️ N/A

S02-C007 — Documentation du plan d'algorithm rollover

Criticité : 🟡 Moyen
Sources : RFC 6781 §4.1.4, RFC 7583 §5

```
# Vérifier que la documentation de rollover d'algorithme existe
# et couvre les étapes RFC 6781 §4.1.4 :
# 1. Publication nouvelle DNSKEY (nouvel algo)
# 2. Début double-signature (ancien + nouvel algo)
# 3. Attente TTL_DNSKEY + propagation
# 4. Soumission nouveau DS au registrar
# 5. Attente TTL_DS propagation
# 6. Retrait ancien DS
# 7. Fin double-signature
# 8. Retrait ancienne DNSKEY
```

Résultat attendu : Procédure écrite, testée en staging, avec durées minimales pour chaque étape basées sur les TTL réels de la zone.

Effet de bord : Un algorithm rollover précipité sans respecter les délais de propagation TTL entraîne systématiquement une zone bogué. L'incident ICANN root KSK rollover 2018 illustre les risques même pour des organisations très expérimentées.

Remédiation : Rédiger un runbook algorithm rollover spécifique à chaque zone, incluant les TTL réels et les durées d'attente calculées.

Évaluation : ✅ ❌ ⚠️ N/A

S02-C008 — Évaluation algorithme ECDSA P-384 (algo 14) pour usage gouvernemental

Criticité : 🟢 Faible
Sources : RFC 8624 §3 (MAY), ANSSI REC-DNSSEC-2021 §3.2

```
# Vérifier si algo 14 est requis selon le référentiel de l'organisation
# Applicable si : durée de vie des clés > 10 ans, conformité RGS (Référentiel Général de Sécurité)
dig <domaine> DNSKEY +short | awk '{if($3==14) print "Algo 14 (ECDSA P-384) actif"}'
```

Résultat attendu : Algo 14 utilisé uniquement si l'organisation relève d'une réglementation spécifique (RGS, secteur défense). Pour les usages standard, algo 13 ou 15 suffit.

Effet de bord : P-384 génère des signatures légèrement plus grandes que P-256, et les performances de signature sont inférieures. Pas de bénéfice pratique pour la durée de vie des ZSK (quelques mois).

Évaluation : ✅ ❌ ⚠️ N/A

S02-C009 — Test de compatibilité algorithmique avec les resolvers cibles de l'organisation

Criticité : 🟡 Moyen
Sources : RFC 6840 §3.7, IANA DNSSEC Algorithm Numbers

```
# Vérifier quels algorithmes sont supportés par les resolvers internes
# Pour Unbound
unbound-checkconf | grep "val-sig-skipped-expirations"
unbound --version
# Pour BIND
named -v
# Algorithmes supportés depuis :
# BIND 9.7+ : algo 13 (ECDSA P-256)
# BIND 9.12+ : algo 15 (Ed25519)
# Unbound 1.5+ : algo 15
```

Résultat attendu : Tous les resolvers internes (Unbound, BIND, Windows DNS Server) supportent l'algorithme utilisé pour signer les zones.

Effet de bord : Un resolver qui ne supporte pas l'algorithme de signature d'une zone traite la zone comme "algorithm not supported" → considère la zone non signée et ne valide pas → perd le bénéfice DNSSEC pour les utilisateurs de ce resolver.

Remédiation : Mettre à jour les resolvers internes. Windows Server 2008 ne supporte pas algo 13 — minimum Windows Server 2012 R2 pour ECDSA.

Évaluation : ✅ ❌ ⚠️ N/A

S02-C010 — Vérification de la longueur des réponses DNS avec l'algorithme actuel

Criticité : 🟡 Moyen
Sources : RFC 6840 §5.3, RFC 4035 §2.3

```
# Mesurer la taille des réponses DNSSEC
dig <domaine> A +dnssec +bufsize=4096 | grep ";; MSG SIZE"
dig <domaine> DNSKEY +dnssec | grep ";; MSG SIZE"
# Seuil critique : > 1232 bytes = fragmentation UDP possible sur réseaux avec MTU 1280
# Seuil absolu : > 4096 bytes = troncature même avec EDNS0
```

Résultat attendu : Réponses DNSSEC < 1232 bytes idéalement, < 4096 bytes obligatoirement. Avec Ed25519, les réponses sont nettement plus compactes qu'avec RSA.

Effet de bord : Réponses > 1232 bytes sur réseaux IPv6 avec MTU 1280 → fragmentation UDP → paquets perdus → TCP fallback → latence 3-10× supérieure. Certains middlewares réseau bloquent les fragments UDP, rendant la résolution DNS impossible pour les clients DNSSEC.

Remédiation : Migrer vers Ed25519 (algo 15) pour minimiser la taille des signatures. Activer EDNS0 avec bufsize = 4096 sur les NS autoritatifs. S'assurer que TCP port 53 est ouvert sur tous les pare-feux intermédiaires.

Évaluation : ✅ ❌ ⚠️ N/A

S03 — Gestion des Clés KSK / ZSK / CSK

Contexte

La gestion des clés est le cœur opérationnel de DNSSEC. On distingue deux types de clés dans le modèle classique : la KSK (Key Signing Key), qui signe uniquement les DNSKEY et dont l'empreinte est publiée en tant que DS chez le parent ; et la ZSK (Zone Signing Key), qui signe tous les autres enregistrements de la zone. Cette séparation permet de conserver la KSK offline (dans un HSM) pendant que la ZSK est accessible en ligne pour la re-signature automatique fréquente.

Le modèle CSK (Combined Signing Key) utilise une seule clé pour tout signer. Il simplifie les opérations mais expose davantage la clé (connexion plus fréquente nécessaire). Ce modèle est acceptable pour les petites zones avec rollover automatisé via CDS/CDNSKEY.

La RFC 6781 et le NIST SP 800-81r2 définissent les durées de vie maximales recommandées : KSK 1-2 ans, ZSK 1-3 mois. Un stockage des clés privées KSK dans un HSM certifié FIPS 140-2 Level 3 est requis pour les zones critiques (OIV, secteur financier, gouvernement).

Contrôles S03

S03-C001 — Séparation correcte KSK / ZSK dans la zone

Criticité : 🟡 Moyen
Sources : RFC 4034 §2.1.1, RFC 6781 §3.1.2

```
# Les DNSKEY avec flags 257 = KSK (SEP=1), flags 256 = ZSK
dig <domaine> DNSKEY +short | awk '{
  if($1==257) print "KSK (flags 257):", $3, $4
  if($1==256) print "ZSK (flags 256):", $3, $4
}'
# Vérifier que la KSK ne signe que les DNSKEY :
dig <domaine> RRSIG DNSKEY +short  # Doit être signée par la KSK
dig <domaine> RRSIG A +short       # Doit être signée par la ZSK
```

Résultat attendu : Au moins une DNSKEY avec flags 257 (KSK) et une avec flags 256 (ZSK). Le DS chez le parent référence uniquement le keytag de la KSK.

Effet de bord : Certaines implémentations (PowerDNS < 4.0) créent parfois des KSK avec flags incorrects. Une ZSK avec le flag SEP positionné à tort peut tenter d'enregistrer son DS chez le parent, créant de la confusion lors des rollovers.

Remédiation : Si le modèle CSK est utilisé volontairement, le documenter dans le DPS avec justification. Sinon, recréer les clés avec les bons flags.

Évaluation : ✅ ❌ ⚠️ N/A

S03-C002 — Stockage de la KSK dans un HSM certifié (clés critiques)

Criticité : 🔴 Critique (zones critiques OIV/OSE/secteur financier)
Sources : NIST SP 800-81r2 §5, ANSSI REC-DNSSEC-2021 §5, RFC 6781 §3.1.3

```
# Vérifier la configuration PKCS#11 pour HSM avec BIND
grep -r "pkcs11\|hsm\|token" /etc/named.conf /etc/bind/
# Pour Knot DNS
knotc conf-read server.nsid
grep -r "hsm\|pkcs11" /etc/knot/
# Tester que la clé privée KSK N'EST PAS sur le système de fichiers :
ls /var/cache/bind/*.private  # Ne doit pas contenir la KSK si HSM utilisé
```

Résultat attendu : La clé privée KSK est stockée exclusivement dans le HSM (YubiHSM 2, Thales Luna, AWS CloudHSM, etc.) et n'existe pas en clair sur le système de fichiers. La ZSK peut être online (protection logicielle acceptable).

Effet de bord : La latence de signature via HSM est supérieure à la signature logicielle (5-50ms vs < 1ms). Pour les zones à haute fréquence de re-signature ou les KSK utilisées comme CSK, cela peut impacter les performances. Le HSM peut devenir un SPOF si non redondé.

Remédiation : Déployer le HSM en configuration HA (cluster). Tester régulièrement l'accès au HSM depuis le serveur DNS signing. Documenter la procédure de récupération si le HSM est inaccessible.

Évaluation : ✅ ❌ ⚠️ N/A

S03-C003 — Durée de vie de la KSK conforme (≤ 2 ans)

Criticité : 🟠 Élevé
Sources : RFC 6781 §3.1.1, NIST SP 800-81r2 §4.3

```
# Identifier la date de création de la KSK active
dig <domaine> DNSKEY +short | awk '{if($1==257) print $3, $4}'
# Cross-référencer avec le keytag dans le fichier de clé BIND :
grep -l "257" /var/cache/bind/K<domaine>.*.key 2>/dev/null | while read f; do
  grep "Created\|Publish\|Activate" ${f%.key}.private
done
# Calculer l'âge de la clé :
# Date création KSK vs date du jour
```

Résultat attendu : La KSK active a été créée/activée il y a moins de 2 ans. Un rollover KSK est planifié avant l'échéance de 2 ans.

Effet de bord : Une KSK utilisée trop longtemps augmente le risque cumulatif de compromission (plus de matériel signé avec la même clé, plus de temps disponible pour une attaque). La RFC 6781 recommande 1-2 ans comme maximum absolu pour les zones non HSM, et jusqu'à 3-5 ans avec HSM FIPS 140-3.

Remédiation : Planifier le rollover KSK dans le calendrier de l'équipe DNS. Ajouter une alerte automatique J-90 avant l'expiration de la durée de vie maximale de la KSK.

Évaluation : ✅ ❌ ⚠️ N/A

S03-C004 — Durée de vie de la ZSK conforme (1-3 mois)

Criticité : 🟡 Moyen
Sources : RFC 6781 §3.1.1, RFC 7583

```
# Vérifier la date d'activation de la ZSK courante
grep -l "256" /var/cache/bind/K<domaine>.*.key 2>/dev/null | while read f; do
  echo "=== $f ==="
  grep "Activate\|Inactive\|Delete" ${f%.key}.private
done
# Pour PowerDNS : vérifier la table cryptokeys
# mysql: SELECT * FROM cryptokeys WHERE flags=256;
```

Résultat attendu : La ZSK active a une durée de vie configurée entre 1 et 3 mois. Le rollover automatique est configuré et documenté.

Effet de bord : ZSK trop courte (< 1 mois) = rollovers fréquents qui augmentent le risque opérationnel et la charge des serveurs. ZSK trop longue (> 6 mois) = fenêtre d'exposition plus grande en cas de compromission ; les RRSIG existantes restent valides jusqu'à leur expiration propre même si la ZSK est retirée.

Remédiation : Configurer le rollover automatique dans le serveur DNS (BIND `auto-dnssec maintain`, PowerDNS `default-ksk-algorithm-dnssec-key-size`, Knot DNS `kasp-db`). Tester le rollover automatique en staging.

Évaluation : ✅ ❌ ⚠️ N/A

S03-C005 — Politique de sauvegarde des clés privées DNSSEC

Criticité : 🔴 Critique
Sources : RFC 6781 §3.1.3, NIST SP 800-57 Part 1

```
# Vérifier l'existence de sauvegardes chiffrées des clés
# Les fichiers .private BIND contiennent la clé privée en clair
ls -la /var/cache/bind/K*.private
# Vérification des sauvegardes chiffrées :
# Les sauvegardes doivent être chiffrées avec AES-256 minimum
# et stockées selon règle 3-2-1 (3 copies, 2 médias, 1 hors-site)
# Vérifier le dernier test de restauration
```

Résultat attendu : Sauvegardes des clés privées ZSK et KSK chiffrées (GPG/AES-256), testées en restauration au moins semestriellement. Stockage hors-site (coffre physique ou cloud chiffré). Journal des restaurations.

Effet de bord : Perte des clés privées sans sauvegarde = impossibilité de re-signer la zone après incident → zone bogué permanente → nécessité de recréer toutes les clés et recommencer le rollover KSK depuis zéro. Pour les zones avec DS chez le parent, cela prend 24-48h minimum (propagation DS).

Remédiation : Mettre en place une procédure de sauvegarde chiffrée automatisée. Chiffrer avec GPG et une clé de récupération stockée dans un coffre physique. Tester la restauration trimestriellement.

Évaluation : ✅ ❌ ⚠️ N/A

S03-C006 — Contrôle d'accès aux clés privées DNSSEC (MFA et privilèges minimaux)

Criticité : 🔴 Critique
Sources : ISO 27001 A.9.4.2, NIST SP 800-81r2 §7.4

```
# Vérifier les permissions sur les fichiers de clés BIND
ls -la /var/cache/bind/K*.private
# Attendu : -rw------- root root (ou bind:bind)
# Vérifier que le processus named ne tourne pas en root
ps aux | grep named | grep -v grep
# Journalisation des accès aux clés
auditctl -l | grep bind
```

Résultat attendu : Fichiers de clés accessibles uniquement par l'utilisateur du processus DNS (bind/named). Accès aux HSM protégé par MFA (token HSM + PIN). Journalisation de tous les accès aux clés dans le HSM audit log.

Effet de bord : Un processus DNS compromis avec accès aux clés privées permet à l'attaquant de signer des enregistrements DNS frauduleux valides, contournant totalement la protection DNSSEC pour les clients.

Remédiation : Appliquer le principe du moindre privilège. Utiliser des comptes de service dédiés. Activer l'audit des fichiers de clés via auditd. Configurer des alertes sur tout accès non planifié aux clés.

Évaluation : ✅ ❌ ⚠️ N/A

S03-C007 — Évaluation du modèle CSK (Combined Signing Key) si utilisé

Criticité : 🟡 Moyen
Sources : RFC 6781 §3.1.2, draft-ietf-dnsop-dnssec-automation

```
# Détecter l'utilisation d'un CSK (DNSKEY flags 257 qui signe tous les RR)
dig <domaine> DNSKEY +short | awk '{if($1==257) print "KSK-type key:", $4}'
dig <domaine> RRSIG A +short | awk '{print "Keytag RRSIG A:", $7}'
dig <domaine> RRSIG DNSKEY +short | awk '{print "Keytag RRSIG DNSKEY:", $7}'
# Si les deux keytags sont identiques = modèle CSK
```

Résultat attendu : Si CSK utilisé : raison documentée dans le DPS (petite zone, hébergement DNS managé, automation complète via CDS/CDNSKEY). Le CSK doit avoir une durée de vie courte (≤ 3 mois) pour compenser l'exposition accrue.

Effet de bord : Avec CSK, la clé doit être accessible en ligne pour signer régulièrement tous les RR → impossible de la stocker offline comme une KSK classique. En cas de compromission du serveur de signing, la clé permettant de modifier le DS est exposée.

Remédiation : Privilégier le modèle KSK/ZSK pour les zones critiques. Réserver le CSK aux zones de faible criticité avec rollover automatisé complet (CDS/CDNSKEY + registrar supportant l'automatisation).

Évaluation : ✅ ❌ ⚠️ N/A

S03-C008 — Vérification de la procédure de destruction sécurisée des clés obsolètes

Criticité : 🟡 Moyen
Sources : RFC 6781 §3.5, NIST SP 800-57 Part 1 §8.3

```
# Identifier les clés DNS en statut "Delete" ou révoquées
grep -l "Delete:" /var/cache/bind/K*.private 2>/dev/null | while read f; do
  echo "Clé à supprimer : $f"
  grep "Delete:" $f
done
# Pour HSM : vérifier que les clés révoquées sont effectivement détruites
# pkcs11-tool --list-objects --type privkey (ne doit plus lister les clés révoquées)
```

Résultat attendu : Les clés DNS expirées/révoquées sont détruites dans les délais (après expiration de toutes les RRSIG signées par cette clé + TTL max). Fichiers `.key` et `.private` supprimés. Clé HSM détruite (shred).

Effet de bord : Une clé ZSK expirée conservée sur le système est un risque si le serveur est compromis (l'attaquant peut créer des RRSIG rétroactives valides tant que la clé est considérée active par certains caches). L'accumulation de vieilles clés peut aussi causer des bugs dans certains implémentations.

Remédiation : Configurer la suppression automatique des clés après la période de grâce (BIND `dnssec-loadkeys-interval`, PowerDNS policy engine). Documenter le processus de destruction dans le DPS.

Évaluation : ✅ ❌ ⚠️ N/A

S03-C009 — Environnement de staging DNSSEC opérationnel et synchronisé

Criticité : 🟡 Moyen
Sources : RFC 6781 §7, ICANN DNSSEC Best Practices

```
# Vérifier l'existence d'un environnement DNS de test
# Le staging doit répliquer exactement la configuration de production
# Tester les opérations suivantes en staging avant production :
# - Rollover KSK
# - Rollover ZSK
# - Algorithm rollover
# - Changement de paramètres NSEC3
# - Migration vers nouveau hébergeur DNS
```

Résultat attendu : Environnement de staging avec : zone de test signée, resolver de test configuré pour valider, registrar test (ou simulation avec ancre de confiance locale pour l'environnement test). Toute procédure DNSSEC testée en staging avant déploiement en production.

Effet de bord : L'absence de staging conduit à tester directement en production. Les conséquences d'une erreur DNSSEC en production sont immédiates et visibles par 100% des utilisateurs utilisant des resolvers validants.

Remédiation : Créer une zone de test avec un sous-domaine dédié (ex: test.internal) et configurer un resolver validant en interne pour ce domaine test. Documenter la procédure de test.

Évaluation : ✅ ❌ ⚠️ N/A

S03-C010 — Vérification de la séparation des rôles pour les opérations sur les clés

Criticité : 🟡 Moyen
Sources : ISO 27001 A.9.4.1, NIST SP 800-81r2 §7.1

Résultat attendu : Trois rôles distincts minimum : (1) Administrateur DNS (configuration serveur) ; (2) Opérateur DNSSEC (opérations sur les clés, rollovers) ; (3) Auditeur (vérification indépendante). Les deux premiers rôles ne peuvent pas être détenus par la même personne seule pour les opérations critiques (KSK rollover, rollover d'urgence).

Effet de bord : Concentration des rôles = risque d'erreur sans second regard, risque de fraude interne, risque de SPOF humain (une seule personne connaît les procédures).

Remédiation : Documenter la matrice RACI pour les opérations DNSSEC. Implémenter la règle des deux personnes (dual control) pour les opérations KSK. Former au moins deux personnes à chaque rôle.

Évaluation : ✅ ❌ ⚠️ N/A

S03-C011 — Vérification de l'inventaire des keytags actifs et leur correspondance DS

Criticité : 🟠 Élevé
Sources : RFC 4034 §5.1, RFC 6840 §5.1

```
# Extraire les keytags des DNSKEY actives
dig <domaine> DNSKEY +short | while read flags proto algo key; do
  echo "Flags=$flags Algo=$algo KeyTag=$(echo "$flags $proto $algo $key" | python3 -c "
import sys, base64, struct
line = sys.stdin.read().split()
data = struct.pack('!HBB', int(line[0]), int(line[1]), int(line[2])) + base64.b64decode(line[3])
ac = 0
for i, b in enumerate(data):
    if i & 1: ac += b
    else: ac += b << 8
ac += (ac >> 16) & 0xffff
print(ac & 0xffff)
  ")"
done
# Cross-check avec les DS chez le parent :
dig <domaine> DS +short | awk '{print "DS keytag:", $1}'
```

Résultat attendu : Tous les keytags des KSK actives ont un DS correspondant chez le parent. Aucun DS orphelin (keytag sans DNSKEY correspondante). Aucune KSK active sans DS.

Effet de bord : Un DS orphelin (DNSKEY supprimée mais DS toujours présent) rend la zone bogué car les resolvers voient un DS qui ne correspond à aucune clé présente → validation échoue → SERVFAIL.

Remédiation : Lors de tout rollover KSK, vérifier systématiquement la cohérence entre keytags des DNSKEY et keytags des DS, avant et après chaque étape.

Évaluation : ✅ ❌ ⚠️ N/A

S03-C012 — Test de régénération des clés depuis les sauvegardes (exercice annuel)

Criticité : 🟡 Moyen
Sources : NIST SP 800-57 Part 1 §6.3.6, RFC 6781 §7.2

Résultat attendu : Au moins un test annuel de restauration des clés depuis les sauvegardes chiffrées, dans l'environnement de staging. Le test vérifie que la zone peut être re-signée correctement après restauration des clés sauvegardées.

Effet de bord : Une sauvegarde non testée est une promesse non tenue. Les cas fréquents de sauvegarde inutilisable incluent : format de clé incompatible avec la nouvelle version du logiciel DNS, passphrase de déchiffrement inconnue du responsable de garde, corruption silencieuse du fichier de sauvegarde.

Remédiation : Documenter le résultat du dernier test de restauration (date, résultat, responsable). Planifier le prochain test dans le DPS.

Évaluation : ✅ ❌ ⚠️ N/A

S04 — Signatures RRSIG : Validité et Gestion

Contexte

Les enregistrements RRSIG (Resource Record Signature) sont les preuves cryptographiques qui authentifient chaque réponse DNS signée. Chaque RRSIG contient : l'algorithme utilisé, le keytag de la clé signataire, la date d'inception (début de validité), la date d'expiration, et la signature elle-même. Un RRSIG expiré est la cause la plus fréquente de zone bogué en production.

La RFC 6781 §4.4 recommande une durée de validité des RRSIG entre 7 et 30 jours pour les zones dynamiques. La marge de re-signature (re-sign margin) doit être au moins 25% de la durée totale de validité : si RRSIG expire dans 14 jours, la re-signature doit être déclenchée à J-3.5 minimum. Le jitter (variation aléatoire des dates d'expiration) évite les pics de charge sur le serveur de signing lorsque toutes les RRSIG d'une zone expirent simultanément.

Contrôles S04

S04-C001 — Vérification que toutes les RRSIG sont valides et non expirées

Criticité : 🔴 Critique
Sources : RFC 4035 §2.2, RFC 6840 §5.1

```
# Vérifier les RRSIG pour les types critiques
for rr_type in SOA A AAAA MX NS TXT DNSKEY; do
  result=$(dig <domaine> RRSIG $rr_type +short 2>/dev/null)
  if [ -z "$result" ]; then
    echo "❌ RRSIG $rr_type : ABSENT"
    continue
  fi
  # Extraire date expiration
  exp_date=$(echo "$result" | awk '{print $5}' | head -1)
  echo "RRSIG $rr_type expire : $exp_date"
  # Comparer avec today (format YYYYMMDDHHMMSS)
  today=$(date -u +%Y%m%d%H%M%S)
  if [ "$exp_date" -lt "$today" ]; then
    echo "❌ RRSIG $rr_type EXPIRÉE ! ($exp_date < $today)"
  else
    echo "✅ RRSIG $rr_type valide"
  fi
done
```

Résultat attendu : Toutes les RRSIG présentes pour les types RR existants dans la zone, dates d'expiration dans le futur (minimum J+7 pour ne pas être en zone d'alerte).

Effet de bord : Une RRSIG expirée sur un seul type RR (ex: MX) bloque uniquement la résolution de ce type pour les resolvers validants → perte de la messagerie pour les expéditeurs avec resolver validant. Une RRSIG SOA ou NS expirée bloque toute la zone.

Remédiation : Redémarrage du daemon de signing BIND/PowerDNS/Knot avec re-signature forcée. En urgence : `rndc sign <zone>` (BIND) ou `pdns_control rectify-zone <zone>` (PowerDNS).

Évaluation : ✅ ❌ ⚠️ N/A

S04-C002 — Durée de validité RRSIG conforme (7-30 jours)

Criticité : 🟡 Moyen
Sources : RFC 6781 §4.4.2, ANSSI REC-DNSSEC-2021 §4

```
# Calculer la durée de vie des RRSIG
dig <domaine> RRSIG SOA +short | awk '{
  exp = $5; inc = $6
  # Format : YYYYMMDDHHMMSS
  # Durée en secondes approximative
  printf "Inception: %s, Expiration: %s\n", inc, exp
}'
# Durée effective = expiration - inception en jours
# Pour BIND, configurer sig-validity-interval en jours
grep "sig-validity-interval" /etc/named.conf /etc/bind/named.conf* 2>/dev/null
```

Résultat attendu : Durée de validité RRSIG entre 7 et 30 jours. Recommandé : 14 jours pour les zones dynamiques, 30 jours pour les zones statiques. La re-signature automatique intervient à 75% de la durée écoulée (ou 25% restant).

Effet de bord : RRSIG < 7 jours = charge CPU signing élevée, risque d'expiration si le serveur de signing est indisponible 24h. RRSIG > 30 jours = fenêtre d'exposition si la ZSK est compromise (les RRSIG restent valides jusqu'à leur expiration même si la ZSK est révoquée).

Remédiation : BIND : `sig-validity-interval 14 7;` (14 jours de validité, re-signe à J-7). PowerDNS : `default-ttl=86400` + `sign-validity-days=14`.

Évaluation : ✅ ❌ ⚠️ N/A

S04-C003 — Configuration du jitter sur les dates d'expiration RRSIG

Criticité : 🟢 Faible
Sources : RFC 6781 §4.4.1

```
# Vérifier la dispersion des dates d'expiration RRSIG
dig <domaine> RRSIG +short | awk '{print $5}' | sort | uniq -c | head -20
# Si toutes les dates sont identiques = pas de jitter
# Vérifier la configuration BIND :
grep "sig-validity-interval\|sig-signing-signatures\|sig-signing-nodes" /etc/named.conf
```

Résultat attendu : Les dates d'expiration RRSIG sont dispersées sur une plage de ±20% de la durée de validité. BIND ajoute automatiquement un jitter de ±20% par défaut depuis la version 9.10.

Effet de bord : Sans jitter, toutes les RRSIG d'une zone expirent simultanément → pic de charge CPU de signing massif à une heure fixe → risque de timeout du processus de signing → zone bogué si le signing dépasse la date d'expiration.

Remédiation : BIND gère le jitter automatiquement. Pour PowerDNS, le jitter est configurable via les politiques KASP. Vérifier la version du logiciel et la configuration.

Évaluation : ✅ ❌ ⚠️ N/A

S04-C004 — Présence de RRSIG pour TOUS les types RR présents dans la zone

Criticité : 🟠 Élevé
Sources : RFC 4035 §2.2

```
# Obtenir tous les types RR de la zone (nécessite accès au fichier de zone ou AXFR)
dig <domaine> AXFR @<ns-autoritatif> 2>/dev/null | awk '{print $4}' | sort -u
# Pour chaque type trouvé, vérifier la présence de RRSIG
for rr_type in $(dig <domaine> AXFR @<ns-auth> 2>/dev/null | awk '{print $4}' | sort -u | grep -v "RRSIG\|NSEC\|
NSEC3"); do
  rrsig=$(dig <domaine> RRSIG $rr_type +short 2>/dev/null)
  if [ -z "$rrsig" ]; then
    echo "⚠️ Pas de RRSIG pour $rr_type"
  fi
done
```

Résultat attendu : RRSIG présente pour chaque type RR dans la zone sauf les types non signables (RRSIG elle-même, NSEC/NSEC3 ont leur propre RRSIG, NS à l'apex est signé mais les NS de délégation ne le sont pas).

Effet de bord : Un type RR sans RRSIG dans une zone signée est traité comme "insecure" par les resolvers validants : ils acceptent la réponse sans valider mais émettent un avertissement. Ce comportement peut masquer une modification malveillante du RR non signé.

Remédiation : Re-signer la zone complète. Vérifier si un type RR a été ajouté manuellement sans déclencher le re-signing automatique.

Évaluation : ✅ ❌ ⚠️ N/A

S04-C005 — Vérification de la marge de re-signature (25% minimum)

Criticité : 🟠 Élevé
Sources : RFC 6781 §4.4.2

```
# Calculer la marge résiduelle des RRSIG critiques
dig <domaine> RRSIG SOA +short | awk '{
  exp=$5; inc=$6
  # Durée totale en approximation (différence des timestamps)
  printf "Durée totale: %s→%s\n", inc, exp
}'
# Vérifier configuration BIND : sig-validity-interval <total> <resign-at>
grep "sig-validity-interval" /etc/named.conf
# Ex: sig-validity-interval 14 4; = 14j de validité, re-signe à J-4 (28% de marge)
```

Résultat attendu : La re-signature est déclenchée quand 75% de la durée de validité est écoulée (25% restant). Exemple : RRSIG 14j → re-signe à J-3.5 minimum.

Effet de bord : Marge trop faible (< 10%) = si le serveur de signing est indisponible pendant la fenêtre de re-signature, la zone expire avant la prochaine tentative. Marge trop grande (> 50%) = signing fréquent et inutile.

Remédiation : Ajuster la configuration `sig-validity-interval` pour garantir une marge confortable. Documenter le calcul de la marge en fonction des TTL de la zone et de la fréquence maximale d'indisponibilité du serveur de signing.

Évaluation : ✅ ❌ ⚠️ N/A

S04-C006 — Test de la détection et réponse à une expiration RRSIG simulée

Criticité : 🟡 Moyen
Sources : RFC 6781 §7.1

```
# Simuler une RRSIG expirée en staging :
# 1. Arrêter le signing automatique
# 2. Attendre l'expiration naturelle ou modifier manuellement la date
# 3. Vérifier que le monitoring déclenche l'alerte
# 4. Appliquer la procédure de re-signature d'urgence
# Commande de re-signature forcée (BIND) :
rndc sign <domaine>
rndc loadkeys <domaine>
# Commande de re-signature forcée (PowerDNS) :
pdns_control rectify-zone <domaine>
pdns_control bind-reload-now <domaine>
```

Résultat attendu : L'équipe détecte l'expiration en moins de TTL_DNSKEY (1h) via monitoring. La re-signature d'urgence est appliquée en moins de 30 minutes. La zone est de nouveau validable en moins de 2h.

Effet de bord : Ce test doit impérativement être réalisé en staging, jamais en production sans plan de rollback immédiat. Si le test est réalisé en production par erreur : appliquer immédiatement `rndc sign` et surveiller la propagation.

Remédiation : Inclure ce test dans les exercices DNSSEC annuels. Documenter le temps de réponse réel vs objectif RTO.

Évaluation : ✅ ❌ ⚠️ N/A

S04-C007 — Vérification des RRSIG sur les enregistrements wildcards

Criticité : 🟡 Moyen
Sources : RFC 4035 §2.2.2, RFC 7129

```
# Tester la résolution d'un sous-domaine wildcard avec DNSSEC
dig test-wildcard.<domaine> A +dnssec
# La réponse doit inclure :
# - L'enregistrement A avec RRSIG
# - Un enregistrement NSEC/NSEC3 prouvant l'absence du RR exact
# - La RRSIG du NSEC/NSEC3
```

Résultat attendu : Les wildcards DNSSEC génèrent des réponses correctement signées avec preuve de non-existence du nom exact via NSEC/NSEC3. Bit NSEC3 opt-out correctement configuré si applicable.

Effet de bord : Les wildcards DNSSEC sont complexes à implémenter correctement. Une configuration incorrecte peut provoquer des SERVFAIL pour les noms wildcard ou des faux positifs de type "name does not exist" pour des noms couverts par le wildcard.

Remédiation : Tester les wildcards avec `delv` en mode trace complet. Vérifier la documentation du serveur DNS utilisé sur le comportement DNSSEC des wildcards.

Évaluation : ✅ ❌ ⚠️ N/A

S04-C008 — Cohérence des dates de signature entre tous les NS (multi-NS)

Criticité : 🟠 Élevé
Sources : RFC 4035 §4.1, RFC 6781 §3.3.4

```
# Comparer les RRSIG SOA sur plusieurs NS
NS_LIST=$(dig <domaine> NS +short)
for ns in $NS_LIST; do
  exp=$(dig @$ns <domaine> RRSIG SOA +short | awk '{print $5}' | head -1)
  echo "NS $ns — RRSIG SOA expire : $exp"
done
# Les dates d'expiration peuvent différer (jitter normal) mais ne doivent pas
# dépasser 24h d'écart si le signing est centralisé
```

Résultat attendu : Dates d'expiration RRSIG cohérentes entre tous les NS (écart acceptable < durée de jitter configuré). Aucun NS avec RRSIG significativement plus ancienne que les autres.

Effet de bord : Si un NS secondaire reçoit ses mises à jour par transfert de zone AXFR/IXFR retardé, ses RRSIG peuvent être moins fraîches que le NS primaire. Si le délai dépasse la durée de validité RRSIG, ce NS rend la zone bogué pour les resolvers qui l'interrogent.

Remédiation : Surveiller le délai de transfert de zone (AXFR/IXFR) entre NS primaire et secondaires. Configurer des alertes si le délai dépasse 50% de la durée de validité RRSIG.

Évaluation : ✅ ❌ ⚠️ N/A

S04-C009 — Documentation et test du processus de re-signature d'urgence

Criticité : 🟠 Élevé
Sources : RFC 6781 §7.1

```
# Procédure de re-signature d'urgence documentée et testée
# BIND :
echo "rndc sign <domaine> && rndc notify <domaine>"
# PowerDNS :
echo "pdns_control rectify-zone <domaine> && pdns_control notify <domaine>"
# Knot DNS :
echo "knotc zone-sign <domaine>"
# Vérifier après re-signature :
dig <domaine> RRSIG SOA +short | awk '{print "Nouvelle expiration:", $5}'
```

Résultat attendu : La procédure de re-signature d'urgence est documentée, accessible 24/7, et a été testée en staging. RTO ≤ 30 minutes du déclenchement de l'alerte à la re-signature effective.

Remédiation : Créer un runbook de re-signature d'urgence avec les commandes exactes pour chaque logiciel DNS utilisé. Inclure les commandes de vérification post-action.

Évaluation : ✅ ❌ ⚠️ N/A

S04-C010 — Vérification RRSIG sur les zones de délégation (glue records)

Criticité : 🟡 Moyen
Sources : RFC 4035 §2.2.3

```
# Les glue records (NS et A/AAAA des NS fils) dans une zone parente
# ne sont PAS signés (ils sont "out-of-zone" pour la zone parente)
# Vérifier que le DS des sous-zones déléguées est correctement signé
dig <domaine> RRSIG DS +short  # DS doit avoir RRSIG (signé par la zone parente)
```

Résultat attendu : Les enregistrements DS de délégation sont signés par la zone parente (RRSIG DS présente). Les glue records (NS/A/AAAA de la zone fille dans la zone parente) ne sont PAS signés et c'est normal (comportement RFC 4035 §2.2.3).

Remédiation : Si le DS n'a pas de RRSIG = bug de signing. Forcer la re-signature de la zone parente.

Évaluation : ✅ ❌ ⚠️ N/A

S05 — NSEC / NSEC3 / NSEC5 : Déni d'existence Authentifié

Contexte

NSEC (Next Secure), NSEC3 et NSEC5 sont les mécanismes qui permettent à DNSSEC de prouver qu'un nom de domaine N'EXISTE PAS dans une zone, ou qu'un type RR particulier est absent pour un nom existant. Sans

ce mécanisme, un attaquant pourrait supprimer des réponses NXDOMAIN et forcer les clients à croire qu'un domaine inexistant existe.

NSEC est le plus simple mais permet le "zone walking" : en suivant la chaîne NSEC, on peut énumérer tous les noms d'une zone, révélant potentiellement des informations sensibles (noms de serveurs internes, etc.). NSEC3 (RFC 5155) ajoute un hash (SHA-1 par défaut) des noms pour masquer l'énumération. Cependant, les iterations NSEC3 > 0 sont vulnérables aux attaques par dictionnaire offline (hashcat, John the Ripper). La RFC 9276 (2022) mandate iterations=0 pour neutraliser cette vulnérabilité. NSEC5 (draft-vcelak-nsec5) est une proposition post-quantum safe encore en standardisation.

Contrôles S05

S05-C001 — Vérification du type NSEC utilisé (NSEC vs NSEC3)

Criticité : 🟡 Moyen
Sources : RFC 4034 §4, RFC 5155 §1, RFC 9276

```
# Détecter si la zone utilise NSEC ou NSEC3
dig <domaine> NSEC3PARAM +short
# Si résultat vide = zone utilise NSEC (pas NSEC3)
# Si résultat présent = zone utilise NSEC3

# Test NXDOMAIN pour voir le type de réponse :
dig nonexistent-xyz123.<domaine> A +dnssec
# Chercher "NSEC" ou "NSEC3" dans la section additionnelle
```

Résultat attendu : NSEC3 est utilisé pour toutes les zones dont les noms pourraient révéler des informations sensibles si énumérés. NSEC est acceptable uniquement pour des zones totalement publiques dont le contenu intégral est déjà accessible (zones anycast open, zones de TLD, etc.).

Effet de bord : Le passage de NSEC à NSEC3 (ou vice versa) nécessite une re-signature complète de la zone et peut provoquer une brève augmentation du trafic DNS pendant la transition. NSEC3 augmente la taille des réponses NXDOMAIN (plusieurs enregistrements NSEC3 nécessaires pour certaines preuves).

Remédiation : Activer NSEC3 avec `dnssec-settime` ou la configuration KASP du serveur DNS. Pour BIND : `dnssec-policy <zone> { nsec3param salt auto iterations 0 optout no; };`

Évaluation : ✅ ❌ ⚠️ N/A

S05-C002 — Paramètres NSEC3 conformes RFC 9276 (iterations=0, salt aléatoire ≥ 128 bits)

Criticité : 🔴 Critique
Sources : RFC 9276 §3, RFC 5155

```
# Lire les paramètres NSEC3PARAM
dig <domaine> NSEC3PARAM +short
# Format : algo iterations flags salt
# Exemple conforme : 1 0 0 ab12cd34ef56...  (algo=1=SHA-1, iter=0, flags=0, salt hex)
# Exemple NON conforme : 1 10 0 deadbeef     (iter=10 = vulnérable!)

# Vérifier la longueur du salt (doit être ≥ 128 bits = 32 hex chars, recommandé)
dig <domaine> NSEC3PARAM +short | awk '{print "Iterations:", $2, "Salt:", $4, "Salt length (bits):",
length($4)*4}'
```

Résultat attendu : - Algorithme : 1 (SHA-1 — seul algorithme défini pour NSEC3) - **Iterations : 0** (RFC 9276 §3 — MUST) - Salt : chaîne hexadécimale aléatoire de 128 bits minimum (32 chars hex), changée à chaque rollover ZSK - Ou salt = "-" (tiret) = pas de salt (acceptable selon RFC 9276 avec iterations=0)

Effet de bord : NSEC3 avec iterations > 0 est vulnérable aux attaques par dictionnaire offline : un attaquant obtient les hashs NSEC3 et les passe dans hashcat avec une wordlist de domaines communs (mail, www, vpn,

ftp, admin, etc.). Avec des GPU modernes, 1 million de hashs/seconde par iter → iterations=25 = 25M hash/sec nécessaire → cracke des noms communs en quelques secondes. RFC 9276 a été publiée spécifiquement pour corriger cette vulnérabilité.

Remédiation :

```
# BIND — changer les paramètres NSEC3
rndc sign <zone>  # Après modification de dnssec-policy
# PowerDNS
pdns_control set-nsec3 <zone> '1 0 0 -'  # SHA-1, 0 iter, no flags, no salt
# Knot DNS — dans knot.conf
# nsec3: on
# nsec3-iterations: 0
# nsec3-salt-length: 16  # 128 bits
```

Évaluation : ✅ ❌ ⚠️ N/A

S05-C003 — Vérification de l'absence de NSEC3 avec opt-out non justifié

Criticité : 🟡 Moyen
Sources : RFC 5155 §6, RFC 7129

```
# Vérifier le flag opt-out dans NSEC3PARAM
dig <domaine> NSEC3PARAM +short | awk '{print "Flags:", $3, "(1=opt-out actif, 0=opt-out inactif)"}'
# Vérifier la présence de NS non signés
dig <domaine> AXFR @<ns-auth> 2>/dev/null | awk '$4=="NS" && $3!~/<domaine>/ {print "Délégation:", $3}'
```

Résultat attendu : Opt-out (flags=1) justifié uniquement si la zone contient plus de 50% de délégations vers des sous-zones non signées (cas typique : TLD avec de nombreux domaines non-DNSSEC). Pour les zones standard d'entreprise, opt-out=0 (désactivé) est requis.

Effet de bord : Avec opt-out activé, les délégations vers des sous-zones non signées ne sont pas couvertes par un enregistrement NSEC3. Cela signifie qu'un attaquant peut insérer des fausses réponses NXDOMAIN pour des sous-zones déléguées non signées sans être détecté par les resolvers validants.

Remédiation : Désactiver opt-out pour les zones d'entreprise. Si opt-out est nécessaire, documenter les sous-zones non signées et planifier leur signature.

Évaluation : ✅ ❌ ⚠️ N/A

S05-C004 — Test du déni d'existence authentifié (NXDOMAIN DNSSEC)

Criticité : 🟠 Élevé
Sources : RFC 4035 §3.2.2, RFC 5155 §7, RFC 7129

```
# Test NXDOMAIN avec DNSSEC
dig nonexistant-name-xyz987.<domaine> A +dnssec @9.9.9.9
# La réponse doit inclure :
# - RCODE: NXDOMAIN
# - Section additionnelle avec NSEC3 (ou NSEC)
# - RRSIG du NSEC3/NSEC

# Test avec delv pour validation complète
delv nonexistant-xyz.<domaine> A @9.9.9.9 +vtrace 2>&1 | grep -E "validated|BOGUS|nxdomain"
```

Résultat attendu : Réponse NXDOMAIN avec NSEC3 authentifié (RRSIG valide sur le NSEC3). delv confirme la validation. Aucune réponse "insecure" pour un domaine NXDOMAIN dans une zone signée.

Effet de bord : Si la preuve NSEC3 est absente ou invalide sur les NXDOMAIN, les resolvers validants retournent SERVFAIL (zone considérée bogué). Ce bug se manifeste typiquement après une migration vers NSEC3 ou un changement de paramètres.

Remédiation : Re-signer la zone complète après changement de paramètres NSEC3. Vérifier avec `ldns-verify-zone` si le fichier de zone est cohérent.

Évaluation : ✅ ❌ ⚠️ N/A

S05-C005 — Test du déni de type (NODATA DNSSEC)

Criticité : 🟡 Moyen
Sources : RFC 4035 §3.2.3, RFC 5155 §7.2.3

```
# Test NODATA : nom existant, type absent
dig <domaine> CAA +dnssec @9.9.9.9  # Si pas de CAA, doit retourner NODATA avec NSEC3
# Chercher NSEC3 dans la réponse prouvant l'absence du type CAA
dig <domaine> MX +dnssec @9.9.9.9 | grep -E "NSEC3|NSEC"
```

Résultat attendu : Réponse NODATA (RCODE NOERROR, section answer vide) avec NSEC3 prouvant l'absence du type demandé pour ce nom. Bit AD présent dans la réponse.

Effet de bord : NODATA avec NSEC3 incorrect peut conduire certains resolvers à traiter le domaine comme "insecure", perdant le bénéfice de la validation DNSSEC pour ce type RR.

Évaluation : ✅ ❌ ⚠️ N/A

S05-C006 — Évaluation de la migration vers NSEC5 (post-quantique safe)

Criticité : 🟢 Faible
Sources : draft-vcelak-nsec5-09, RFC 9276 §4

```
# NSEC5 = Verifiable Random Function (VRF) à la place du hash SHA-1
# Avantage : résistance au zone-walking même avec iterations=0
# Statut 2026 : draft IETF, pas encore adopté en RFC
# Vérifier si le serveur DNS supporte NSEC5 (expérimental uniquement)
```

Résultat attendu : Suivi du statut du draft NSEC5 au sein du groupe DNSOP de l'IETF. Plan de migration documenté pour adoption dès standardisation finale. En 2026, NSEC3 avec iterations=0 reste la solution recommandée.

Effet de bord : NSEC5 avec VRF nécessite une cryptographie plus complexe que NSEC3 SHA-1. Performances de génération légèrement inférieures. La standardisation finale peut encore évoluer.

Remédiation : Inscrire le suivi NSEC5 dans la veille technologique DNSSEC de l'organisation. Tester NSEC5 en lab sur Knot DNS (première implémentation expérimentale).

Évaluation : ✅ ❌ ⚠️ N/A

S05-C007 — Vérification de la couverture NSEC3 pour tous les types RR à l'apex

Criticité : 🟠 Élevé
Sources : RFC 5155 §7.1

```
# À l'apex de zone, NSEC3 doit couvrir tous les types présents
dig <domaine> NSEC3 +dnssec @<ns-auth>
# Vérifier le champ "type bitmaps" du NSEC3 à l'apex
# Il doit lister A, NS, SOA, MX, TXT, AAAA, RRSIG, NSEC3PARAM, DNSKEY etc.
dig nonexistent.<domaine> +dnssec | grep "NSEC3" | head -3
```

Résultat attendu : Le NSEC3 couvrant l'apex liste tous les types RR présents. L'absence d'un type dans le bitmap NSEC3 peut provoquer des NODATA incorrects.

Remédiation : Re-signer la zone pour recalculer les bitmaps NSEC3. Vérifier avec `ldns-verify-zone --verbose`.

Évaluation : ✅ ❌ ⚠️ N/A

S05-C008 — Test de zone walking sur NSEC (si NSEC utilisé)

Criticité : 🟠 Élevé (si NSEC utilisé)
Sources : RFC 4034 §4.1, RFC 5155 §1.3

```
# Si la zone utilise NSEC (pas NSEC3), tester le zone walking
# L'outil ldns-walk permet d'énumérer une zone via NSEC :
ldns-walk <domaine> 2>/dev/null | head -20
# Si des noms sensibles apparaissent = risque d'information disclosure
```

Résultat attendu : Si NSEC est utilisé, s'assurer que la zone ne contient aucun nom sensible (serveurs internes, noms de services confidentiels) qui ne devrait pas être publiquement énumérable. Sinon, migrer vers NSEC3.

Effet de bord : Le zone walking via NSEC révèle 100% des noms de la zone. Pour les zones publiques avec uniquement des informations publiques, c'est acceptable. Pour les zones contenant des noms de serveurs internes accessibles depuis Internet, c'est une fuite d'information.

Remédiation : Migrer vers NSEC3 si la zone contient des noms sensibles. Séparer les zones publiques et privées si nécessaire.

Évaluation : ✅ ❌ ⚠️ N/A

S05-C009 — Vérification que le sel NSEC3 est renouvelé à chaque rollover ZSK

Criticité : 🟡 Moyen
Sources : RFC 5155 §12.1.3, RFC 9276 §3.1

```
# Historique des sels NSEC3 (vérifier le journal du serveur DNS)
grep -i "nsec3\|salt" /var/log/named/named.log 2>/dev/null | tail -20
# Le sel actuel :
dig <domaine> NSEC3PARAM +short | awk '{print "Sel actuel:", $4}'
# Comparer avec le sel de la semaine/du mois précédent dans les archives DNS
```

Résultat attendu : Le sel NSEC3 est renouvelé automatiquement lors de chaque rollover ZSK, ou au minimum trimestriellement. Les anciens sels ne doivent pas réapparaître.

Effet de bord : Un sel NSEC3 statique utilisé pendant des années permet de construire un index offline des hashs NSEC3 (rainbow table DNS), facilitant le zone walking même avec iterations=0. Le renouvellement régulier du sel invalide ces tables précomputées.

Remédiation : Configurer le renouvellement automatique du sel dans la politique DNSSEC (BIND KASP, Knot DNS policy). En cas de renouvellement manuel, ajouter une tâche calendaire trimestrielle.

Évaluation : ✅ ❌ ⚠️ N/A

S05-C010 — Test de cohérence NSEC3 via ldns-verify-zone

Criticité : 🟡 Moyen
Sources : RFC 5155, ldnsutils documentation

```
# Télécharger le fichier de zone (si AXFR autorisé depuis le serveur d'audit)
dig <domaine> AXFR @<ns-interne> > /tmp/<domaine>.zone 2>/dev/null
# Vérifier la cohérence cryptographique de la zone
ldns-verify-zone /tmp/<domaine>.zone
# Résultat attendu : "Zone is verified and complete."
```

Résultat attendu : `ldns-verify-zone` confirme : toutes les RRSIG valides, tous les NSEC3 cohérents, chaîne DNSKEY valide, aucun RR orphelin.

Effet de bord : L'autorisation AXFR depuis le serveur d'audit doit être limitée (ACL IP stricte). Ne pas exposer l'AXFR depuis Internet.

Remédiation : Si ldns-verify-zone détecte des erreurs : identifier les RR incohérents et forcer une re-signature complète de la zone.

Évaluation : ✅ ❌ ⚠️ N/A

S05-C011 — Vérification de l'absence de boucles NSEC (zone corrompue)

Criticité : 🟠 Élevé
Sources : RFC 4034 §4.1.1

```
# Vérifier que la chaîne NSEC ne forme pas de boucle (bug de signing)
# Outil de détection :
ldns-verify-zone /tmp/<domaine>.zone 2>&1 | grep -i "loop\|cycle\|chain"
# Vérification manuelle : le dernier NSEC doit pointer vers le premier nom de la zone
```

Résultat attendu : Chaîne NSEC/NSEC3 linéaire sans boucle. Le dernier enregistrement NSEC pointe vers le premier nom de la zone (apex). Pas de NSEC orphelin.

Remédiation : Recréer la zone signée depuis le fichier de zone brut non signé. Identifier le bug de signing et mettre à jour le logiciel si nécessaire.

Évaluation : ✅ ❌ ⚠️ N/A

S05-C012 — Documentation de la politique NSEC3 dans le DPS

Criticité : 🟢 Faible
Sources : RFC 6781 §3.2.4, RFC 9276

Résultat attendu : Le DPS documente : choix NSEC vs NSEC3, paramètres NSEC3 (algorithme, iterations=0, longueur du sel, fréquence de renouvellement du sel), justification du choix opt-out ou non, procédure de migration si changement de paramètres.

Remédiation : Ajouter une section NSEC3 au DPS existant, ou créer le document si inexistant.

Évaluation : ✅ ❌ ⚠️ N/A

S06 — Enregistrements DS et Chaîne de Confiance

Contexte

L'enregistrement DS (Delegation Signer) est le pivot de la chaîne de confiance DNSSEC. Stocké dans la zone parente, il contient l'empreinte cryptographique (hash) de la KSK de la zone fille. Quand un resolver valide une zone, il récupère le DS depuis le parent, calcule l'empreinte de la DNSKEY de la zone, et compare : si les deux correspondent, la zone est considérée sécurisée.

Le format DS est : `<keytag> <algorithme> <type-digest> <digest>`. Le type de digest 1 (SHA-1) est obsolète et interdit par RFC 8624. Le type 2 (SHA-256) est le standard actuel. Le type 4 (SHA-384) est recommandé pour les usages à haute sécurité. Un DS orphelin (DNSKEY supprimée mais DS toujours présent chez le parent) rend instantanément la zone bogué pour 100% des resolvers validants.

Contrôles S06

S06-C001 — Présence et validité du DS chez le registrar/parent

Criticité : 🔴 Critique
Sources : RFC 4034 §5, RFC 6840 §5.1

```
# Vérifier DS depuis un resolver validant
dig <domaine> DS +short @9.9.9.9
# Vérifier DS directement chez le NS parent (TLD)
TLD_NS=$(dig <tld> NS +short | head -1)
dig @$TLD_NS <domaine> DS +short
# Cross-check keytag DS vs keytag DNSKEY KSK
echo "DS keytags :" && dig <domaine> DS +short | awk '{print $1}'
echo "DNSKEY (KSK) keytags :" && dig <domaine> DNSKEY +short | awk '{if($1==257) print $4}' | while read b64; do
  echo "$b64" | python3 -c "import sys,base64,struct; d=base64.b64decode(sys.stdin.read()); a=0; [a.__add__(b<<8)
if i%2==0 else a.__add__(b) for i,b in enumerate(d)]; print('keytag calcul')"
done
```

Résultat attendu : DS présent chez le parent, keytag correspondant à la KSK active, digest type 2 (SHA-256) ou 4 (SHA-384). Au minimum un DS valide doit être présent pour chaque KSK active.

Effet de bord : DS absent = zone non validable par les resolvers (traitée comme "insecure" si le parent ne publie pas de DS, ou bogué si le parent publie un DS incorrect). Un seul DS incorrect suffit à rendre la zone bogué si c'est le seul DS présent.

Remédiation : Soumettre le DS correct au registrar via l'interface d'administration. Attendre la propagation (TTL DS du parent, généralement 86400s). Vérifier la propagation depuis plusieurs points de mesure géographiques.

Évaluation : ✅ ❌ ⚠️ N/A

S06-C002 — Absence de DS avec digest type 1 (SHA-1) obsolète

Criticité : 🔴 Critique
Sources : RFC 8624 §3.1 (MUST NOT use for signing)

```
# Détecter les DS SHA-1 (type 1)
dig <domaine> DS +short | awk '{
  if($3==1) print "❌ DS DIGEST TYPE 1 (SHA-1) DÉTECTÉ — OBSOLÈTE RFC 8624 !"
  if($3==2) print "✅ DS type 2 (SHA-256) — conforme"
  if($3==4) print "✅ DS type 4 (SHA-384) — conforme"
}'
```

Résultat attendu : Aucun DS avec digest type 1 (SHA-1). Tous les DS utilisent le type 2 (SHA-256) ou le type 4 (SHA-384).

Effet de bord : SHA-1 est cryptographiquement cassé (collision attacks, SHAttered 2017). Un attaquant avec des ressources importantes peut forger un DS SHA-1 correspondant à une DNSKEY malveillante. La RFC 8624 interdit formellement SHA-1 pour les nouveaux déploiements DNSSEC.

Remédiation : Contacter le registrar pour remplacer le DS SHA-1 par un DS SHA-256 ou SHA-384. Si le registrar n'accepte que SHA-1, escalader (service non conforme RFC 8624) et envisager une migration de registrar.

Évaluation : ✅ ❌ ⚠️ N/A

S06-C003 — Vérification de la cohérence DS ↔ DNSKEY (keytag, algo, digest)

Criticité : 🔴 Critique
Sources : RFC 4034 §5.1, RFC 6840

```
# Calculer le DS attendu à partir de la DNSKEY et comparer avec le DS publié
# Méthode avec dnssec-dsfromkey (BIND utils) :
dig <domaine> DNSKEY +short | while read flags proto algo key; do
  if [ "$flags" = "257" ]; then  # KSK only
    echo "$flags $proto $algo $key" | dnssec-dsfromkey -f /dev/stdin <domaine>
  fi
done
# Comparer la sortie avec :
dig <domaine> DS +short

# Méthode Python :
python3 -c "
import dns.resolver, dns.dnssec, dns.rdata, sys
answers = dns.resolver.resolve('<domaine>', 'DNSKEY')
for rdata in answers:
    if rdata.flags & 0x0100:  # KSK
        ds = dns.dnssec.make_ds('<domaine>.', rdata, 'SHA256')
        print('DS calculé:', ds)
"
```

Résultat attendu : Le hash du DS chez le parent correspond exactement au hash calculé à partir de la DNSKEY active. Keytag, algorithme et digest identiques.

Effet de bord : Une incohérence entre DS et DNSKEY peut survenir lors d'un rollover KSK mal exécuté (nouveau DS soumis mais ancienne DNSKEY toujours en place, ou nouvelle DNSKEY activée avant propagation du nouveau DS). Ce scénario est la zone bogué la plus courante.

Remédiation : Identifier la source de l'incohérence (DS chez parent ou DNSKEY dans zone). Ne retirer l'ancienne clé qu'après confirmation de propagation du nouveau DS (vérifier depuis plusieurs DNS autoritatifs du parent et depuis des resolvers géodistribués).

Évaluation : ✅ ❌ ⚠️ N/A

S06-C004 — Présence d'un DS de type SHA-384 pour les zones critiques

Criticité : 🟢 Faible
Sources : RFC 8624 §3.1, NIST SP 800-81r2, ANSSI REC-DNSSEC-2021

```
# Vérifier si un DS SHA-384 (type 4) est publié
dig <domaine> DS +short | awk '{if($3==4) print "✅ DS SHA-384 présent"; else if($3==2) print "DS SHA-256 présent (conforme mais non maximal)"}'
```

Résultat attendu : Pour les zones critiques (OIV, secteur financier, gouvernement), un DS de type 4 (SHA-384) est publié en plus ou à la place du DS SHA-256. Les registrars ne supportent pas tous SHA-384 — vérifier la compatibilité.

Effet de bord : SHA-384 est légèrement plus coûteux en calcul mais offre une marge de sécurité supérieure. Son principal avantage est la résistance accrue aux attaques futures sur SHA-256. Pour les registrars ne supportant pas SHA-384, le SHA-256 reste le standard acceptable.

Évaluation : ✅ ❌ ⚠️ N/A

S06-C005 — Procédure de mise à jour du DS chez le registrar documentée et testée

Criticité : 🟠 Élevé
Sources : RFC 7344, RFC 6781 §4.1

```
# Documenter les étapes de mise à jour DS chez chaque registrar :
# 1. Interface web ou API (certains registrars ont une API EPP)
# 2. Délai de propagation du registrar vers le TLD
# 3. Contact support en cas de blocage
# Tester la procédure en staging si possible
# Vérifier le délai SLA du registrar pour la mise à jour DS
```

Résultat attendu : Procédure documentée pour chaque registrar utilisé, incluant : URL d'accès, méthode de mise à jour (UI/API/email), format DS requis, délai moyen observé, contact support d'urgence.

Effet de bord : Lors d'un rollover KSK d'urgence, un délai registrar de 24-48h peut maintenir la zone bogué pendant toute cette période. Certains registrars ont des délais de traitement manuel importants le week-end.

Remédiation : Tester le processus de mise à jour DS au moins une fois en non-urgence pour connaître le délai réel. Identifier les contacts d'escalade du registrar. Envisager CDS/CDNSKEY si le registrar le supporte (automatisation).

Évaluation : ✅ ❌ ⚠️ N/A

S06-C006 — Vérification de la propagation DS depuis plusieurs points géographiques

Criticité : 🟠 Élevé
Sources : RFC 4035, RIPE Atlas

```
# Vérifier la propagation DS depuis plusieurs resolvers
for resolver in 9.9.9.9 1.1.1.1 8.8.8.8 208.67.222.222; do
  echo "=== Resolver $resolver ==="
  dig @$resolver <domaine> DS +short
done
# Utiliser RIPE Atlas si disponible pour des sondes géodistribuées
```

Résultat attendu : Le même DS est visible depuis tous les resolvers testés. Aucune incohérence géographique (qui indiquerait une propagation incomplète ou un hijacking BGP ciblé).

Effet de bord : Des incohérences géographiques peuvent indiquer un problème de propagation DNS (NS secondaires non synchronisés) ou un incident BGP (annonce de prefixe malveillante sur un lien spécifique). Les incohérences transitoires sont normales pendant la propagation (< 24h).

Remédiation : Attendre la propagation complète (TTL DS du parent). Si persistant après 48h, contacter le registrar.

Évaluation : ✅ ❌ ⚠️ N/A

S06-C007 — Absence de DS orphelins (DS sans DNSKEY correspondante)

Criticité : 🔴 Critique
Sources : RFC 6840 §5.1

```
# Extraire les keytags des DS chez le parent
DS_KEYTAGS=$(dig <domaine> DS +short | awk '{print $1}')
# Extraire les keytags des DNSKEY actives
DNSKEY_KEYTAGS=$(dig <domaine> DNSKEY +short | awk '{if($1==257) print $4}')
# Comparer (simplification — dans la pratique utiliser dns.dnssec.make_ds)
echo "DS keytags: $DS_KEYTAGS"
echo "DNSKEY (KSK) keytags calculés: $DNSKEY_KEYTAGS"
```

Résultat attendu : Chaque keytag DS a une DNSKEY KSK correspondante dans la zone. Aucun DS ne référence une DNSKEY absente.

Effet de bord : Un DS orphelin (DNSKEY supprimée mais DS laissé chez le parent) est la cause numéro un de zone bogué lors d'un rollover KSK. Les resolvers voient un DS mais ne trouvent pas de DNSKEY correspondante → validation échoue → SERVFAIL.

Remédiation : Lors de tout rollover KSK : ne retirer l'ancien DS chez le parent QUE après avoir confirmé que l'ancienne DNSKEY est toujours présente dans la zone (les deux clés co-existent temporairement). Ne retirer l'ancienne DNSKEY QUE après que l'ancien DS a été retiré du parent et que la propagation est confirmée.

Évaluation : ✅ ❌ ⚠️ N/A

S06-C008 — Test de validation de la chaîne complète avec delv

Criticité : 🟠 Élevé
Sources : RFC 4035, RFC 6840

```
# Test de validation complète depuis la racine
delv <domaine> A +vtrace +rtrace 2>&1 | grep -E "fully validated|BOGUS|insecure|error"
# Test avec ancre de confiance explicite (root DNSKEY)
delv <domaine> A +root=/usr/share/dns/root.ds 2>&1 | tail -5
# Test délégation sécurisée :
delv <sous-domaine>.<domaine> A +vtrace 2>&1 | grep -E "validated|BOGUS"
```

Résultat attendu : `<domaine> is fully validated` — toute la chaîne root → TLD → domaine est valide. Absence de `BOGUS` ou `insecure`. Délégations des sous-zones signées également validées.

Remédiation : Si `BOGUS` : utiliser `+vtrace` pour identifier quel maillon de la chaîne échoue. Corriger le DS ou la DNSKEY concernée.

Évaluation : ✅ ❌ ⚠️ N/A

S06-C009 — Vérification DS pour les sous-domaines délégués critiques

Criticité : 🟡 Moyen
Sources : RFC 4035 §2.3

```
# Lister les sous-domaines délégués dans la zone
dig <domaine> AXFR @<ns-auth> 2>/dev/null | awk '$4=="NS" && $1!="<domaine>." {print $1}' | sort -u
# Pour chaque délégation critique, vérifier la présence du DS
for subdomain in mail vpn api admin; do
  DS=$(dig $subdomain.<domaine> DS +short 2>/dev/null)
  DNSKEY=$(dig $subdomain.<domaine> DNSKEY +short 2>/dev/null)
  if [ -n "$DNSKEY" ] && [ -z "$DS" ]; then
    echo "⚠️ $subdomain.<domaine> : zone signée sans DS dans parent !"
  fi
done
```

Résultat attendu : Toutes les sous-zones déléguées critiques ont leur DS publié dans la zone parente, maintenant la chaîne de confiance complète.

Remédiation : Publier le DS de la sous-zone dans la zone parente via la commande `dnssec-dsfromkey` ou l'interface du serveur DNS parent.

Évaluation : ✅ ❌ ⚠️ N/A

S06-C010 — Monitoring automatique de la cohérence DS/DNSKEY

Criticité : 🟠 Élevé
Sources : RFC 6781 §6, ICANN DNSSEC Monitoring

```
# Script de monitoring DS/DNSKEY cohérence (à intégrer dans cron)
python3 << 'EOF'
import dns.resolver, dns.dnssec, dns.name
domain = dns.name.from_text('<domaine>.')
dnskeys = dns.resolver.resolve(domain, 'DNSKEY')
ds_records = dns.resolver.resolve(domain, 'DS')
# Calculer les DS attendus depuis les DNSKEY KSK
for rdata in dnskeys:
    if rdata.flags & 0x0100:  # KSK
        expected_ds = dns.dnssec.make_ds(domain, rdata, 'SHA256')
        found = any(str(ds) == str(expected_ds) for ds in ds_records)
        print(f"KSK keytag {rdata.key_tag}: DS {'✅' if found else '❌ MANQUANT'}")
EOF
```

Résultat attendu : Script de monitoring exécuté toutes les 6h minimum, alertant sur toute incohérence DS/DNSKEY via email/PagerDuty/Slack.

Remédiation : Intégrer ce contrôle dans le système de monitoring existant (Nagios, Zabbix, Prometheus avec alertmanager). Définir le seuil d'alerte : toute incohérence déclenche une alerte P1.

Évaluation : ✅ ❌ ⚠️ N/A

S07 — Rollover KSK (RFC 7583)

Contexte

Le rollover KSK est l'opération la plus risquée de DNSSEC. Il nécessite une coordination précise entre le serveur DNS autoritatif et le registrar parent, car toute désynchronisation peut rendre la zone bogué pour l'ensemble des resolvers validants sur Internet. La RFC 7583 (DNSSEC Key Rollover Timing Considerations) formalise les durées minimales à respecter.

Le rollover KSK standard se déroule en 5 phases : (1) publication de la nouvelle KSK dans la zone ; (2) attente propagation = TTL_DNSKEY + transport ; (3) soumission du nouveau DS au registrar ; (4) attente propagation DS = TTL_DS_parent + transport ; (5) retrait de l'ancienne KSK et de son DS. Toute tentative d'accélération de ces phases provoque une zone bogué.

L'incident ICANN root KSK rollover en 2018 a montré que même les organisations les mieux préparées peuvent rencontrer des problèmes lors de cette opération : 15% des resolvers Internet avaient des ancres de confiance obsolètes, provoquant des SERVFAIL.

Contrôles S07

S07-C001 — Vérification que la durée de vie de la KSK ne dépasse pas 2 ans

Criticité : 🟠 Élevé
Sources : RFC 6781 §3.1.1, RFC 7583 §3

```
# Trouver la date d'activation de la KSK courante
# BIND : lire les métadonnées du fichier de clé
grep "Activate:" /var/cache/bind/K<domaine>.+013+*.private 2>/dev/null
# Date de création implicite = date du fichier si pas de métadonnée
ls -la /var/cache/bind/K<domaine>.+013+*.key 2>/dev/null
# Calculer l'âge :
date -d "$(grep 'Activate:' /var/cache/bind/K<domaine>*.private | head -1 | awk '{print $2}')" '+%Y-%m-%d' 2>/dev/
null
```

Résultat attendu : La KSK active a été créée/activée il y a moins de 24 mois. Un rollover KSK est planifié dans le calendrier DNS si l'échéance approche (alerte à J-90).

Effet de bord : Une KSK très ancienne accumule plus de matériel signé avec la même clé (risque cryptanalytique théorique croissant). Les anciens algorithmes utilisés pour des KSK créées il y a > 3 ans peuvent aujourd'hui ne plus respecter RFC 8624.

Remédiation : Planifier le rollover KSK dans un créneau de faible risque (semaine standard, pas de jour férié, équipe complète disponible). Durée minimale du rollover = TTL_DNSKEY + TTL_DS_parent × 2 (typiquement 2-4 jours).

Évaluation : ✅ ❌ ⚠️ N/A

S07-C002 — Existence d'une procédure de rollover KSK documentée et testée

Criticité : 🔴 Critique
Sources : RFC 7583, RFC 6781 §4.1

```
# Vérifier l'existence du runbook de rollover KSK
# Le runbook doit spécifier :
# Phase 1 : dnssec-keygen -a ECDSAP256SHA256 -f KSK <domaine>
# Puis : dnssec-settime -P <date-publish> -A <date-activate> <keyfile>
# Phase 2 : rndc loadkeys <domaine>
# Vérifier que la nouvelle KSK est visible :
# dig <domaine> DNSKEY +short  # doit montrer 2 KSK (ancienne + nouvelle)
# Phase 3 : calculer et soumettre le nouveau DS
# dnssec-dsfromkey <new-ksk.key>
# → soumettre au registrar
# Phase 4 : attendre TTL_DS propagation (vérifier depuis plusieurs resolvers)
# Phase 5 : dnssec-settime -I <date-inactive> -D <date-delete> <old-ksk>
```

Résultat attendu : Runbook détaillé existant, testé en staging, avec : commandes exactes, durées d'attente calculées basées sur les TTL réels, critères de validation à chaque étape, procédure de rollback.

Effet de bord : Sans runbook testé, chaque rollover KSK est une improvisation sous pression. L'expérience ICANN montre que même avec un runbook, des problèmes imprévus surviennent. Le staging permet de découvrir ces problèmes sans impact production.

Remédiation : Rédiger et tester le runbook. Inclure un arbre de décision pour les cas d'erreur (que faire si le DS ne se propage pas après 48h, que faire si le registrar est injoignable, etc.).

Évaluation : ✅ ❌ ⚠️ N/A

S07-C003 — Respect des durées minimales inter-phases (RFC 7583)

Criticité : 🔴 Critique
Sources : RFC 7583 §3.3

```
# Calculer les durées minimales pour votre zone
TTL_DS=$(dig <domaine> DS +ttl | grep "^<domaine>" | awk '{print $2}')
TTL_DNSKEY=$(dig <domaine> DNSKEY +ttl | grep "^<domaine>" | awk '{print $2}')
echo "TTL DS (parent)     : ${TTL_DS}s = $((TTL_DS/3600))h"
echo "TTL DNSKEY          : ${TTL_DNSKEY}s = $((TTL_DNSKEY/3600))h"
echo "---"
echo "Phase 1→2 min (publish→DS soumis)  : $((TTL_DNSKEY/3600 + 2))h (TTL_DNSKEY + 2h transport)"
echo "Phase 3→4 min (DS soumis→DS actif) : $((TTL_DS/3600 + 24))h (TTL_DS + 24h propagation)"
echo "Phase 4→5 min (DS actif→old KSK    : $((TTL_DS/3600 + 2))h (TTL_DS + transport)"
```

Résultat attendu : Chaque phase du rollover KSK respecte les durées minimales calculées. Le rollover complet prend typiquement 3-7 jours pour des TTL standards (DNSKEY 3600s, DS 86400s).

Effet de bord : Passer à la phase suivante avant la durée minimale signifie qu'une partie des resolvers n'a pas encore mis à jour leur cache → SERVFAIL intermittent (dépend du resolver qui répond aux clients). L'intermittence rend le diagnostic difficile ("ça marche chez moi mais pas chez le client").

Remédiation : Ajouter des checkpoints chronométrés dans le runbook. Utiliser une checklist à cocher avec les horaires réels à chaque étape.

Évaluation : ✅ ❌ ⚠️ N/A

S07-C004 — Vérification de la co-existence des deux KSK pendant la transition

Criticité : 🔴 Critique
Sources : RFC 6781 §4.1.2, RFC 7583 §4.1

```
# Pendant le rollover, les deux KSK doivent être publiées simultanément
dig <domaine> DNSKEY +short | awk '{if($1==257) print "KSK:", $3, $4}' | wc -l
# Attendu pendant la phase de transition : 2 lignes (2 KSK)
# Les deux DNSKEY doivent avoir un RRSIG DNSKEY correspondant
dig <domaine> RRSIG DNSKEY +short | awk '{print "Keytag signataire:", $7}' | sort -u
```

Résultat attendu : Pendant les phases 1-4 du rollover : exactement 2 DNSKEY avec flags 257 (KSK) dans la zone. Les RRSIG DNSKEY peuvent être signées par l'une ou les deux KSK selon l'implémentation.

Effet de bord : Retirer l'ancienne KSK avant que le nouveau DS soit propagé = zone bogué immédiate pour tous les resolvers qui ont encore l'ancien DS en cache. Ajouter la nouvelle KSK sans la publier suffisamment longtemps = resolvers sans la nouvelle clé en cache lors de la transition DS.

Remédiation : Monitoring temps réel du nombre de DNSKEY pendant le rollover. Procédure de rollback : retirer la nouvelle KSK et son DS si problème détecté en phase 3 ou 4.

Évaluation : ✅ ❌ ⚠️ N/A

S07-C005 — Procédure de rollover KSK d'urgence (compromission de clé)

Criticité : 🔴 Critique
Sources : RFC 6781 §4.1.3, ICANN DNSSEC Emergency Procedures

```
# Scénario : KSK compromise détectée
# Actions immédiates :
# 1. Générer immédiatement une nouvelle KSK
dnssec-keygen -a ECDSAP256SHA256 -f KSK <domaine>
# 2. Publier la nouvelle KSK dans la zone
rndc loadkeys <domaine>
# 3. Contacter le registrar EN URGENCE pour mise à jour DS immédiate
# 4. Publier un DNSKEY avec le bit REVOKE positionné pour l'ancienne KSK
#    (RFC 5011 — signale aux resolvers d'invalider leur ancre de confiance)
dnssec-settime -R now <old-ksk-file>
# 5. Après propagation du nouveau DS : retirer l'ancienne KSK révoquée
```

Résultat attendu : Procédure d'urgence documentée, testée (en staging), avec : temps de réponse cible < 4h depuis détection, contact registrar d'urgence identifié, arbre de décision pour chaque scénario.

Effet de bord : Un rollover KSK d'urgence accepte le risque d'une zone bogué temporaire pour limiter le risque de compromission prolongée. La zone peut être bogué pendant le délai de propagation du nouveau DS (24-48h selon registrar/TTL).

Remédiation : Avoir pré-établi un contact d'urgence avec le registrar (hotline, numéro direct). Certains registrars premium offrent des SLA de mise à jour DS < 1h. Documenter ce SLA dans le DPS.

Évaluation : ✅ ❌ ⚠️ N/A

S07-C006 — Automatisation du rollover KSK via CDS/CDNSKEY si registrar le supporte

Criticité : 🟢 Faible
Sources : RFC 7344, RFC 8078, RFC 7583 §Annexe A

```
# Vérifier si le registrar de la zone supporte CDS/CDNSKEY
# Liste IANA : https://www.iana.org/assignments/cds-cdnskey-registrars/
# Vérifier si CDS est actuellement publié dans la zone
dig <domaine> CDS +short
dig <domaine> CDNSKEY +short
# Pour publier CDS (BIND KASP) :
# parent-ds-ttl 86400; parent-registration-delay 0; (dans dnssec-policy)
```

Résultat attendu : Si le registrar supporte CDS/CDNSKEY : workflow de rollover KSK automatisé configuré et testé. Le CDS est publié lors des rollovers et retiré après confirmation DS mis à jour.

Effet de bord : CDS/CDNSKEY publiés en permanence (et non uniquement pendant les rollovers) peuvent déclencher des mises à jour DS non désirées si mal configurés. Certains registrars ont un délai de polling CDS (12-24h).

Remédiation : Configurer le polling CDS côté registrar si possible. Vérifier la documentation du registrar sur le support CDS. Tester un rollover CDS complet en environnement de test.

Évaluation : ✅ ❌ ⚠️ N/A

S07-C007 — Test de rollover KSK complet en environnement de staging

Criticité : 🟠 Élevé
Sources : RFC 7583, RFC 6781 §7

```
# Séquence de test rollover KSK en staging (avec ancre de confiance locale)
# 1. Créer un resolver de test avec l'ancre de confiance de la zone de test
#    unbound.conf : trust-anchor-file: /etc/unbound/test-zone.ds
# 2. Exécuter le rollover complet en accéléré (TTL réduits pour le test)
# 3. Vérifier la validation depuis le resolver de test à chaque étape
# 4. Documenter les anomalies rencontrées
```

Résultat attendu : Rollover KSK complet testé en staging sans zone bogué. Durée réelle du test documentée. Anomalies identifiées et corrigées dans le runbook.

Remédiation : Planifier ce test annuellement minimum. Réduire les TTL dans l'environnement de staging pour accélérer le test (TTL_DNSKEY=300s, TTL_DS=300s) — jamais en production.

Évaluation : ✅ ❌ ⚠️ N/A

S07-C008 — Monitoring de l'état du rollover KSK en temps réel

Criticité : 🟠 Élevé
Sources : RFC 6781 §6, RFC 7583

```
# Dashboard de monitoring pendant un rollover KSK actif
watch -n 60 '
echo "=== $(date) ==="
echo "DNSKEY dans zone :"
dig <domaine> DNSKEY +short | awk "{if(\$1==257) print \"  KSK:\",\$4}"
echo "DS chez parent :"
dig <domaine> DS +short @8.8.8.8
echo "DS chez parent (autre resolver) :"
dig <domaine> DS +short @1.1.1.1
echo "Validation :"
delv <domaine> A 2>&1 | grep -E "validated|BOGUS|insecure"
'
```

Résultat attendu : Monitoring temps réel actif pendant toute la durée du rollover KSK. Alertes automatiques si : perte de validation, incohérence DS entre resolvers, SERVFAIL détectés.

Remédiation : Intégrer le monitoring de rollover dans le runbook. Définir les critères de rollback (si BOGUS détecté sur > 1 resolver pendant > 5 minutes : rollback immédiat).

Évaluation : ✅ ❌ ⚠️ N/A

S07-C009 — Documentation du contact registrar d'urgence pour mise à jour DS

Criticité : 🟠 Élevé
Sources : RFC 6781 §4.1.3

Résultat attendu : Pour chaque domaine critique : coordonnées du support registrar disponible 24/7, procédure de contact préférentielle (API/hotline/email prioritaire), délai SLA documenté pour la mise à jour DS d'urgence.

Remédiation : Rédiger une fiche "contact urgence registrar" pour chaque registrar utilisé. Tester le canal de communication au moins une fois par an.

Évaluation : ✅ ❌ ⚠️ N/A

S07-C010 — Vérification de l'ancre de confiance RFC 5011 si applicable

Criticité : 🟡 Moyen
Sources : RFC 5011 (Trust Anchor Rollover)

```
# Si le domaine est utilisé comme ancre de confiance explicite dans des resolvers
# (ex: pour des zones privées avec ancre locale configurée)
# Vérifier le support RFC 5011 dans les resolvers internes
# BIND : managed-keys { } dans named.conf
grep "managed-keys\|trust-anchors" /etc/named.conf /etc/bind/named.conf* 2>/dev/null
# Unbound :
grep "auto-trust-anchor-file" /etc/unbound/unbound.conf
```

Résultat attendu : Si des ancres de confiance locales sont configurées pour des zones privées, le mécanisme RFC 5011 (mise à jour automatique des ancres de confiance via les signaux de rollover DNSKEY) est activé et testé.

Remédiation : Configurer `auto-trust-anchor-file` dans Unbound ou `managed-keys` dans BIND pour les zones privées avec ancre locale. Tester la mise à jour automatique en staging.

Évaluation : ✅ ❌ ⚠️ N/A

S07-C011 — Revue post-rollover KSK : rapport et leçons apprises

Criticité : 🟢 Faible
Sources : RFC 6781 §7, NIST SP 800-81r2

Résultat attendu : Après chaque rollover KSK : rapport de post-mortem documenté (durée effective, anomalies rencontrées, temps de propagation réel, recommandations), mis à jour dans le DPS.

Remédiation : Créer un template de rapport post-rollover. Archiver tous les rapports dans le DPS pour traçabilité et amélioration continue.

Évaluation : ✅ ❌ ⚠️ N/A

S07-C012 — Calendrier de rollover KSK planifié et approuvé par la direction

Criticité : 🟡 Moyen
Sources : RFC 7583, ISO 27001 A.10.1.2

Résultat attendu : Le prochain rollover KSK est planifié dans le calendrier de l'équipe DNS, approuvé par le RSSI, avec une communication préalable aux équipes impactées. Le rollover est documenté comme une opération de maintenance planifiée.

Remédiation : Ajouter le rollover KSK au calendrier CMDB/ITSM de l'organisation (Change Request approuvée). Planifier le rollover pendant une période de faible activité DNS.

Évaluation : ✅ ❌ ⚠️ N/A

S08 — Rollover ZSK

Contexte

Le rollover ZSK (Zone Signing Key) est beaucoup plus simple que le rollover KSK car il ne nécessite pas d'interaction avec le registrar parent (la ZSK n'est pas référencée dans le DS). Deux méthodes principales existent : le Pre-publish rollover (publier la nouvelle ZSK avant de l'activer) et le Double-signature rollover (signer avec les deux ZSK simultanément). Le Pre-publish est recommandé pour sa légèreté (une seule signature active à la fois), mais le Double-signature est plus simple à implémenter.

La fréquence de rollover ZSK recommandée est mensuelle ou trimestrielle. Trop fréquent = charge opérationnelle et réseau excessive. Trop rare = surface d'exposition en cas de compromission ZSK. La plupart des serveurs DNS modernes (BIND, Knot, PowerDNS avec KASP) automatisent complètement le rollover ZSK.

Contrôles S08

S08-C001 — Vérification de l'automatisation du rollover ZSK

Criticité : 🟠 Élevé
Sources : RFC 6781 §4.2, RFC 7583 §4

```
# BIND : vérifier la politique de signing automatique
grep -A 20 "dnssec-policy" /etc/named.conf | grep -E "zsk|zone-signing-key|lifetime|rollover"
# Ou : mode auto-dnssec maintenu
grep "auto-dnssec" /etc/named.conf
# PowerDNS : vérifier l'activation KASP
grep -i "kasp\|zone-signing" /etc/powerdns/*.conf
# Knot DNS :
grep -A 10 "policy:" /etc/knot/knot.conf | grep -E "ksk|zsk|lifetime"
```

Résultat attendu : Rollover ZSK automatisé via KASP (Key And Signature Policy) ou équivalent. Aucune intervention manuelle requise pour les rollovers ZSK en conditions normales.

Effet de bord : Le rollover ZSK manuel est source d'erreurs humaines : oubli, mauvais timing, mauvais fichier de clé utilisé. Le rollover automatique est fortement recommandé pour toutes les zones avec plusieurs ZSK actives.

Remédiation : Activer KASP dans le serveur DNS. Pour BIND 9.16+ : configurer `dnssec-policy`. Pour Knot DNS : configurer la section `policy` dans `knot.conf`. Tester l'automatisation en staging.

Évaluation : ✅ ❌ ⚠️ N/A

S08-C002 — Fréquence de rollover ZSK conforme (mensuelle ou trimestrielle)

Criticité : 🟡 Moyen
Sources : RFC 6781 §3.1.1, NIST SP 800-81r2 §4.3

```
# Vérifier la date de la dernière ZSK et la durée de vie configurée
# BIND KASP : durée de vie ZSK
grep -A 5 "zone-signing-key" /etc/named.conf | grep "lifetime"
# Vérifier la date d'activation de la ZSK courante :
for f in /var/cache/bind/K<domaine>.+013+*.private; do
  echo "$f :"
  grep -E "Activate:|Inactive:|Delete:" $f
done
```

Résultat attendu : Durée de vie ZSK configurée entre 30 et 90 jours. Rollover automatique déclenché avant la date d'inactivation (Inactive:) avec une marge de sécurité (pre-publish de TTL_DNSKEY avant activation).

Effet de bord : ZSK trop courte (< 14 jours) : charge de signing importante, risque de race condition pendant les transitions. ZSK trop longue (> 6 mois) : fenêtre de compromission trop grande, anomalies non détectées plus longtemps.

Remédiation : Configurer la durée de vie ZSK dans la politique KASP. Pour les zones à faible charge : 90 jours (trimestriel). Pour les zones à forte charge ou haute sécurité : 30 jours (mensuel).

Évaluation : ✅ ❌ ⚠️ N/A

S08-C003 — Vérification de la méthode pre-publish ZSK

Criticité : 🟡 Moyen
Sources : RFC 6781 §4.2.1

```
# Pendant un rollover pre-publish ZSK :
# Phase 1 : publication de la nouvelle ZSK (avant activation)
# Les deux ZSK doivent être visibles avec un flag 256
dig <domaine> DNSKEY +short | awk '{if($1==256) print "ZSK:", $4}'
# Pendant la transition : 2 ZSK (flags 256) sont visibles
# Phase 2 : l'ancienne ZSK signe encore, la nouvelle est juste publiée
# Phase 3 : activation de la nouvelle ZSK (commence à signer)
# Phase 4 : retrait de l'ancienne ZSK (après TTL_DNSKEY)
```

Résultat attendu : Pendant le rollover pre-publish : 2 DNSKEY flags 256 (ZSK) visibles dans la zone. La transition est transparente pour les resolvers (pas de SERVFAIL).

Effet de bord : Si la méthode double-signature est utilisée à la place de pre-publish : taille des réponses DNSSEC augmentée (deux RRSIG pour chaque RR signé), risque de fragmentation UDP sur les zones volumineuses.

Remédiation : Préférer pre-publish si le serveur DNS le supporte nativement (BIND KASP, Knot DNS). Double-signature acceptable pour les zones petites ou statiques.

Évaluation : ✅ ❌ ⚠️ N/A

S08-C004 — Surveillance automatique de la ZSK expirante

Criticité : 🔴 Critique
Sources : RFC 6781 §6.1

```
# Script de surveillance d'expiration ZSK (à intégrer dans monitoring)
for f in /var/cache/bind/K<domaine>.+*.private; do
  inactive_date=$(grep "^Inactive:" $f 2>/dev/null | awk '{print $2}')
  if [ -n "$inactive_date" ]; then
    inactive_epoch=$(date -d "$inactive_date" +%s 2>/dev/null)
    now_epoch=$(date +%s)
    days_left=$(( (inactive_epoch - now_epoch) / 86400 ))
    echo "ZSK $f expire dans $days_left jours"
    if [ $days_left -le 14 ]; then
      echo "⚠️ ALERTE : ZSK expire dans moins de 14 jours !"
    fi
  fi
done
```

Résultat attendu : Monitoring automatique actif, alertes déclenchées à J-14 (avertissement), J-7 (alerte), J-1 (critique). Intégré dans le système de monitoring centralisé.

Effet de bord : Une ZSK expirée sans nouvelle ZSK en place = zone bogué. Le délai entre la détection et la correction peut être de plusieurs heures si le monitoring n'est pas en place.

Remédiation : Intégrer le script dans cron + monitoring. Alertes par email et PagerDuty/Opsgenie pour les seuils J-7 et J-1.

Évaluation : ✅ ❌ ⚠️ N/A

S08-C005 — Vérification que l'ancienne ZSK n'est retirée qu'après expiration de toutes ses RRSIG

Criticité : 🔴 Critique
Sources : RFC 6781 §4.2.2, RFC 7583 §4.2

```
# L'ancienne ZSK peut être retirée seulement APRÈS :
# max(TTL_RRSIG) depuis le dernier moment où cette ZSK a signé
# En pratique : attendre que toutes les RRSIG créées avec l'ancienne ZSK aient expiré

# Vérifier si des RRSIG actives référencent l'ancienne ZSK (par keytag)
OLD_KEYTAG="<ancien-keytag>"
dig <domaine> AXFR @<ns-auth> 2>/dev/null | grep "RRSIG" | awk -v kt=$OLD_KEYTAG '{if($7==kt) print "RRSIG active
avec ancienne ZSK :", $0}'
```

Résultat attendu : Aucune RRSIG active dans la zone ne fait référence à l'ancienne ZSK (keytag) avant son retrait. Le retrait intervient après expiration de toutes les RRSIG signées par cette ZSK + TTL_max.

Effet de bord : Retirer une ZSK alors que des RRSIG créées avec cette clé sont encore en cache sur des resolvers = ces resolvers obtiennent des RRSIG qu'ils ne peuvent plus vérifier (clé absente) → SERVFAIL pour ces clients.

Remédiation : BIND KASP gère automatiquement cette contrainte via les dates Inactive: et Delete: des clés. En gestion manuelle : calculer la date de suppression = date_inactivation + durée_validité_RRSIG + TTL_DNSKEY.

Évaluation : ✅ ❌ ⚠️ N/A

S08-C006 — Test de rollover ZSK automatique en staging

Criticité : 🟡 Moyen
Sources : RFC 6781 §7

```
# Test rollover ZSK automatique en staging
# 1. Réduire les TTL et durées de vie pour le test (ex: ZSK lifetime = 1 heure)
# 2. Observer le rollover automatique déclenché par KASP
# 3. Vérifier la validation DNSSEC tout au long du rollover
# 4. Vérifier les logs du serveur DNS
journalctl -u named --since "1 hour ago" | grep -i "signing\|key\|rollover\|zsk"
```

Résultat attendu : Rollover ZSK automatique complet sans intervention manuelle, sans SERVFAIL pendant la transition, logs cohérents avec la procédure pre-publish.

Remédiation : Si le rollover automatique échoue en staging, diagnostiquer via les logs et corriger la configuration KASP avant déploiement en production.

Évaluation : ✅ ❌ ⚠️ N/A

S08-C007 — Documentation de la procédure de rollover ZSK manuel d'urgence

Criticité : 🟠 Élevé
Sources : RFC 6781 §4.2, RFC 7583

```
# Procédure rollover ZSK manuelle d'urgence (si KASP défaillant) :
# 1. Générer nouvelle ZSK
dnssec-keygen -a ECDSAP256SHA256 <domaine>
# 2. Publier la nouvelle ZSK dans la zone (pre-publish)
dnssec-settime -P now -A +3600 K<domaine>.+013+<newtag>.key
rndc loadkeys <domaine>
# 3. Attendre TTL_DNSKEY (1h typiquement)
# 4. Activer la nouvelle ZSK (re-signe avec la nouvelle)
dnssec-settime -A now K<domaine>.+013+<newtag>.key
rndc sign <domaine>
# 5. Désactiver l'ancienne ZSK
dnssec-settime -I now K<domaine>.+013+<oldtag>.key
# 6. Après TTL_RRSIG : supprimer l'ancienne ZSK
dnssec-settime -D +<ttl_rrsig_seconds> K<domaine>.+013+<oldtag>.key
```

Résultat attendu : Procédure manuelle documentée et testée, utilisable en urgence si l'automatisation KASP est défaillante. RTO ≤ 2h depuis détection de compromission ZSK.

Remédiation : Inclure cette procédure dans le runbook DNSSEC et le DPS.

Évaluation : ✅ ❌ ⚠️ N/A

S08-C008 — Vérification que le rollover ZSK n'augmente pas la taille des réponses au-delà du seuil

Criticité : 🟡 Moyen
Sources : RFC 6840 §5.3, RFC 8906

```
# Pendant la phase de double-signature (si méthode double-sig utilisée)
# Mesurer la taille des réponses
dig <domaine> A +dnssec +bufsize=4096 | grep "MSG SIZE"
# Vérifier qu'on reste sous 1232 bytes (limite IPv6 MTU 1280)
# Si proche de la limite : envisager migration vers Ed25519 (signatures plus petites)
```

Résultat attendu : Taille des réponses DNSSEC < 1232 bytes même pendant la phase de transition ZSK. Sinon, activer le TCP fallback et s'assurer que le port TCP/53 est ouvert sur les pare-feux.

Remédiation : Migrer vers l'algorithme Ed25519 (algo 15) pour réduire la taille des signatures de 256 bytes (RSA-2048) à 64 bytes, permettant plus de flexibilité lors des doubles signatures.

Évaluation : ✅ ❌ ⚠️ N/A

S08-C009 — Rapport sur la fréquence réelle des rollovers ZSK (audit)

Criticité : 🟢 Faible
Sources : RFC 6781 §6, NIST SP 800-81r2

```
# Historique des rollovers ZSK (via les logs DNS)
grep -i "new key\|key rollover\|znssec-keygen\|ZSK" /var/log/named/named.log | \
  grep -E "20[2-9][0-9]" | tail -30
# Ou via les fichiers de clés archivés
ls -la /var/cache/bind/K<domaine>*.key | sort -k6,7
```

Résultat attendu : Les rollovers ZSK se sont produits à la fréquence prévue (mensuelle/trimestrielle). Aucun gap non expliqué. Journal des rollovers archivé dans le DPS.

Remédiation : Créer un tableau de bord des rollovers (date, keytag, méthode, durée, anomalies). Intégrer dans le rapport mensuel DNSSEC.

Évaluation : ✅ ❌ ⚠️ N/A

S08-C010 — Vérification de la cohérence ZSK lors d'un rollover en architecture multi-NS

Criticité : 🟠 Élevé
Sources : RFC 6781 §4.2.3, RFC 8901

```
# En architecture multi-NS : vérifier que tous les NS ont la même ZSK active
NS_LIST=$(dig <domaine> NS +short)
for ns in $NS_LIST; do
  echo "=== ZSK sur $ns ==="
  dig @$ns <domaine> DNSKEY +short | awk '{if($1==256) print "ZSK:", $4}'
done
```

Résultat attendu : Tous les NS autoritatifs montrent la même ZSK active. En architecture multi-signer (RFC 8901), chaque NS peut avoir sa propre ZSK, mais toutes les ZSK doivent être publiées dans toutes les réponses DNSKEY.

Remédiation : Vérifier la synchronisation AXFR/IXFR entre NS primaire et secondaires. Si multi-signer : vérifier que le processus d'échange de ZSK entre signataires est correctement automatisé.

Évaluation : ✅ ❌ ⚠️ N/A

S09 — CDS / CDNSKEY (RFC 7344 / RFC 8078)

Contexte

CDS (Child DS) et CDNSKEY (Child DNSKEY) sont des enregistrements DNS publiés dans la zone fille qui signalent au parent les changements de clés souhaités, permettant l'automatisation des rollovers KSK sans intervention manuelle côté registrar. La RFC 7344 définit le mécanisme, la RFC 8078 complète avec la gestion de l'activation/désactivation initiale de DNSSEC.

Un CDS/CDNSKEY publié signifie : "voici le DS/DNSKEY que je souhaite que mon parent publie". Le parent (registrar) interroge périodiquement la zone pour détecter les changements CDS et met à jour le DS en conséquence. CDS avec valeur 0 signifie "supprimer le DNSSEC" (délégation non sécurisée souhaitée).

Contrôles S09

S09-C001 — Vérification du support CDS/CDNSKEY chez le registrar

Criticité : 🟡 Moyen
Sources : RFC 7344, RFC 8078, IANA registry

```
# Vérifier si CDS/CDNSKEY sont publiés dans la zone
dig <domaine> CDS +short
dig <domaine> CDNSKEY +short
# Vérifier le support registrar via la liste IANA :
# https://www.iana.org/assignments/cds-cdnskey-registrars/
# Test pratique : publier un CDS et vérifier si le DS est mis à jour
# Consulter la documentation du registrar
```

Résultat attendu : Si le registrar supporte CDS/CDNSKEY : CDS publié lors des rollovers KSK, workflow documenté et testé. Si non supporté : rollover KSK manuel documenté avec contact registrar.

Effet de bord : Un CDS laissé publié en permanence (et non uniquement pendant les rollovers) peut provoquer des mises à jour DS répétitives côté registrar si leur implémentation est aggressive. Certains registrars limitent la fréquence de polling à 24h.

Remédiation : Configurer la publication CDS uniquement pendant les phases de rollover KSK (BIND KASP gère cela automatiquement via les états de politique).

Évaluation : ✅ ❌ ⚠️ N/A

S09-C002 — Format et cohérence du CDS publié

Criticité : 🟡 Moyen
Sources : RFC 7344 §4

```
# Vérifier le contenu du CDS (doit correspondre au nouveau DS voulu)
dig <domaine> CDS +short
# Format : <keytag> <algo> <digest-type> <digest>
# Cross-check : le CDS doit correspondre à la DNSKEY KSK active ou nouvelle
# Calculer le DS attendu :
dig <domaine> DNSKEY +short | awk '{if($1==257) print}' | dnssec-dsfromkey -f /dev/stdin <domaine>
```

Résultat attendu : Le CDS correspond exactement au DS qui devrait être publié chez le parent (keytag, algorithme, digest type et digest cohérents avec la DNSKEY KSK cible).

Effet de bord : Un CDS incorrect (mauvais keytag, mauvais digest) fera soumettre un DS erroné au parent → zone bogué dès que l'ancien DS est retiré par le parent.

Remédiation : Vérifier le CDS avant publication. Comparer avec la sortie de `dnssec-dsfromkey` appliqué à la DNSKEY KSK cible.

Évaluation : ✅ ❌ ⚠️ N/A

S09-C003 — Vérification que le CDS est signé par la KSK active (RFC 7344 §4.1)

Criticité : 🟠 Élevé
Sources : RFC 7344 §4.1

```
# Le CDS doit être signé par la KSK correspondant au DS actuellement en vigueur chez le parent
# (pas uniquement par la nouvelle KSK en cours de déploiement)
dig <domaine> RRSIG CDS +short
# Vérifier le keytag de la signature CDS
dig <domaine> RRSIG CDS +short | awk '{print "CDS signé par keytag:", $7}'
# Ce keytag doit correspondre à la KSK dont le DS est actuellement chez le parent
dig <domaine> DS +short | awk '{print "DS actuel keytag:", $1}'
```

Résultat attendu : Le CDS est signé par la KSK actuelle (dont le DS est chez le parent), garantissant l'authenticité de la demande de changement. La signature par la nouvelle KSK seule ne suffit pas (le parent ne peut pas encore la vérifier).

Effet de bord : Un CDS signé uniquement par la nouvelle KSK (dont le DS n'est pas encore chez le parent) ne peut pas être vérifié par le parent → ignoré → rollover bloqué.

Remédiation : Pendant le rollover KSK en double-signature, signer le CDS avec les deux KSK (ancienne + nouvelle) pour garantir la vérifiabilité par le parent à toutes les étapes.

Évaluation : ✅ ❌ ⚠️ N/A

S09-C004 — Surveillance de la consommation du CDS par le parent

Criticité : 🟡 Moyen
Sources : RFC 7344 §4.2

```
# Surveiller si le parent a bien consommé le CDS (mis à jour le DS)
# Avant soumission CDS :
dig <domaine> DS +short @8.8.8.8  # DS avant changement
# Après publication CDS + délai polling :
dig <domaine> DS +short @8.8.8.8  # DS après mise à jour attendue
# Si pas de changement après 48h = contacter le registrar
```

Résultat attendu : Le DS chez le parent est mis à jour dans les 24-48h après publication du CDS (délai dépend de la fréquence de polling du registrar).

Remédiation : Si le registrar ne consomme pas le CDS après 48h : vérifier que le CDS est correctement signé et que le format est valide. Contacter le support registrar.

Évaluation : ✅ ❌ ⚠️ N/A

S09-C005 — Retrait du CDS après confirmation de mise à jour DS

Criticité : 🟡 Moyen
Sources : RFC 7344 §4.2

```
# Vérifier si CDS est toujours publié alors que DS est déjà mis à jour
CDS=$(dig <domaine> CDS +short)
DS=$(dig <domaine> DS +short)
if [ -n "$CDS" ] && [ -n "$DS" ]; then
  echo "⚠️ CDS toujours publié — vérifier si DS est déjà à jour"
  echo "CDS: $CDS"
  echo "DS: $DS"
fi
```

Résultat attendu : Le CDS est retiré de la zone après confirmation que le DS chez le parent correspond au CDS. Un CDS permanent (non retiré après rollover) peut causer de la confusion lors du prochain rollover.

Remédiation : Intégrer le retrait du CDS dans le runbook de rollover KSK, comme étape explicite après confirmation DS propagé.

Évaluation : ✅ ❌ ⚠️ N/A

S09-C006 — Procédure CDS "delete" pour désactivation propre de DNSSEC

Criticité : 🟡 Moyen
Sources : RFC 8078 §4

```
# Pour désactiver DNSSEC proprement : publier un CDS avec valeur "0 0 0 00"
# (RFC 8078 §4 : CDS "delete")
# JAMAIS simplement supprimer le DS sans désactiver DNSSEC proprement
# Procédure de désactivation DNSSEC :
# 1. Publier CDS "0 0 0 00" (signale la désactivation)
# 2. Attendre que le parent retire le DS
# 3. Retirer les DNSKEY et RRSIG de la zone
# 4. Retirer le NSEC3PARAM
dig <domaine> CDS +short  # "0 0 0 00" si désactivation en cours
```

Résultat attendu : Si une désactivation DNSSEC est prévue : procédure propre via CDS delete documentée et testée, jamais de suppression brutale du DS sans CDS delete préalable.

Effet de bord : Supprimer le DS directement sans CDS delete préalable = zone bogué jusqu'à expiration des caches. Les resolvers qui ont le DS en cache continuent de tenter la validation → SERVFAIL pendant TTL_DS.

Évaluation : ✅ ❌ ⚠️ N/A

S09-C007 — Test du workflow complet CDS en environnement de staging

Criticité : 🟡 Moyen
Sources : RFC 7344, RFC 8078

```
# Test en staging :
# 1. Configurer un resolver de test avec ancre de confiance locale
# 2. Simuler un rollover KSK via CDS
# 3. Vérifier que la validation reste active pendant tout le rollover
# 4. Tester le CDS delete
```

Résultat attendu : Workflow CDS complet testé (publication → polling → DS update → retrait CDS) sans interruption de validation. Temps de propagation réel documenté.

Remédiation : Inclure ce test dans les exercices annuels DNSSEC. Adapter le runbook si des problèmes sont détectés en staging.

Évaluation : ✅ ❌ ⚠️ N/A

S09-C008 — Documentation du workflow CDS dans le DPS

Criticité : 🟢 Faible
Sources : RFC 7344, RFC 6781

Résultat attendu : Le DPS documente : utilisation ou non de CDS/CDNSKEY, registrars supportant ce mécanisme, fréquence de polling connue, procédure de vérification post-update, critères de fallback vers le rollover manuel si CDS échoue.

Remédiation : Mettre à jour le DPS pour inclure une section dédiée CDS/CDNSKEY.

Évaluation : ✅ ❌ ⚠️ N/A

S10 — Multi-Signer DNSSEC (RFC 8901)

Contexte

Le Multi-Signer DNSSEC (RFC 8901, 2020) permet à plusieurs fournisseurs DNS autoritatifs de signer indépendamment la même zone, chacun avec sa propre ZSK, tout en maintenant une zone DNSSEC valide.

Cette architecture est typique des déploiements anycast multi-fournisseurs (ex: Cloudflare + NS1 + Dyn), des migrations DNS sans coupure, et des architectures de haute disponibilité géodistribuée.

Le principe : chaque fournisseur maintient sa propre ZSK. Toutes les ZSK sont publiées dans la zone (par tous les fournisseurs). Chaque fournisseur signe uniquement avec sa propre ZSK, mais tous les clients peuvent vérifier toutes les signatures car ils ont accès à toutes les ZSK.

Contrôles S10

S10-C001 — Vérification de la présence de toutes les ZSK dans la zone

Criticité : 🔴 Critique (si multi-signer actif)
Sources : RFC 8901 §2.1

```
# En architecture multi-signer :
# Chaque NS devrait avoir les mêmes DNSKEY (toutes les ZSK de tous les signataires)
NS_LIST=$(dig <domaine> NS +short)
for ns in $NS_LIST; do
  echo "=== ZSK publiées par $ns ==="
  dig @$ns <domaine> DNSKEY +short | awk '{if($1==256) print "ZSK:", $4}' | sort
done
# Toutes les lignes doivent être identiques
```

Résultat attendu : Tous les NS publient l'ensemble des ZSK de tous les co-signataires. Aucune ZSK manquante sur un NS.

Effet de bord : Si un NS ne publie pas toutes les ZSK : les resolvers qui interrogent ce NS obtiennent des RRSIG signées avec une ZSK non publiée dans les DNSKEY → SERVFAIL.

Remédiation : Mettre en place un mécanisme de synchronisation des ZSK entre signataires (API propriétaires des fournisseurs, ou protocole automatisé RFC 8901 §2.2).

Évaluation : ✅ ❌ ⚠️ N/A

S10-C002 — Vérification que chaque NS signe uniquement avec sa propre ZSK

Criticité : 🟠 Élevé
Sources : RFC 8901 §2.1.1

```
# Vérifier la cohérence des RRSIG par NS
NS_LIST=$(dig <domaine> NS +short)
for ns in $NS_LIST; do
  echo "=== RRSIG A depuis $ns ==="
  dig @$ns <domaine> RRSIG A +short | awk '{print "Keytag signataire:", $7}'
done
# En multi-signer : chaque NS doit signer avec sa propre ZSK (keytags différents selon NS)
```

Résultat attendu : Chaque NS répond avec des RRSIG signées par sa propre ZSK (keytag différent entre fournisseurs). Les clients peuvent vérifier toutes les RRSIG car toutes les ZSK sont publiées dans les DNSKEY.

Effet de bord : Si un NS signe avec la ZSK d'un autre fournisseur (mauvaise configuration), cela casse le modèle de sécurité multi-signer et peut provoquer des SERVFAIL si la ZSK utilisée n'est pas celle attendue pour ce NS.

Évaluation : ✅ ❌ ⚠️ N/A

S10-C003 — Coordination du rollover ZSK en architecture multi-signer

Criticité : 🟠 Élevé
Sources : RFC 8901 §2.2, draft-ietf-dnsop-dnssec-automation

```
# Procédure de rollover ZSK en multi-signer :
# Phase 1 (Fournisseur A rollover) :
#   A publie sa nouvelle ZSK_A2
#   A informe B : "ajouter ZSK_A2 dans vos DNSKEY"
#   B ajoute ZSK_A2 dans ses DNSKEY (sans l'utiliser pour signer)
# Phase 2 : A active ZSK_A2 (commence à signer avec ZSK_A2)
# Phase 3 : A et B retirent ZSK_A1 après TTL_DNSKEY
# Communication nécessaire entre fournisseurs à chaque étape
```

Résultat attendu : Procédure de coordination rollover ZSK documentée entre tous les co-signataires, avec canaux de communication définis (API, email sécurisé, portail partagé). Délais de réponse SLA définis entre signataires.

Effet de bord : Un rollover ZSK non coordonné en multi-signer est la cause la plus fréquente d'incident DNSSEC dans ces architectures. Si fournisseur A retire son ancienne ZSK sans attendre que B l'ait également retirée de ses DNSKEY, des incohérences apparaissent.

Remédiation : Établir un processus de change management entre fournisseurs pour les opérations DNSSEC. Envisager l'automatisation via draft-ietf-dnsop-dnssec-automation.

Évaluation : ✅ ❌ ⚠️ N/A

S10-C004 — Choix du modèle KSK en architecture multi-signer

Criticité : 🟡 Moyen
Sources : RFC 8901 §2.1.2

```
# Modèle 1 : KSK unique partagée entre tous les signataires
# Avantage : un seul DS chez le parent
# Inconvénient : la clé privée KSK doit être partagée entre fournisseurs (risque)
# Modèle 2 : KSK séparée par fournisseur (plusieurs DS chez le parent)
# Avantage : clés privées isolées par fournisseur
# Inconvénient : multiple DS chez le parent, coordinations rollover KSK complexes
dig <domaine> DS +short  # Un seul DS = modèle 1, plusieurs DS = modèle 2
dig <domaine> DNSKEY +short | awk '{if($1==257) print "KSK:", $4}' | wc -l
```

Résultat attendu : Choix de modèle documenté dans le DPS, avec justification de sécurité. Le modèle 2 (KSK séparées) est préféré pour la sécurité mais exige plus de DS chez le parent.

Remédiation : Documenter le modèle choisi dans le DPS. Si passage d'un modèle à l'autre est envisagé, planifier une opération de rollover complexe.

Évaluation : ✅ ❌ ⚠️ N/A

S10-C005 — Tests de validation depuis des resolvers géodistribués en multi-signer

Criticité : 🟠 Élevé
Sources : RFC 8901, RFC 4035

```
# Tests depuis plusieurs points géographiques
# Utiliser RIPE Atlas ou Catchpoint
for resolver in 9.9.9.9 1.1.1.1 8.8.8.8 64.6.64.6; do
  result=$(dig @$resolver <domaine> A +dnssec 2>/dev/null | grep -E "SERVFAIL|NOERROR")
  echo "Resolver $resolver: $result"
done
```

Résultat attendu : NOERROR avec bit AD depuis tous les resolvers de test, indépendamment du NS anycast interrogé. Aucun SERVFAIL géographique.

Remédiation : En cas de SERVFAIL géographique : identifier le fournisseur DNS responsable et la ZSK manquante. Synchroniser les DNSKEY entre fournisseurs.

Évaluation : ✅ ❌ ⚠️ N/A

S10-C006 — Documentation des accords SLA entre co-signataires

Criticité : 🟡 Moyen
Sources : RFC 8901 §2

Résultat attendu : Accords contractuels ou SLA documentés entre les fournisseurs co-signataires couvrant : délai de réponse pour les opérations ZSK, procédure en cas de SERVFAIL multi-signer, contacts d'urgence, responsabilités lors d'un incident DNSSEC.

Remédiation : Rédiger un accord de collaboration DNSSEC entre fournisseurs. Inclure les SLA dans les contrats de service.

Évaluation : ✅ ❌ ⚠️ N/A

S10-C007 — Vérification de la disponibilité du draft dnssec-automation pour automatisation

Criticité : 🟢 Faible
Sources : draft-ietf-dnsop-dnssec-automation

```
# Le draft propose un protocole standardisé pour automatiser la
# coordination DNSSEC en multi-signer via des signaux DNS
# Vérifier si les fournisseurs supportent ce draft
# (fonctionnalité émergente en 2026, implémentation partielle)
```

Résultat attendu : Suivi du draft IETF `draft-ietf-dnsop-dnssec-automation`. Plan d'adoption documenté dès standardisation finale. En attendant, coordination manuelle via API propriétaires des fournisseurs.

Évaluation : ✅ ❌ ⚠️ N/A

S10-C008 — Test de migration DNS en multi-signer sans DNSSEC outage

Criticité : 🟡 Moyen
Sources : RFC 8901 §3 (Migration use case)

```
# Scénario de test : migration d'un fournisseur DNS vers un autre
# avec maintien de DNSSEC continu (pas de zone bogué pendant la migration)
# Phase 1 : ajouter le nouveau fournisseur comme co-signataire
# Phase 2 : vérifier la validation depuis tous les resolvers
# Phase 3 : retirer l'ancien fournisseur
# Phase 4 : finaliser la configuration
```

Résultat attendu : Migration DNS avec zéro downtime DNSSEC testée en staging. Procédure documentée pour les migrations futures.

Remédiation : Utiliser l'architecture multi-signer RFC 8901 comme base pour toutes les futures migrations DNS, évitant les zones bogué pendant les transitions de fournisseurs.

Évaluation : ✅ ❌ ⚠️ N/A

S11 — Résolveur : Validation DNSSEC

Contexte

Un resolver validant DNSSEC est un resolver récursif qui vérifie les signatures DNSSEC des réponses DNS avant de les transmettre aux clients. Sans resolver validant, toute la chaîne de signatures DNSSEC n'a aucun

effet de sécurité pour les clients internes : les réponses signées mais non vérifiées n'apportent aucune protection contre le spoofing DNS.

La configuration minimale requise est `dnssec-validation auto` pour BIND (qui charge automatiquement l'ancre de confiance racine IANA) ou `val-override-date` pour Unbound. Les resolvers validants doivent être testés régulièrement via `dnssec-failed.org` (zone intentionnellement bogué pour tester la validation).

Contrôles S11

S11-C001 — Activation de la validation DNSSEC sur tous les resolvers récursifs internes

Criticité : 🔴 Critique
Sources : RFC 4033-4035, RFC 6840, ANSSI REC-DNSSEC-2021 §6

```
# Test : le resolver interne doit SERVFAIL sur une zone bogué
dig @<resolver-interne> dnssec-failed.org A
# Attendu : SERVFAIL (validation active)
# Si NOERROR : validation non activée ❌

# Vérifier la configuration BIND
grep "dnssec-validation" /etc/named.conf
# Attendu : dnssec-validation auto; ou dnssec-validation yes;

# Vérifier la configuration Unbound
grep -E "val|dnssec" /etc/unbound/unbound.conf | grep -v "#"
# Attendu : auto-trust-anchor-file: "/etc/unbound/root.key"
```

Résultat attendu : Tous les resolvers récursifs internes (BIND, Unbound, Windows DNS Server) ont la validation DNSSEC activée. Test dnssec-failed.org retourne SERVFAIL.

Effet de bord : L'activation de la validation DNSSEC peut casser la résolution de zones bogués actuellement "silencieusement" acceptées. Un audit préalable des zones internes et externes utilisées est recommandé avant activation en production.

Remédiation : BIND : ajouter `dnssec-validation auto;` dans `options {}`. Unbound : ajouter `auto-trust-anchor-file: "/etc/unbound/root.key"` et exécuter `unbound-anchor` pour initialiser l'ancre de confiance. Redémarrer le service après modification.

Évaluation : ✅ ❌ ⚠️ N/A

S11-C002 — Configuration et mise à jour de l'ancre de confiance racine

Criticité : 🔴 Critique
Sources : RFC 5011, RFC 7958, IANA Root DNSKEY

```
# Vérifier l'ancre de confiance racine (root KSK)
# BIND : managed-keys ou trust-anchors
grep -A 5 "managed-keys\|trust-anchors" /etc/named.conf | head -10
# L'ancre de confiance root doit correspondre au keytag IANA actuel
# KSK root actuelle (2026) : keytag 20326
dig . DNSKEY +short | awk '{if($1==257) print "KSK root keytag disponible"}'

# Unbound : vérifier root.key
cat /etc/unbound/root.key | grep "DNSKEY\|KSK"
```

Résultat attendu : Ancre de confiance racine configurée et à jour (keytag 20326 en 2026). Mise à jour automatique via RFC 5011 configurée (BIND `auto-dnssec auto` ou Unbound `auto-trust-anchor-file`).

Effet de bord : Ancre de confiance obsolète = toute la résolution DNS validante retourne BOGUS/SERVFAIL pour l'ensemble des zones signées. L'incident ICANN root KSK rollover 2018 a affecté ~15% des resolvers avec ancre de confiance non mise à jour.

Remédiation : S'assurer que la mise à jour automatique RFC 5011 est activée. Pour les systèmes sans accès Internet pour le resolver, mettre à jour manuellement l'ancre de confiance depuis le site IANA (fichier root-anchors.xml).

Évaluation : ✅ ❌ ⚠️ N/A

S11-C003 — Test de la présence du bit AD (Authenticated Data) dans les réponses

Criticité : 🟠 Élevé
Sources : RFC 4035 §3.2.3, RFC 6840 §5.7

```
# Le bit AD indique que le resolver a validé les données DNSSEC
dig @<resolver-interne> <domaine-signé> A +dnssec | grep "ad"
# Output attendu : ;; flags: qr rd ra ad; → bit 'ad' présent

# Test depuis le réseau client (pas directement depuis le resolver)
dig <domaine-signé> A +dnssec | grep "flags:"
```

Résultat attendu : Bit AD présent dans les réponses pour les zones correctement signées. Bit AD absent sur les zones non signées (comportement normal). SERVFAIL sur les zones bogués.

Effet de bord : Certains applications/clients vérifient le bit AD pour prendre des décisions de sécurité (ex: applications DANE, certaines implémentations d'email). L'absence de bit AD quand attendu peut provoquer des rejets de connexion.

Remédiation : Vérifier que les pare-feux entre resolver et clients ne filtrent pas le bit AD (certains firewalls DNS strippent les flags DNSSEC). Configurer les resolvers pour transmettre le bit AD aux clients.

Évaluation : ✅ ❌ ⚠️ N/A

S11-C004 — Comportement SERVFAIL documenté et testé pour les zones bogués

Criticité : 🟠 Élevé
Sources : RFC 4035 §5.5

```
# Tester le comportement sur plusieurs types de zones bogués
# Zone avec RRSIG expirée
dig @<resolver> dnssec-failed.org A  # Doit retourner SERVFAIL

# Zone avec DS incorrect
# (difficile à tester sans zone de test dédiée)

# Vérifier que les clients sont informés du SERVFAIL (pas de réponse vide)
dig @<resolver> dnssec-failed.org A +dnssec | grep -E "SERVFAIL|RCODE"
```

Résultat attendu : Comportement documenté : SERVFAIL retourné aux clients pour les zones bogués. Les logs du resolver indiquent la raison du SERVFAIL (DNSSEC validation failed). Un mécanisme d'override est disponible pour les cas d'urgence (mode validation relaxée temporaire).

Effet de bord : SERVFAIL pour les clients = impossibilité d'accéder à la ressource. Pour les zones internes, cela peut bloquer des applications critiques. Il est essentiel de tester TOUTES les zones internes avant d'activer la validation.

Remédiation : Créer une liste de toutes les zones DNS utilisées en interne et vérifier leur statut DNSSEC avant activation de la validation. Corriger les zones bogués existantes avant activation.

Évaluation : ✅ ❌ ⚠️ N/A

S11-C005 — Vérification de la validation sur les resolvers forwarders

Criticité : 🟠 Élevé
Sources : RFC 5625

```
# Si des forwarders sont utilisés, vérifier qu'ils valident aussi
grep "forwarders\|forward" /etc/named.conf | head -5
# Tester le resolver forwarder directement :
FORWARDER_IP=$(grep "forwarders" /etc/named.conf | awk '{print $1}')
dig @$FORWARDER_IP dnssec-failed.org A
# Doit retourner SERVFAIL
```

Résultat attendu : Si des forwarders sont utilisés, ils effectuent eux-mêmes la validation DNSSEC. Ou le resolver local effectue la validation même si les requêtes sont transférées (mode `forward only` + validation locale).

Effet de bord : Un forwarder qui n'effectue pas la validation et répond NOERROR pour des zones bogués → le resolver local en mode `forward only` transmet la réponse non validée → perte totale de protection DNSSEC.

Remédiation : Soit activer la validation sur le forwarder, soit configurer le resolver local en mode `forward first` ou `forward only` avec validation locale (`dnssec-validation yes` + ancre de confiance locale).

Évaluation : ✅ ❌ ⚠️ N/A

S11-C006 — Monitoring du ratio SERVFAIL DNSSEC (< 0.1% du trafic)

Criticité : 🟡 Moyen
Sources : RFC 6781 §6, ICANN DNS Monitoring

```
# Extraire les métriques SERVFAIL depuis les logs BIND
rndc stats && grep "queries resulted in SERVFAIL" /var/named/data/named_stats.txt
# Ou via dnstap/tcpdump :
# Pour Unbound :
unbound-control stats | grep -E "servfail|num.queries"
# Calculer le ratio SERVFAIL/total
```

Résultat attendu : Ratio SERVFAIL < 0.1% du total des requêtes DNS. Un pic de SERVFAIL > 0.5% déclenche une alerte d'investigation immédiate (possible zone bogué non détectée).

Effet de bord : Un ratio SERVFAIL élevé peut indiquer une zone bogué non détectée, un problème de validation (ancre de confiance obsolète), ou une attaque de flooding DNS. La discrimination entre ces causes nécessite une analyse des logs détaillée.

Remédiation : Configurer un dashboard Prometheus/Grafana sur les métriques BIND/Unbound. Alertes sur seuil SERVFAIL rate > 0.1%.

Évaluation : ✅ ❌ ⚠️ N/A

S11-C007 — Vérification du support EDNS0 et taille buffer pour les grandes réponses DNSSEC

Criticité : 🟡 Moyen
Sources : RFC 6891, RFC 6840 §5.3

```
# Vérifier la taille du buffer EDNS0 configurée sur le resolver
dig <domaine> A +dnssec | grep "EDNS"
# Vérifier la configuration BIND :
grep "edns-udp-size\|max-udp-size" /etc/named.conf
# Recommandé : edns-udp-size 4096; max-udp-size 4096;
# Vérifier la taille réelle des réponses DNSSEC :
dig <domaine> DNSKEY +dnssec | grep "MSG SIZE"
```

Résultat attendu : EDNS0 activé avec buffer 4096 bytes minimum. TCP fallback activé pour les réponses > buffer EDNS0. Réponses DNSSEC correctement transmises sans troncature.

Effet de bord : Buffer EDNS0 trop petit = réponses DNSSEC tronquées → TC bit → TCP retry → si TCP bloqué par firewall → SERVFAIL. Configuration critique pour les environnements avec pare-feux restrictifs.

Remédiation : S'assurer que TCP/53 est ouvert entre clients et resolvers. Configurer `edns-udp-size 4096;` dans les options BIND. Tester avec `dig +bufsize=4096 +dnssec`.

Évaluation : ✅ ❌ ⚠️ N/A

S11-C008 — Test de la validation depuis Windows DNS Server (si environnement hybride)

Criticité : 🟡 Moyen
Sources : Microsoft DNSSEC Deployment Guide

```
# Windows DNS Server : activer la validation DNSSEC via PowerShell
# Get-DnsServerDnsSecZone (pour les zones locales)
# Vérifier que le resolver Windows est configuré pour valider :
# Set-DnsClientServerAddress -InterfaceIndex <n> -ServerAddresses <resolver-validant>
# Test Windows :
Resolve-DnsName dnssec-failed.org -Server <resolver-interne>
# Doit retourner une erreur DNS si validation active
```

Résultat attendu : Les clients Windows utilisent un resolver validant DNSSEC. Windows DNS Server en mode forwarder vers un resolver BIND/Unbound validant. La validation DNSSEC est fonctionnelle depuis les postes Windows.

Effet de bord : Windows DNS Server avant Server 2012 R2 ne supporte pas ECDSA P-256 (algo 13). Pour les zones signées avec algo 13, les clients Windows anciens peuvent ne pas valider correctement.

Remédiation : Mettre à jour Windows DNS Server vers 2016+ pour le support complet DNSSEC. Configurer comme forwarder vers BIND/Unbound plutôt que comme resolver validant natif.

Évaluation : ✅ ❌ ⚠️ N/A

S11-C009 — Configuration du timeout et retry DNSSEC sur les resolvers

Criticité : 🟢 Faible
Sources : RFC 4035 §5.1

```
# Vérifier les timeouts configurés (DNSSEC peut nécessiter plus de temps)
grep -E "timeout|retry|attempts" /etc/resolv.conf
# BIND : tcp-clients et timeout
grep "tcp-clients\|query-timeout" /etc/named.conf
# Unbound :
grep "outgoing-num-tcp\|tcp-upstream" /etc/unbound/unbound.conf
```

Résultat attendu : Timeout suffisant pour les requêtes DNSSEC (la validation de la chaîne peut nécessiter 3-5 requêtes supplémentaires vs une requête non DNSSEC). Timeout recommandé : 5-10 secondes. TCP upstream activé.

Remédiation : Ajuster les timeouts dans la configuration du resolver. Activer TCP upstream dans Unbound pour les zones avec réponses volumineuses.

Évaluation : ✅ ❌ ⚠️ N/A

S11-C010 — Audit des applications qui ignorent ou contournent DNSSEC

Criticité : 🟡 Moyen
Sources : RFC 6840 §5.9, RFC 4035 §4.9

```
# Identifier les applications qui font leurs propres requêtes DNS
# et ignorent le bit AD ou effectuent leurs propres requêtes stub
# Exemples : Java (JVM avec sa propre résolution DNS), browsers (DoH)
# Vérifier si des applications bypasse le resolver interne :
ss -lnup | grep ":53"
tcpdump -i any -n port 53 -c 100 2>/dev/null | awk '{print $3}' | sort | uniq -c | sort -rn | head -10
```

Résultat attendu : Toutes les applications utilisent le resolver interne validant. Aucune application ne bypass le resolver (DoH direct, résolution DNS propre sans validation).

Effet de bord : Applications Java avec DNS TTL override, applications avec résolution DNS hardcodée, navigateurs avec DoH activé → bypass du resolver interne → perte de la protection DNSSEC.

Remédiation : Documenter les applications qui effectuent leur propre résolution DNS. Configurer des politiques de réseau pour forcer le DNS interne (blocage DoH externe, interception DNS).

Évaluation : ✅ ❌ ⚠️ N/A

S12 — DANE / TLSA

Contexte

DANE (DNS-Based Authentication of Named Entities, RFC 6698) exploite DNSSEC pour associer des certificats TLS à des enregistrements DNS, offrant une alternative ou un complément à la PKI traditionnelle. TLSA est l'enregistrement DNS qui contient l'empreinte ou le certificat du service TLS, publié sous le nom `_<port>._<protocole>.<domaine>`.

L'usage le plus mature de DANE est SMTP DANE (RFC 7672) : les serveurs de messagerie SMTP qui valident DNSSEC peuvent utiliser les enregistrements TLSA pour vérifier le certificat TLS du serveur destinataire sans dépendre des CA commerciales. Cela protège contre les attaques MITM sur le flux email entre MTA.

Les paramètres TLSA : Usage (0-3), Selector (0-1), Matching Type (0-2). Le format recommandé pour HTTPS est `3 1 1` (DANE-EE + Subject Public Key Info + SHA-256) ; pour SMTP, `2 0 1` (DANE-TA + Full Certificate + SHA-256) ou `3 1 1`.

Contrôles S12

S12-C001 — Présence d'enregistrements TLSA pour les serveurs SMTP (DANE pour email)

Criticité : 🟠 Élevé
Sources : RFC 7672, RFC 6698

```
# Trouver les serveurs MX du domaine
MX_HOSTS=$(dig <domaine> MX +short | awk '{print $2}')
# Vérifier le TLSA pour chaque MX (port 25)
for mx in $MX_HOSTS; do
  echo "=== TLSA pour $mx (SMTP) ==="
  dig _25._tcp.$mx TLSA +short
done
# Test complet SMTP DANE avec swaks :
# swaks --to test@<domaine> --tls --tls-verify --tls-protocol tlsv1_2
```

Résultat attendu : Enregistrement TLSA présent pour chaque serveur MX, format `2 0 1` ou `3 1 1`, signé DNSSEC, cohérent avec le certificat TLS actuel du MTA.

Effet de bord : Un enregistrement TLSA incorrect ou non cohérent avec le certificat TLS actuel → rejet immédiat des connexions SMTP entrantes par tous les MTA DANE-validants (Postfix avec DNSSEC, Exchange Online, Google Workspace). Impact direct sur la délivrabilité des emails.

Remédiation : Générer le hash TLSA correctement : `openssl s_client -starttls smtp -connect <mx>:25 2>/dev/null | openssl x509 -pubkey -noout | openssl pkey -pubin -outform DER | openssl dgst -sha256 -binary | xxd -p -c 256`. Publier en tant que `_25._tcp.<mx> TLSA 3 1 1 <hash>`.

Évaluation : ✅ ❌ ⚠️ N/A

S12-C002 — Présence et validité des enregistrements TLSA pour HTTPS

Criticité : 🟡 Moyen
Sources : RFC 6698 §2, RFC 7218

```
# Vérifier TLSA pour HTTPS (port 443)
dig _443._tcp.<domaine> TLSA +short
# Format recommandé : 3 1 1 <hash-sha256-spki>
# Générer le hash depuis le certificat actuel :
openssl s_client -connect <domaine>:443 2>/dev/null | \
  openssl x509 -pubkey -noout | \
  openssl pkey -pubin -outform DER | \
  openssl dgst -sha256 -binary | \
  xxd -p -c 256
# Comparer avec le hash dans le TLSA
```

Résultat attendu : TLSA `3 1 1` présent pour le service HTTPS principal, hash correspondant à la clé publique du certificat actuel. Vérifié via `check.sidnlabs.nl` ou `https://www.huque.com/bin/danecheck`.

Effet de bord : TLSA HTTPS n'est actuellement supporté que par peu de browsers (Firefox via add-on). L'impact de sécurité est limité pour HTTPS côté client, mais utile pour les services inter-applications (API mTLS, services B2B).

Remédiation : Publier le TLSA avec le hash de la clé publique (SPKI) plutôt que du certificat complet (usage 3 1 1 vs 3 0 1) pour survivre aux renouvellements de certificat Let's Encrypt qui changent le certificat mais conservent la clé.

Évaluation : ✅ ❌ ⚠️ N/A

S12-C003 — Procédure de rollover TLSA lors du renouvellement de certificat TLS

Criticité : 🔴 Critique
Sources : RFC 7671 §8.1

```
# Procédure rollover TLSA "double publication" :
# Phase 1 : Générer nouvelle clé et CSR (SANS remplacer encore le certificat)
openssl genpkey -algorithm EC -pkeyopt ec_paramgen_curve:P-256 -out new-privkey.pem
# Calculer le TLSA du nouveau certificat avant déploiement
NEW_HASH=$(openssl pkey -in new-privkey.pem -pubout | openssl pkey -pubin -outform DER | openssl dgst -sha256
-binary | xxd -p -c 256)
# Phase 2 : Publier les DEUX TLSA (ancien + nouveau) dans le DNS
# Phase 3 : Attendre TTL_TLSA propagation
# Phase 4 : Déployer le nouveau certificat
# Phase 5 : Retirer l'ancien TLSA après TTL
```

Résultat attendu : Procédure de rollover TLSA documentée et intégrée dans le processus de renouvellement de certificat TLS. Aucune interruption de service SMTP DANE pendant les transitions.

Effet de bord : Renouveler un certificat sans mettre à jour le TLSA → tous les MTA DANE-validants rejettent les connexions SMTP immédiatement. Impact critique sur la messagerie. Let's Encrypt renouvelle automatiquement toutes les 60-90 jours → le rollover TLSA doit être automatisé.

Remédiation : Automatiser la mise à jour TLSA via certbot hook (`--deploy-hook`). Script disponible : `certbot-dns-hook` ou scripts personnalisés via l'API DNS du fournisseur.

Évaluation : ✅ ❌ ⚠️ N/A

S12-C004 — Test DANE SMTP depuis des vérificateurs externes

Criticité : 🟠 Élevé
Sources : RFC 7672 §2

```
# Tester la configuration DANE SMTP depuis des outils en ligne :
# https://dane.sys4.de/ — test complet SMTP DANE
# https://mxtoolbox.com/SuperTool.aspx — vérification TLSA
# Test local avec kdig (supporte DANE) :
kdig _25._tcp.<mx-domaine> TLSA +dnssec
# Vérifier que les TLSA sont correctement signés (bit AD)
dig _25._tcp.<mx-domaine> TLSA +dnssec | grep "ad"
```

Résultat attendu : Validation réussie depuis `dane.sys4.de` et `mxtoolbox`. Bit AD présent sur les réponses TLSA. Le test de connexion SMTP DANE réussit sans erreurs de certificat.

Remédiation : Si le test échoue : identifier si le problème est dans le TLSA (hash incorrect), le DNSSEC (RRSIG TLSA expirée), ou le certificat TLS (certificat différent du TLSA).

Évaluation : ✅ ❌ ⚠️ N/A

S12-C005 — Vérification de la cohérence TLSA avec le certificat TLS déployé

Criticité : 🔴 Critique
Sources : RFC 6698 §2.1

```
# Vérifier la cohérence en temps réel
python3 << 'EOF'
import ssl, socket, hashlib, dns.resolver

hostname = "<domaine>"
port = 443

# Obtenir le certificat TLS actuel
context = ssl.create_default_context()
with socket.create_connection((hostname, port)) as sock:
    with context.wrap_socket(sock, server_hostname=hostname) as ssock:
        cert_der = ssock.getpeercert(binary_form=True)

# Hash SHA-256 du certificat complet (usage 3 0 1)
cert_hash = hashlib.sha256(cert_der).hexdigest()
print(f"Hash cert complet (3 0 1): {cert_hash}")

# Comparer avec TLSA DNS
answers = dns.resolver.resolve(f"_443._tcp.{hostname}", "TLSA")
for rdata in answers:
    print(f"TLSA DNS: {rdata}")
EOF
```

Résultat attendu : Le hash du certificat TLS actuel correspond exactement au hash dans l'enregistrement TLSA DNS. Aucune divergence.

Remédiation : Si divergence : mettre à jour le TLSA immédiatement (ou re-déployer le bon certificat). La procédure double-publication doit être appliquée pour éviter l'interruption.

Évaluation : ✅ ❌ ⚠️ N/A

S12-C006 — Vérification TLSA pour IMAP/POP3/XMPP si applicable

Criticité : 🟢 Faible
Sources : RFC 7673 (XMPP DANE), RFC 6698

```
# TLSA pour IMAP (port 993)
dig _993._tcp.<domaine-mail> TLSA +short
# TLSA pour POP3S (port 995)
dig _995._tcp.<domaine-mail> TLSA +short
# TLSA pour XMPP C2S (port 5222)
dig _5222._tcp.<domaine-xmpp> TLSA +short
# TLSA pour SIP TLS (port 5061)
dig _5061._tcp.<domaine-sip> TLSA +short
```

Résultat attendu : TLSA publié pour tous les services TLS exposés. Priorité : SMTP (critique), IMAP/POP3S (haute), XMPP/SIP (si applicable).

Remédiation : Étendre le déploiement TLSA aux services de messagerie client (IMAP, POP3S) pour une protection complète de la chaîne email.

Évaluation : ✅ ❌ ⚠️ N/A

S12-C007 — Documentation de la politique DANE dans le DPS

Criticité : 🟢 Faible
Sources : RFC 6698, RFC 7671

Résultat attendu : Le DPS documente : liste des services avec TLSA, types de TLSA utilisés (usage, selector, matching), procédure de mise à jour lors des renouvellements de certificat, procédure de test post-déploiement.

Remédiation : Créer une section DANE/TLSA dans le DPS ou dans la politique de gestion des certificats TLS.

Évaluation : ✅ ❌ ⚠️ N/A

S12-C008 — Automatisation de la mise à jour TLSA lors des renouvellements de certificat

Criticité : 🟠 Élevé
Sources : RFC 7671 §8

```
# Hook certbot pour mise à jour TLSA automatique
# /etc/letsencrypt/renewal-hooks/deploy/update-tlsa.sh
#!/bin/bash
DOMAIN=$RENEWED_DOMAINS
# Calculer nouveau TLSA
NEW_HASH=$(openssl x509 -in /etc/letsencrypt/live/$DOMAIN/cert.pem \
  -pubkey -noout | openssl pkey -pubin -outform DER | \
  openssl dgst -sha256 -binary | xxd -p -c 256)
# Mise à jour via API DNS
# (adapter selon le fournisseur DNS)
echo "Nouveau TLSA pour $DOMAIN: 3 1 1 $NEW_HASH"
```

Résultat attendu : Mise à jour TLSA entièrement automatisée lors des renouvellements de certificat. Aucune intervention manuelle requise. Tests post-déploiement automatisés.

Remédiation : Implémenter le hook certbot (ou équivalent ACME). Tester le hook en simulation `--dry-run`. Configurer des alertes si la mise à jour TLSA échoue après un renouvellement de certificat.

Évaluation : ✅ ❌ ⚠️ N/A

S12-C009 — Monitoring de la cohérence TLSA/certificat (surveillance quotidienne)

Criticité : 🟠 Élevé
Sources : RFC 7672 §2

```
# Script de monitoring TLSA (à exécuter via cron quotidiennement)
#!/usr/bin/env python3
import ssl, socket, hashlib, dns.resolver, sys

def check_tlsa(hostname, port, protocol="tcp"):
    try:
        ctx = ssl.create_default_context()
        with socket.create_connection((hostname, port), timeout=10) as s:
            with ctx.wrap_socket(s, server_hostname=hostname) as ss:
                cert = ss.getpeercert(binary_form=True)
        cert_hash = hashlib.sha256(cert).hexdigest()
        tlsa = dns.resolver.resolve(f"_{port}._{protocol}.{hostname}", "TLSA")
        for r in tlsa:
            if cert_hash in str(r):
                return "OK"
        return f"MISMATCH: cert={cert_hash[:16]}..."
    except Exception as e:
        return f"ERROR: {e}"

print(check_tlsa("<domaine>", 443))
```

Résultat attendu : Monitoring quotidien actif, alertes instantanées en cas de divergence TLSA/certificat.

Évaluation : ✅ ❌ ⚠️ N/A

S12-C010 — Test de compatibilité DANE avec les MTA destinataires courants

Criticité : 🟡 Moyen
Sources : RFC 7672 §7

```
# Tester l'envoi d'email via SMTP DANE vers des MTA supportant DANE
# Postfix avec DANE : envoyer vers un domaine avec DANE (ex: nlnetlabs.nl)
# Vérifier les logs Postfix :
grep "Verified TLS connection\|DANE required\|DANE-verified" /var/log/mail.log | tail -10
```

Résultat attendu : Les emails sortants utilisent SMTP DANE quand le MTA destinataire supporte DANE. Logs Postfix confirment "DANE-verified" ou "Verified TLS connection established".

Remédiation : Activer DANE SMTP dans Postfix : `smtp_tls_security_level = dane` dans `main.cf`. Configurer le resolver DNS Postfix vers un resolver validant DNSSEC.

Évaluation : ✅ ❌ ⚠️ N/A

S13 — DNS over TLS / DoH / DoQ + DNSSEC

Contexte

DoT (DNS over TLS, RFC 7858, port 853), DoH (DNS over HTTPS, RFC 8484, port 443) et DoQ (DNS over QUIC, RFC 9250, port 853 UDP) sont des protocoles de chiffrement du transport DNS. Ils protègent la confidentialité des requêtes DNS contre les écoutes réseau, mais ne remplacent pas DNSSEC (qui protège l'intégrité du contenu). DNSSEC et DoT/DoH/DoQ sont complémentaires.

Un point critique : quand un client utilise DoH vers un provider externe (Cloudflare, Google), il dépend de ce provider pour la validation DNSSEC. Le bit AD dans la réponse DoH indique si le provider a validé. Si le

provider DoH ne valide pas, le client perd la protection DNSSEC. Il faut donc distinguer : (1) DoT/DoH interne (vers le resolver interne validant) et (2) DoT/DoH externe (vers Cloudflare/Google qui valident).

Contrôles S13

S13-C001 — Activation de DoT sur le resolver interne

Criticité : 🟡 Moyen
Sources : RFC 7858, ANSSI REC-DNSSEC-2021 §6.2

```
# Vérifier que DoT est disponible sur le resolver interne
openssl s_client -connect <resolver-interne>:853 2>/dev/null | grep "subject\|issuer"
# Test de résolution via DoT
kdig -d @<resolver-interne> +tls <domaine> A
# Vérifier le certificat TLS du resolver (self-signed ou CA interne)
openssl s_client -connect <resolver-interne>:853 2>/dev/null | openssl x509 -noout -subject -dates
```

Résultat attendu : DoT opérationnel sur le resolver interne (port 853). Certificat TLS valide (CA interne ou Let's Encrypt). Connexion DoT chiffrée avec TLS 1.3 minimum.

Effet de bord : DoT sur le resolver interne expose un nouveau service (port 853) qui peut être utilisé comme vecteur de tunnel exfiltration DNS chiffré. Filtrer les connexions DoT entrantes vers le resolver interne aux seuls réseaux internes.

Remédiation : Configurer DoT dans Unbound (`tls-port: 853`) ou BIND. Certificat TLS via Let's Encrypt (via DNS challenge) ou CA interne. Mettre en place des règles pare-feu pour limiter l'accès au port 853.

Évaluation : ✅ ❌ ⚠️ N/A

S13-C002 — Vérification que le provider DoH/DoT externe effectue la validation DNSSEC

Criticité : 🟠 Élevé
Sources : RFC 8484 §4.1

```
# Si des clients utilisent Cloudflare DoH (1.1.1.1) ou Google DoH :
# Vérifier que ces providers valident DNSSEC
# Test : requête DoH vers 1.1.1.1 sur dnssec-failed.org
curl -H 'accept: application/dns-json' \
  'https://cloudflare-dns.com/dns-query?name=dnssec-failed.org&type=A' | jq '.Status'
# Attendu : Status 2 (SERVFAIL) si validation active
# Vérifier le flag AD dans la réponse DoH :
curl -H 'accept: application/dns-json' \
  'https://cloudflare-dns.com/dns-query?name=<domaine-signé>&type=A' | jq '.AD'
# Attendu : true
```

Résultat attendu : Le provider DoH/DoT utilisé (Cloudflare, Google, Quad9) effectue la validation DNSSEC et transmet le bit AD. Quad9 (9.9.9.9) est recommandé pour sa politique de sécurité stricte et sa validation DNSSEC systématique.

Effet de bord : Si un provider DoH désactive silencieusement la validation DNSSEC (ex: en mode maintenance), les clients perdent la protection sans s'en rendre compte. Monitoring du bit AD nécessaire.

Remédiation : Configurer les clients pour utiliser un provider DoH qui valide DNSSEC. Préférer un resolver DoT/DoH interne pour un contrôle total.

Évaluation : ✅ ❌ ⚠️ N/A

S13-C003 — Vérification de la présence du bit AD dans les réponses DoH

Criticité : 🟡 Moyen
Sources : RFC 8484 §4.2.1, RFC 4035 §3.2.3

```
# Vérifier le bit AD dans les réponses DoH (format JSON)
curl -s -H 'accept: application/dns-json' \
  "https://cloudflare-dns.com/dns-query?name=<domaine>&type=A" | \
  python3 -c "import sys,json; d=json.load(sys.stdin); print('AD:', d.get('AD', False))"
# Format wire (application/dns-message) :
curl -s -H 'accept: application/dns-message' \
  "https://cloudflare-dns.com/dns-query?name=<domaine>&type=A" | \
  python3 -c "import sys; d=sys.stdin.buffer.read(); print('flags hex:', d[2:4].hex())"
# Bit AD = bit 5 du deuxième octet de flags (0x0020)
```

Résultat attendu : Bit AD = true dans les réponses DoH pour les zones correctement signées. Les applications qui dépendent du bit AD pour la sécurité reçoivent des informations fiables.

Remédiation : Si le bit AD est absent : soit le provider ne valide pas, soit il y a un problème de configuration DNSSEC sur la zone. Diagnostiquer avec un resolver validant direct (`dig +dnssec`).

Évaluation : ✅ ❌ ⚠️ N/A

S13-C004 — Politique de blocage du DoH externe (bypass du resolver interne)

Criticité : 🟠 Élevé
Sources : RFC 8484, ANSSI recommandations DNS

```
# Détecter les clients utilisant DoH externe (contournement du resolver interne)
# DoH utilise HTTPS (port 443) vers les IPs des providers DoH
# IPs principales à bloquer (si politique DNS interne):
# Cloudflare: 1.1.1.1, 1.0.0.1
# Google: 8.8.8.8, 8.8.4.4
# Quad9: 9.9.9.9, 9.9.9.11
# Vérifier les règles firewall existantes :
iptables -L FORWARD -n | grep -E "1\.1\.1\.|8\.8\.8\."
# Dans les navigateurs : DoH peut être activé par défaut (Firefox, Chrome)
```

Résultat attendu : Politique DNS interne documentée. Si la résolution DNS interne est obligatoire (zone split-horizon, résolution interne) : DoH externe bloqué au pare-feu. Si DoH externe est autorisé : s'assurer que le provider valide DNSSEC.

Effet de bord : Bloquer DoH externe peut réduire la confidentialité DNS des utilisateurs vers des sites externes. Trade-off entre confidentialité (DoH) et contrôle/sécurité (resolver interne). À documenter dans la politique de sécurité réseau.

Remédiation : Déployer un resolver DoT/DoH interne pour offrir la confidentialité DNS aux utilisateurs tout en maintenant le contrôle. Bloquer DoH externe uniquement si des zones split-horizon sont critiques pour la sécurité.

Évaluation : ✅ ❌ ⚠️ N/A

S13-C005 — Évaluation de DoQ (DNS over QUIC, RFC 9250) pour 2026+

Criticité : 🟢 Faible
Sources : RFC 9250, IETF DPRIVE WG

```
# DoQ = DNS over QUIC (port 853/UDP)
# Avantages : latence réduite (0-RTT), multiplexage, migration de connexion
# Support 2026 : Unbound ≥ 1.17, knot-resolver ≥ 5.7
unbound --version | grep "Unbound"
# Vérifier disponibilité DoQ :
kdig -d @<resolver>:853 +quic <domaine> A 2>/dev/null || echo "DoQ non disponible"
```

Résultat attendu : Évaluation de DoQ planifiée. Déploiement envisagé sur le resolver interne quand le logiciel le supporte en production stable. DoT reste la référence en 2026.

Effet de bord : DoQ utilise UDP/853, nécessite des règles pare-feu supplémentaires. QUIC peut être bloqué par certains équipements réseau d'entreprise (identification incorrecte comme trafic inconnu).

Évaluation : ✅ ❌ ⚠️ N/A

S13-C006 — Configuration TLS 1.3 minimum pour DoT

Criticité : 🟡 Moyen
Sources : RFC 7858 §4.2, ANSSI TLS Guide 2023

```
# Vérifier la version TLS sur le service DoT
openssl s_client -connect <resolver-interne>:853 -tls1_2 2>&1 | grep "Protocol\|Cipher"
openssl s_client -connect <resolver-interne>:853 -tls1_3 2>&1 | grep "Protocol\|Cipher"
# TLS 1.2 peut être accepté pour compatibilité mais TLS 1.3 doit être préféré
# Vérifier la configuration Unbound :
grep "tls-min-version\|tls-cipher" /etc/unbound/unbound.conf
```

Résultat attendu : TLS 1.3 négocié par défaut pour DoT. TLS 1.2 avec suites cipher fortes acceptable en fallback. TLS 1.0 et 1.1 refusés.

Remédiation : Configurer dans Unbound : `tls-ciphersuites: "TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256"`. Pour BIND : `tls local-tls { min-tls-version "TLSv1.2"; ciphers "ECDHE+AESGCM:ECDHE+CHACHA20"; };`

Évaluation : ✅ ❌ ⚠️ N/A

S13-C007 — Monitoring de la disponibilité DoT et intégrité des réponses

Criticité : 🟡 Moyen
Sources : RFC 7858

```
# Monitoring DoT : vérifier disponibilité et cohérence des réponses
# Via kdig :
kdig @<resolver>:853 +tls <domaine> A | grep -E "NOERROR|SERVFAIL|;; MSG SIZE"
# Via script Python :
python3 -c "
import ssl, socket, struct, hashlib
ctx = ssl.create_default_context()
with socket.create_connection(('<resolver>', 853), timeout=5) as s:
    with ctx.wrap_socket(s) as ss:
        print('DoT OK, TLS:', ss.version())
"
```

Résultat attendu : Monitoring DoT actif toutes les 5 minutes. Alertes si service DoT indisponible > 2 minutes ou si les réponses DNSSEC ne contiennent plus le bit AD.

Remédiation : Intégrer dans Nagios/Zabbix/Prometheus. Plugin disponible pour check_dns avec DoT support.

Évaluation : ✅ ❌ ⚠️ N/A

S13-C008 — Séparation des politiques DNS interne et externe (split-horizon avec DNSSEC)

Criticité : 🟡 Moyen
Sources : RFC 6840 §3.7

```
# Vérifier la configuration split-horizon (zones différentes selon réseau source)
grep -A 5 "view internal\|view external" /etc/named.conf 2>/dev/null | head -20
# Test : résolution depuis réseau interne vs externe
dig @<resolver-interne> <domaine-interne> A +dnssec
dig @<resolver-externe> <domaine-interne> A +dnssec  # Ne doit pas retourner l'IP interne
```

Résultat attendu : Architecture split-horizon correctement configurée avec DNSSEC actif sur les deux vues (interne et externe). Les zones internes sont signées avec des ancres de confiance distinctes pour les resolvers internes.

Effet de bord : DNSSEC sur des zones split-horizon est complexe : les resolvers externes ne peuvent pas valider les zones internes (ancre de confiance différente). Cela peut provoquer des SERVFAIL si un client externe essaie de valider une zone interne.

Remédiation : Pour les zones split-horizon, configurer des ancres de confiance locales distinctes pour les resolvers internes. Ne pas publier de DS pour les zones internes dans le DNS public.

Évaluation : ✅ ❌ ⚠️ N/A

S14 — RPKI et BGPsec (Couche Routage Complémentaire)

Contexte

RPKI (Resource Public Key Infrastructure, RFC 6480) et BGPsec (RFC 8205) sécurisent la couche routage BGP qui transporte le trafic vers les serveurs DNS. Sans RPKI, un attaquant peut détourner les annonces BGP vers les adresses IP hébergeant les NS DNSSEC, rendant les signatures DNSSEC inopérantes (les requêtes arrivent chez l'attaquant, pas chez le NS légitime). RPKI et DNSSEC sont strictement complémentaires : DNSSEC sécurise le contenu DNS, RPKI sécurise le chemin réseau vers les NS.

Contrôles S14

S14-C001 — Vérification de la présence de ROA RPKI pour les préfixes des NS autoritatifs

Criticité : 🟠 Élevé
Sources : RFC 6480, RFC 6482, RIPE NCC RPKI

```
# Identifier les IPs des NS autoritatifs
NS_LIST=$(dig <domaine> NS +short)
for ns in $NS_LIST; do
  IP=$(dig $ns A +short | head -1)
  echo "=== NS: $ns IP: $IP ==="
  # Vérifier ROA via RIPE NCC (nécessite accès web ou outil rpki-validator)
  curl -s "https://stat.ripe.net/data/rpki-validation/data.json?resource=$IP" 2>/dev/null | \
    python3 -c "import sys,json; d=json.load(sys.stdin); print('RPKI status:', d.get('data',
{}).get('status','unknown'))"
done
```

Résultat attendu : ROA valide pour tous les préfixes IP hébergeant les NS autoritatifs. Statut RPKI "valid" depuis les validators RIPE NCC, ARIN, ou Hurricane Electric.

Effet de bord : Un ROA invalide ou absent peut déclencher des filtrages BGP "Invalid" par des opérateurs qui ont activé le filtrage RPKI strict, rendant les NS inaccessibles pour une partie des clients Internet.

Remédiation : Créer les ROA dans le portail RPKI de l'opérateur/RIR (RIPE NCC, ARIN, APNIC). Vérifier les préfixes annoncés vs les ROA publiés. Éviter les ROA trop larges (max-length > longueur du préfixe annoncé).

Évaluation : ✅ ❌ ⚠️ N/A

S14-C002 — Vérification de la cohérence IRR/RPKI pour les préfixes DNS

Criticité : 🟡 Moyen
Sources : RFC 6480, RIPE NCC IRR

```
# Vérifier la cohérence entre IRR (Internet Routing Registry) et RPKI
for ns in $(dig <domaine> NS +short); do
  IP=$(dig $ns A +short | head -1)
  echo "=== $ns ($IP) ==="
  # Requête WHOIS pour les routes IRR :
  whois -h whois.radb.net "route:$IP" 2>/dev/null | grep -E "route:|origin:|source:"
  # Vérifier ASN propriétaire
  whois $IP | grep -E "^OrgName:|^netname:|^origin:|^aut-num:"
done
```

Résultat attendu : Les objets route/route6 dans l'IRR sont cohérents avec les ROA RPKI publiés. ASN d'origine identique entre IRR et RPKI. Pas d'objet route IRR obsolète ou incohérent.

Effet de bord : Un objet route IRR permettant une annonce BGP que le ROA RPKI invaliderait → filtrage BGP possible chez des opérateurs RPKI-strict → inaccessibilité des NS.

Remédiation : Nettoyer les objets IRR obsolètes. Aligner IRR et RPKI. Utiliser l'outil RIPE NCC "RPKI Dashboard" pour surveiller la cohérence.

Évaluation : ✅ ❌ ⚠️ N/A

S14-C003 — Évaluation BGPsec pour les chemins vers les NS autoritatifs

Criticité : 🟢 Faible
Sources : RFC 8205, RFC 8206

```
# BGPsec = sécurisation des annonces BGP par signatures cryptographiques
# Maturité 2026 : déploiement très partiel (~5% des opérateurs)
# Vérifier si l'opérateur/hébergeur des NS supporte BGPsec
# Pas de test pratique généralisé disponible en 2026
# Surveiller l'avancement via IETF SIDROPS WG
```

Résultat attendu : Suivi de l'avancement BGPsec documenté. BGPsec non exigé en 2026 (trop peu déployé pour être opérationnel), mais RPKI ROA obligatoire comme protection de premier niveau.

Remédiation : Prioriser RPKI (ROA) comme protection BGP immédiate. Planifier l'évaluation BGPsec à 2028-2030.

Évaluation : ✅ ❌ ⚠️ N/A

S14-C004 — Monitoring RPKI : détection des invalids BGP

Criticité : 🟡 Moyen
Sources : RFC 6480, RIPE NCC BGP Monitoring

```
# Monitorer l'état RPKI des préfixes hébergeant les NS
# Via RIPE NCC BGP Routing Information :
curl -s "https://stat.ripe.net/data/bgp-state/data.json?resource=<prefixe-ns>" 2>/dev/null | \
  python3 -c "import sys,json; d=json.load(sys.stdin); [print(p) for p in d.get('data',{}).get('routes',[])]"
# Alertes : souscrire aux notifications RIPE NCC si préfixe hijacked
```

Résultat attendu : Monitoring RPKI automatisé pour les préfixes des NS. Alertes en cas de détection d'un préfixe "RPKI Invalid" ou d'un BGP hijack potentiel.

Remédiation : Souscrire au service RIPE NCC "Route Object Notifications". Configurer des alertes BGPmon pour les préfixes critiques.

Évaluation : ✅ ❌ ⚠️ N/A

S14-C005 — Documentation de la relation DNSSEC + RPKI dans la politique de sécurité

Criticité : 🟡 Moyen
Sources : RFC 6480, NIST SP 800-81r2

Résultat attendu : La politique de sécurité DNS de l'organisation documente explicitement la complémentarité DNSSEC/RPKI : DNSSEC protège le contenu, RPKI protège le chemin réseau. Les deux sont nécessaires pour une protection complète.

Remédiation : Ajouter une section RPKI au DPS DNSSEC. Documenter les préfixes couverts par des ROA et les procédures de mise à jour.

Évaluation : ✅ ❌ ⚠️ N/A

S14-C006 — Test de résilience DNS face à une simulation de BGP hijack

Criticité : 🟡 Moyen
Sources : RFC 6480, ANSSI

```
# Test de résistance DNSSEC à un BGP hijack simulé :
# Principe : DNSSEC + RPKI empêche un attaquant qui a détourné le trafic BGP
# de forger des réponses DNS valides (il n'a pas la clé privée KSK/ZSK)
# Valider le scénario de menace dans le modèle de risque DNS

# Simulation : si l'attaquant contrôle le NS mais pas la clé :
# Les réponses DNSSEC seraient invalides (BOGUS) pour les resolvers validants
# Les clients non validants seraient victimes

# Test : vérifier que tous les resolvers critiques valident DNSSEC
dig @<resolver-critique> dnssec-failed.org A  # Doit être SERVFAIL
```

Résultat attendu : Confirmation que la chaîne DNSSEC + resolver validant + RPKI ROA offre une protection en profondeur contre les attaques BGP hijacking sur les NS. Documentation dans le modèle de risque DNS.

Remédiation : Documenter le vecteur d'attaque BGP hijack dans l'analyse de risque DNSSEC. DNSSEC ne protège pas contre le hijack si le resolver ne valide pas → insister sur la validation côté resolver (S11).

Évaluation : ✅ ❌ ⚠️ N/A

S15 — Monitoring et Alertes DNSSEC

Contexte

Le monitoring DNSSEC est la couche de détection qui permet d'identifier les problèmes avant qu'ils n'impactent les utilisateurs. Les incidents DNSSEC les plus fréquents (zone bogué suite à RRSIG expirée, rollover ZSK manqué) auraient pu être évités avec un monitoring approprié. La règle empirique : si vous n'avez pas d'alerte RRSIG J-14, vous découvrirez la zone bogué quand vos clients vous l'annoncent.

Contrôles S15

S15-C001 — Monitoring automatique de l'expiration des RRSIG (J-14, J-7, J-1)

Criticité : 🔴 Critique
Sources : RFC 6781 §6.1

```
#!/usr/bin/env python3
"""Script de monitoring RRSIG — à intégrer dans cron toutes les 6h"""
import dns.resolver, datetime, sys

DOMAINS = ["<domaine1>", "<domaine2>"]
ALERT_DAYS = [14, 7, 1]
RRTYPES = ["SOA", "A", "MX", "NS", "DNSKEY"]

for domain in DOMAINS:
    for rrtype in RRTYPES:
        try:
            answers = dns.resolver.resolve(domain, "RRSIG")
            for rrsig in answers:
                expire = rrsig.expiration
                now = datetime.datetime.now(datetime.timezone.utc).timestamp()
                days_left = (expire - now) / 86400
                for alert in ALERT_DAYS:
                    if days_left <= alert:
                        print(f"⚠️ ALERTE J-{alert}: RRSIG {domain} {rrtype} expire dans {days_left:.1f} jours")
        except Exception as e:
            print(f"❌ Erreur {domain} {rrtype}: {e}")
```

Résultat attendu : Script exécuté via cron toutes les 6h. Alertes envoyées par email + PagerDuty à J-14 (info), J-7 (warning), J-1 (critique). Intégré dans le système de monitoring centralisé.

Effet de bord : Trop d'alertes pour des zones avec de nombreuses RRSIG peut provoquer de l'alert fatigue. Regrouper les alertes par domaine et par seuil d'expiration.

Remédiation : Intégrer le script dans Nagios/Prometheus/Zabbix. Créer un check dédié avec les seuils configurables.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C002 — Monitoring de l'expiration des clés KSK et ZSK (J-90, J-30, J-14)

Criticité : 🟠 Élevé
Sources : RFC 6781 §6.2

```
# Monitoring des fichiers de clés BIND
#!/bin/bash
for keyfile in /var/cache/bind/K*.private; do
  domain=$(basename $keyfile | cut -d'+' -f1 | sed 's/^K//')
  inactive=$(grep "^Inactive:" $keyfile 2>/dev/null | awk '{print $2}')
  if [ -n "$inactive" ]; then
    inactive_epoch=$(date -d "$inactive" +%s 2>/dev/null)
    now_epoch=$(date +%s)
    days=$(( (inactive_epoch - now_epoch) / 86400 ))
    flags=$(grep "^Flags:" ${keyfile%.private}.key 2>/dev/null | awk '{print $3}')
    type="ZSK"
    [ "$flags" = "257" ] && type="KSK"
    echo "$type $domain expire dans $days jours ($inactive)"
    if [ $days -le 14 ]; then echo "🔴 CRITIQUE : $type expire J-$days"; fi
    if [ $days -le 30 ]; then echo "🟠 ALERTE : $type expire J-$days"; fi
  fi
done
```

Résultat attendu : Alertes KSK J-90 (planification rollover), J-30 (préparation), J-7 (urgence). Alertes ZSK J-14 (warning), J-7 (alerte), J-1 (critique).

Remédiation : Automatiser le rollover ZSK (KASP) pour ne jamais atteindre le seuil J-7. Pour KSK, planifier le rollover dès l'alerte J-90.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C003 — Monitoring de la propagation DS après soumission au registrar

Criticité : 🟠 Élevé
Sources : RFC 7344, RFC 4034

```
# Script de suivi de propagation DS après soumission
#!/usr/bin/env python3
"""Surveille la propagation DS depuis plusieurs resolvers"""
import dns.resolver, time

DOMAIN = "<domaine>"
NEW_KEYTAG = <nouveau-keytag>
RESOLVERS = ["9.9.9.9", "1.1.1.1", "8.8.8.8", "208.67.222.222"]

while True:
    propagated = []
    for resolver in RESOLVERS:
        r = dns.resolver.Resolver()
        r.nameservers = [resolver]
        try:
            answers = r.resolve(DOMAIN, "DS")
            for rdata in answers:
                if rdata.key_tag == NEW_KEYTAG:
                    propagated.append(resolver)
        except:
            pass
    print(f"DS propagé chez : {propagated}/{len(RESOLVERS)} resolvers")
    if len(propagated) == len(RESOLVERS):
        print("✅ Propagation complète !")
        break
    time.sleep(300)  # Vérifier toutes les 5 minutes
```

Résultat attendu : Suivi temps réel de la propagation DS pendant les rollovers KSK. Confirmation de propagation complète avant de passer à l'étape suivante du rollover.

Remédiation : Intégrer ce script dans le runbook de rollover KSK comme checkpoint obligatoire avant retrait de l'ancienne DNSKEY.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C004 — Monitoring du taux SERVFAIL DNSSEC sur le resolver interne

Criticité : 🟡 Moyen
Sources : RFC 6781 §6, ICANN

```
# Métriques BIND via rndc stats
rndc stats
# Lire /var/named/data/named_stats.txt
grep -E "queries resulted in SERVFAIL|Queries received" /var/named/data/named_stats.txt
# Métriques Unbound
unbound-control stats_noreset | grep -E "total.num.queries|total.num.query.tcp|num.answer.rcode.SERVFAIL"
# Calculer le taux SERVFAIL :
TOTAL=$(unbound-control stats_noreset | grep "total.num.queries=" | cut -d= -f2)
SERVFAIL=$(unbound-control stats_noreset | grep "SERVFAIL=" | cut -d= -f2)
echo "Taux SERVFAIL: $(echo "scale=4; $SERVFAIL * 100 / $TOTAL" | bc)%"
```

Résultat attendu : Taux SERVFAIL < 0.1% en conditions normales. Alerte si > 0.5% pendant > 5 minutes. Dashboard Grafana avec métrique SERVFAIL rate.

Remédiation : Configurer Prometheus scrape sur Unbound (unbound_exporter) ou BIND stats. Dashboard Grafana avec alertes Alertmanager.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C005 — Monitoring de la cohérence DNSKEY entre tous les NS autoritatifs

Criticité : 🟠 Élevé
Sources : RFC 4035 §4.1

```
#!/bin/bash
# Comparer les DNSKEY sur tous les NS et alerter en cas d'incohérence
DOMAIN="<domaine>"
NS_LIST=$(dig $DOMAIN NS +short)
REFERENCE_DNSKEY=""

for ns in $NS_LIST; do
  CURRENT=$(dig @$ns $DOMAIN DNSKEY +short | sort | md5sum)
  if [ -z "$REFERENCE_DNSKEY" ]; then
    REFERENCE_DNSKEY=$CURRENT
  elif [ "$CURRENT" != "$REFERENCE_DNSKEY" ]; then
    echo "❌ INCOHÉRENCE DNSKEY sur $ns !"
    dig @$ns $DOMAIN DNSKEY +short
  fi
done
echo "✅ DNSKEY cohérentes sur tous les NS"
```

Résultat attendu : DNSKEY identiques sur tous les NS. Script exécuté toutes les heures. Alerte immédiate en cas d'incohérence (SERVFAIL intermittent possible).

Remédiation : En cas d'incohérence : vérifier le transfert de zone AXFR/IXFR entre NS primaire et le NS incohérent. Forcer un transfert de zone si nécessaire.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C006 — Utilisation de sondes RIPE Atlas pour monitoring DNSSEC géodistribué

Criticité : 🟡 Moyen
Sources : RIPE Atlas DNS Monitoring

```
# Créer des mesures RIPE Atlas pour surveiller la validation DNSSEC
# depuis des points de mesure géodistribués
# API RIPE Atlas : https://atlas.ripe.net/api/v2/measurements/
# Mesure DNS : résolution de <domaine> avec DNSSEC activé
# Vérifier le bit AD dans les réponses
# 5 sondes minimum, sur 3 continents différents
```

Résultat attendu : Au moins 5 sondes RIPE Atlas configurées pour surveiller la validation DNSSEC de toutes les zones critiques, sur des continents différents. Alertes mail si validation échoue sur > 2 sondes.

Remédiation : Créer un compte RIPE Atlas, configurer des mesures DNS récurrentes. Budget : ~500 crédits RIPE Atlas par semaine pour 5 sondes quotidiennes.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C007 — Dashboard métriques DNSSEC (Grafana + Prometheus ou Zabbix)

Criticité : 🟡 Moyen
Sources : RFC 6781 §6, ICANN DNSSEC Monitoring

```
# Métriques à exposer dans le dashboard :
# - RRSIG_days_until_expiration{domain,rrtype}
# - DNSKEY_days_until_inactive{domain,keytype}
# - dns_servfail_rate{resolver}
# - dnssec_validation_success{domain,resolver}
# - ds_propagation_count{domain}
# Exporters disponibles :
# - unbound_exporter (Prometheus)
# - bind_exporter (Prometheus)
# - check_dnssec (Nagios/Zabbix)
```

Résultat attendu : Dashboard DNSSEC opérationnel avec toutes les métriques clés visibles en temps réel. Accessible à l'équipe DNS et RSSI. Historique conservé ≥ 90 jours.

Remédiation : Déployer unbound_exporter + Prometheus + Grafana. Importer un dashboard DNSSEC depuis grafana.com ou créer un dashboard personnalisé basé sur les métriques identifiées.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C008 — Alertes sur changement inattendu de DS chez le parent

Criticité : 🔴 Critique
Sources : RFC 4034 §5

```
# Surveiller les changements de DS qui ne correspondent pas aux rollovers planifiés
#!/usr/bin/env python3
import dns.resolver, hashlib, json, os

DOMAIN = "<domaine>"
STATE_FILE = "/tmp/ds_state.json"

def get_ds():
    return sorted([str(r) for r in dns.resolver.resolve(DOMAIN, "DS")])

current = get_ds()
if os.path.exists(STATE_FILE):
    with open(STATE_FILE) as f:
        previous = json.load(f)
    if current != previous:
        print(f"⚠️ ALERTE : DS modifié de façon inattendue pour {DOMAIN} !")
        print(f"Avant : {previous}")
        print(f"Après : {current}")

with open(STATE_FILE, 'w') as f:
    json.dump(current, f)
```

Résultat attendu : Monitoring exécuté toutes les 15 minutes. Toute modification de DS non planifiée déclenche une alerte P0 (possible compromission ou erreur registrar).

Remédiation : En cas d'alerte DS inattendu : vérifier immédiatement si une opération de rollover est en cours. Si non : suspicion de compromission ou d'erreur registrar → contacts d'urgence registrar immédiats.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C009 — Rapport hebdomadaire DNSSEC automatisé

Criticité : 🟢 Faible
Sources : RFC 6781 §6

```
# Script de rapport hebdomadaire DNSSEC
#!/bin/bash
echo "=== Rapport DNSSEC Hebdomadaire $(date) ==="
echo ""
echo "1. Statut des zones :"
for zone in <zone1> <zone2>; do
  delv $zone A 2>&1 | grep -E "fully validated|BOGUS" | head -1
done
echo ""
echo "2. Expiration RRSIG (jours restants) :"
# Appeler le script de monitoring RRSIG
echo ""
echo "3. Statut des clés (jours restants) :"
# Appeler le script de monitoring clés
echo ""
echo "4. Taux SERVFAIL semaine :"
unbound-control stats_noreset | grep SERVFAIL
```

Résultat attendu : Rapport hebdomadaire envoyé automatiquement par email au RSSI et à l'équipe DNS le lundi matin. Inclut tous les indicateurs DNSSEC clés.

Remédiation : Automatiser via cron `0 8 * * 1 /opt/scripts/dnssec-weekly-report.sh | mail -s "Rapport DNSSEC" rssi@entreprise.fr`.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C010 — Test de validation DNSSEC depuis des outils en ligne (audit trimestriel)

Criticité : 🟢 Faible
Sources : AFNIC Zonemaster, SIDN Labs DNSSEC Check

```
# Outils de vérification DNSSEC en ligne à utiliser trimestriellement :
# 1. https://zonemaster.net — test complet DNSSEC (AFNIC/SIDN)
# 2. https://dnsviz.net — visualisation graphique chaîne de confiance
# 3. https://dnssec-analyzer.verisignlabs.com — analyse Verisign Labs
# 4. https://check.sidnlabs.nl — DNSSEC + DANE check SIDN Labs

# Ou via CLI :
zonemaster-cli <domaine> --test dnssec 2>/dev/null | grep -E "ERROR|WARNING|NOTICE" | head -20
```

Résultat attendu : Aucune erreur DNSSEC critique sur Zonemaster. Avertissements documentés avec justification. Test réalisé trimestriellement et résultats archivés.

Remédiation : Pour chaque erreur Zonemaster : analyser la cause, corriger et re-tester jusqu'à validation complète.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C011 — Monitoring de la disponibilité du service DNSSEC après un déploiement

Criticité : 🟠 Élevé
Sources : RFC 6781 §7.1

```
# Post-déploiement : surveiller intensivement pendant les 24 premières heures
# Test toutes les 5 minutes depuis plusieurs resolvers
watch -n 300 '
echo "=== $(date) ==="
for r in 9.9.9.9 1.1.1.1 8.8.8.8; do
  result=$(dig @$r <domaine> A +dnssec 2>/dev/null | grep -E "SERVFAIL|NOERROR|ad")
  echo "[$r] $result"
done
'
```

Résultat attendu : Monitoring intensif actif les 24-48h suivant tout déploiement DNSSEC (nouveau signing, rollover, changement de paramètres). Aucun SERVFAIL non expliqué.

Remédiation : Définir la fenêtre de surveillance post-déploiement dans le runbook DNSSEC. Critère de rollback : SERVFAIL > 5% pendant > 5 minutes.

Évaluation : ✅ ❌ ⚠️ N/A

S15-C012 — Inventaire et test des contacts de monitoring et d'astreinte DNSSEC

Criticité : 🟡 Moyen
Sources : ISO 27001 A.16.1.1

```
# Vérifier que les alertes DNSSEC parviennent aux bonnes personnes
# Test d'alerte : déclencher manuellement une alerte de test
# Vérifier la liste de distribution : dns-ops@entreprise.fr, astreinte@entreprise.fr
# Vérifier que les alertes fonctionnent en dehors des heures ouvrables
```

Résultat attendu : Chaîne d'escalade DNSSEC documentée et testée. Test d'alerte mensuel. Contact d'astreinte disponible 24/7 pour les incidents P0 (zone bogué).

Remédiation : Configurer PagerDuty/Opsgenie avec l'escalade DNSSEC. Tester la chaîne d'escalade trimestriellement.

Évaluation : ✅ ❌ ⚠️ N/A

S16 — Réponse aux Incidents DNSSEC

Contexte

Les incidents DNSSEC ont la particularité d'être binaires et immédiats : une zone est soit valide (les clients résolvent normalement) soit bogué (100% des resolvers validants retournent SERVFAIL). Il n'y a pas de dégradation progressive. Cela rend la réponse aux incidents DNSSEC particulièrement critique et urgente. La procédure doit être connue par coeur par l'équipe DNS : toute minute perdue à chercher la procédure est une minute de plus avec la zone bogué.

Contrôles S16

S16-C001 — Procédure documentée en cas d'expiration RRSIG en production

Criticité : 🔴 Critique
Sources : RFC 6781 §7.1

```
# Procédure d'urgence RRSIG expirée :
# 1. DIAGNOSTIC (2 minutes)
dig <domaine> RRSIG SOA +short | awk '{print "Expire:", $5}'
delv <domaine> A 2>&1 | grep -E "BOGUS|expired"
# 2. RE-SIGNATURE FORCÉE (5 minutes)
# BIND :
rndc sign <domaine>
rndc notify <domaine>
# PowerDNS :
pdns_control rectify-zone <domaine>
pdns_control notify <domaine>
# Knot DNS :
knotc zone-sign <domaine>
# 3. VÉRIFICATION (3 minutes)
dig <domaine> RRSIG SOA +short  # Vérifier nouvelle date expiration
delv <domaine> A  # "fully validated" attendu
# 4. NOTIFICATION (si RTO > 30 min : notifier les parties prenantes)
```

Résultat attendu : Procédure écrite, testée, accessible en < 30 secondes (signets, wiki, runbook imprimé). RTO ≤ 30 minutes depuis détection à résolution.

Effet de bord : La re-signature forcée régénère toutes les RRSIG et peut prendre 1-10 minutes selon la taille de la zone. Pendant ce temps, les réponses DNSSEC peuvent être incohérentes. Surveiller jusqu'à stabilisation complète.

Remédiation : Tester la procédure en staging avec une RRSIG expirée intentionnellement. Chronométrer le temps de résolution et l'inclure dans les objectifs RTO.

Évaluation : ✅ ❌ ⚠️ N/A

S16-C002 — Procédure de compromission ZSK (rollover d'urgence ZSK)

Criticité : 🔴 Critique
Sources : RFC 6781 §7.2

```
# Procédure rollover ZSK d'urgence (compromission suspectée) :
# 1. Générer immédiatement une nouvelle ZSK
dnssec-keygen -a ECDSAP256SHA256 <domaine>
dnssec-settime -P now -A now K<domaine>.+013+<newtag>.key
# 2. Ajouter à la zone et re-signer
rndc loadkeys <domaine>
rndc sign <domaine>  # Re-signe tout avec la nouvelle ZSK
# 3. Désactiver l'ancienne ZSK (marquer inactive)
dnssec-settime -I now K<domaine>.+013+<oldtag>.key
# 4. Les RRSIG de l'ancienne ZSK expireront naturellement
# NE PAS supprimer l'ancienne ZSK immédiatement (voir S08-C005)
# 5. Vérifier la validation
delv <domaine> A
```

Résultat attendu : Nouvelle ZSK opérationnelle en < 15 minutes. Validation DNSSEC restaurée. Ancienne ZSK désactivée mais non supprimée immédiatement.

Effet de bord : Un rollover ZSK d'urgence peut provoquer une brève incohérence DNS (SERVFAIL intermittent de quelques secondes à quelques minutes) pendant la propagation de la nouvelle ZSK sur les NS secondaires.

Remédiation : Documenter le runbook de compromission ZSK. Préparer les commandes à l'avance (ne pas les composer sous pression). Pratiquer le rollover en staging régulièrement.

Évaluation : ✅ ❌ ⚠️ N/A

S16-C003 — Procédure de compromission KSK (coordination urgente registrar)

Criticité : 🔴 Critique
Sources : RFC 6781 §7.3, RFC 5011

```
# Procédure de compromission KSK — implique le registrar parent
# IMPORTANT : coordination impérative avec le registrar

# 1. Générer nouvelle KSK et publier dans la zone
dnssec-keygen -a ECDSAP256SHA256 -f KSK <domaine>
# 2. Publier l'ancienne KSK avec flag REVOKE (RFC 5011)
dnssec-settime -R now K<domaine>.+013+<oldksk-tag>.key
rndc loadkeys <domaine>
rndc sign <domaine>
# 3. CONTACTER LE REGISTRAR D'URGENCE
# → Supprimer l'ancien DS, publier le nouveau DS (nouvelle KSK)
# 4. Attendre propagation nouveau DS (TTL_DS)
# 5. Retirer l'ancienne KSK révoquée
dnssec-settime -I +86400 -D +172800 K<domaine>.+013+<oldksk-tag>.key
```

Résultat attendu : Procédure d'urgence KSK documentée avec numéro hotline registrar. Contact registrar testé. RTO ≤ 4h (incluant délai registrar). Révocation via RFC 5011 pour signaler aux resolvers avec ancre de confiance locale.

Effet de bord : Pendant l'attente de propagation du nouveau DS (24-48h selon registrar), la zone est BOGUÉ pour tous les resolvers validants. Il n'existe pas de moyen de court-circuiter ce délai. La zone peut être temporairement passée en mode non-signé (retrait de DNSSEC) si l'impact business est inacceptable.

Remédiation : Avoir le numéro hotline de chaque registrar dans le runbook. Considérer des registrars avec SLA DS update < 1h pour les zones critiques (certains registrars offrent des accès API pour les mises à jour DS d'urgence).

Évaluation : ✅ ❌ ⚠️ N/A

S16-C004 — Procédure de zone bogué : diagnostic et rollback

Criticité : 🔴 Critique
Sources : RFC 6781 §7.1, RFC 4035

```
# Arbre de décision diagnostic zone bogué :

# 1. Identifier la cause
delv <domaine> A +vtrace 2>&1 | tail -20

# Cause A : RRSIG expirée
dig <domaine> RRSIG SOA +short | awk '{print "Expiration:", $5}'
# Remédiation A : rndc sign <domaine>

# Cause B : DS orphelin (KSK supprimée mais DS chez parent)
dig <domaine> DS +short  # keytag DS
dig <domaine> DNSKEY +short | awk '{if($1==257) print}'  # keytag KSK
# Remédiation B : ajouter la KSK manquante OU retirer le DS chez le registrar

# Cause C : DS incorrect (hash ne correspond pas à la DNSKEY)
# Remédiation C : soumettre le bon DS au registrar

# Cause D : Ancre de confiance resolver obsolète
dig @<resolver> . DNSKEY +short | awk '{if($1==257) print $4}' | wc -c
# Remédiation D : mettre à jour l'ancre de confiance racine (RFC 5011)

echo "Option de dernier recours : désactiver DNSSEC temporairement"
echo "NE FAIRE QUE si le business impact justifie la perte de protection DNSSEC"
```

Résultat attendu : Arbre de décision documenté couvrant les 4 causes principales de zone bogué. Procédure de rollback (désactivation temporaire DNSSEC) documentée avec critères de déclenchement.

Remédiation : Inclure l'arbre de décision dans le runbook DNSSEC. Le plastifier et l'afficher dans le bureau des opérateurs DNS.

Évaluation : ✅ ❌ ⚠️ N/A

S16-C005 — RTO et RPO définis pour chaque type d'incident DNSSEC

Criticité : 🟠 Élevé
Sources : ISO 27001 A.17.1, ITIL

```
# Tableau des RTO DNSSEC recommandés :

| Incident                              | RTO cible | RPO         |
|---------------------------------------|-----------|-------------|
| RRSIG expirée (zone bogué)           | 30 min    | J-1 signing |
| ZSK compromise (rollover d'urgence)  | 1h        | N/A         |
| KSK compromise (rollover + registrar) | 4h        | N/A         |
| DS orphelin                           | 2h        | N/A         |
| Resolver validant hors-service        | 15 min    | N/A         |
| TLSA/certificat incohérent            | 1h        | N/A         |
```

Résultat attendu : RTO/RPO définis, documentés, approuvés par la direction. Tests de reprise réguliers pour valider les RTO.

Remédiation : Intégrer les RTO DNSSEC dans le plan de continuité DNS de l'organisation. Les soumettre à validation du RSSI et de la direction.

Évaluation : ✅ ❌ ⚠️ N/A

S16-C006 — Contact registrar d'urgence documenté et testé pour chaque domaine

Criticité : 🔴 Critique
Sources : RFC 6781 §7.3

Résultat attendu : Fiche de contact urgence par registrar : URL interface admin, numéro hotline 24/7 (si disponible), API endpoint pour mise à jour DS, contact email support prioritaire, délai SLA DS update. Fiche testée au moins une fois par an.

Remédiation : Créer la fiche de contact pour chaque registrar. Stocker dans un endroit accessible hors-ligne (impression physique, coffre-fort). Tester le canal de contact en période non-urgente.

Évaluation : ✅ ❌ ⚠️ N/A

S16-C007 — Exercice annuel de simulation d'incident DNSSEC (tabletop)

Criticité : 🟡 Moyen
Sources : NIST SP 800-81r2 §7, ISO 22301

```
# Scénarios d'exercice recommandés :
# Exercice 1 : "RRSIG expirée un vendredi soir"
#   → Tester la détection automatique et la procédure de rappel
# Exercice 2 : "KSK suspecte de compromission"
#   → Tester la décision de rollover d'urgence et la communication registrar
# Exercice 3 : "Registrar inaccessible pendant 48h"
#   → Tester les alternatives (migration d'urgence, désactivation DNSSEC)
# Exercice 4 : "Zone bogué non détectée pendant 2h"
#   → Tester le post-mortem et les améliorations monitoring
```

Résultat attendu : Exercice tabletop réalisé annuellement, avec toute l'équipe DNS et le RSSI. Rapport de l'exercice documantant les gaps découverts et les mesures correctives.

Remédiation : Planifier l'exercice dans le calendrier annuel de sécurité. Faire appel à un prestataire externe pour une évaluation indépendante tous les 2 ans.

Évaluation : ✅ ❌ ⚠️ N/A

S16-C008 — Post-mortem systématique après tout incident DNSSEC

Criticité : 🟡 Moyen
Sources : RFC 6781 §7, ITIL Problem Management

```
# Template de post-mortem DNSSEC :
# 1. Chronologie de l'incident (heure de début, heure de détection, heure de résolution)
# 2. Cause racine (Root Cause Analysis)
# 3. Impact (nombre d'utilisateurs affectés, durée, services touchés)
# 4. Actions de résolution effectuées
# 5. Timeline de résolution vs RTO cible
# 6. Actions préventives (pour éviter la récurrence)
# 7. Mise à jour du runbook si nécessaire
# 8. Validation par le RSSI
```

Résultat attendu : Post-mortem rédigé dans les 48h suivant chaque incident DNSSEC. Archivé dans le DPS. Les actions préventives sont suivies jusqu'à leur implémentation.

Remédiation : Créer un template de post-mortem DNSSEC. Inclure dans la politique de gestion des incidents DNS.

Évaluation : ✅ ❌ ⚠️ N/A

S17 — Conformité NIS 2 / ANSSI / Réglementaire

Contexte

DNSSEC est explicitement mentionné dans NIS 2 (Directive UE 2022/2555, Art. 21 §2.d) comme mesure de sécurité pour l'authenticité et l'intégrité des communications réseau, incluant le DNS. Les entités essentielles (OIV/OSE) et entités importantes ont l'obligation de mettre en œuvre des mesures techniques et organisationnelles adaptées, dont DNSSEC fait partie. L'ANSSI a publié ses recommandations DNSSEC en 2021 (REC-DNSSEC-2021), qui constituent le référentiel français. Le règlement DORA (Digital Operational Resilience Act, 2022/2554) impose des exigences similaires au secteur financier.

Contrôles S17

S17-C001 — Vérification de la conformité NIS 2 Art.21 sur l'authenticité DNS

Criticité : 🔴 Critique (entités essentielles et importantes)
Sources : NIS 2 Art.21 §2.d, ENISA NIS 2 Implementation Guide

```
# Article 21 §2.d NIS 2 :
# "des mesures relatives à la sécurité de la chaîne d'approvisionnement"
# et "l'authenticité et l'intégrité des informations"
# DNSSEC satisfait directement cette exigence pour la couche DNS

# Vérifier la couverture DNSSEC des domaines critiques de l'entité
CRITICAL_DOMAINS=("entreprise.fr" "espace-client.entreprise.fr" "api.entreprise.fr")
for domain in "${CRITICAL_DOMAINS[@]}"; do
  status=$(delv $domain A 2>&1 | grep -E "fully validated|BOGUS|insecure" | head -1)
  echo "$domain : $status"
done
```

Résultat attendu : Tous les domaines critiques de l'entité NIS 2 sont signés DNSSEC et validés. Evidence documentée pour audit NIS 2 (rapport de validation, capture delv, certificat de conformité).

Effet de bord : L'absence de DNSSEC sur un domaine critique peut être signalée comme non-conformité NIS 2 lors d'un audit. Depuis le 17 octobre 2024, NIS 2 est transposée en droit français (Loi n°2024-1172).

Remédiation : Établir un mapping entre les domaines critiques et les exigences NIS 2. Déployer DNSSEC en priorité sur les domaines les plus exposés. Documenter la conformité dans le registre des mesures de sécurité NIS 2.

Évaluation : ✅ ❌ ⚠️ N/A

S17-C002 — Conformité aux recommandations ANSSI REC-DNSSEC-2021

Criticité : 🟠 Élevé
Sources : ANSSI REC-DNSSEC-2021

```
# Récapitulatif des recommandations ANSSI DNSSEC 2021 :
# R1 : Déployer DNSSEC sur toutes les zones DNS publiques de l'entité
# R2 : Utiliser les algorithmes ECDSA P-256 (13) ou Ed25519 (15)
# R3 : Durée de vie KSK : 1-2 ans maximum
# R4 : Durée de vie ZSK : 1-3 mois maximum
# R5 : Durée de validité RRSIG : 7-30 jours
# R6 : NSEC3 avec iterations=0 (RFC 9276)
# R7 : Stockage KSK dans HSM certifié
# R8 : Monitoring automatique de l'expiration RRSIG
# R9 : DPS (DNSSEC Practice Statement) rédigé et maintenu
# R10 : Résolveurs internes en mode validant
```

Résultat attendu : Toutes les recommandations ANSSI R1-R10 implémentées ou documentées avec justification d'exception. Rapport de conformité ANSSI disponible pour audit.

Remédiation : Créer une matrice de conformité ANSSI REC-DNSSEC-2021 avec le statut de chaque recommandation. Planifier l'implémentation des recommandations manquantes.

Évaluation : ✅ ❌ ⚠️ N/A

S17-C003 — Obligation DNSSEC pour les zones .fr (AFNIC DPS)

Criticité : 🟡 Moyen (si domaine .fr)
Sources : AFNIC DPS (DNS Practice Statement), RFC 7344

```
# Le TLD .fr est signé DNSSEC depuis 2010
# L'AFNIC publie son DPS à : https://www.afnic.fr/domaines/dnssec/
# Les registrars .fr acceptent les DS pour activer DNSSEC sur les domaines .fr
# Vérifier que les domaines .fr de l'entité ont bien leur DS chez AFNIC
dig <domaine>.fr DS +short @ns1.nic.fr  # NS autoritatif du TLD .fr
```

Résultat attendu : DS publié chez AFNIC pour tous les domaines .fr de l'entité. Chaîne root → .fr → domaine.fr validée.

Remédiation : Contacter le registrar .fr pour soumettre le DS. L'AFNIC dispose d'une interface de validation DS à https://www.afnic.fr/domaines/dnssec/.

Évaluation : ✅ ❌ ⚠️ N/A

S17-C004 — Conformité DORA Art.8 pour le secteur financier

Criticité : 🟠 Élevé (entités financières régulées par DORA)
Sources : DORA (EU) 2022/2554 Art.8, EBA Guidelines

```
# DORA Art.8 §1 : "les entités financières gèrent toutes les sources de risques liées
# aux TIC, y compris les risques associés aux réseaux et services de communications"
# DNSSEC s'inscrit dans la gestion des risques DNS (spoofing, empoisonnement)

# Exigences DORA applicables à DNSSEC :
# - Test de résilience des systèmes DNS (Art.24-25)
# - Cartographie des actifs critiques incluant les zones DNS (Art.8 §5)
# - Plans de continuité DNS (Art.11)
# - Reporting incidents DNS significatifs (Art.19)
```

Résultat attendu : DNSSEC intégré dans le programme de résilience opérationnelle numérique DORA. Tests de résilience DNS documentés. Incidents zone bogué reportés si significatifs.

Remédiation : Inclure DNSSEC dans le périmètre DORA : cartographie des zones DNS critiques, tests de résilience (simulation zone bogué), procédures d'incident DNS.

Évaluation : ✅ ❌ ⚠️ N/A

S17-C005 — Audit DNSSEC annuel documenté (rapport d'audit et preuves)

Criticité : 🟠 Élevé
Sources : ISO 27001 A.12.7, NIS 2 Art.21

```
# Audit DNSSEC annuel — éléments à documenter :
# 1. Statut de validation de toutes les zones (delv +vtrace)
# 2. Algorithmes utilisés (conformité RFC 8624)
# 3. Durées de vie des clés (conformité RFC 6781)
# 4. Paramètres NSEC3 (conformité RFC 9276 : iterations=0)
# 5. Résultats Zonemaster pour chaque zone
# 6. Statut TLSA/DANE pour les services critiques
# 7. Test resolver validant (dnssec-failed.org = SERVFAIL)
# 8. Revue du DPS (mise à jour si nécessaire)
# 9. Revue des incidents DNSSEC de l'année
# 10. Plan de rollover KSK pour les 12 mois suivants
```

Résultat attendu : Rapport d'audit DNSSEC annuel rédigé, signé par le RSSI, archivé. Preuves de conformité disponibles pour audit NIS 2 / ISO 27001.

Remédiation : Planifier l'audit DNSSEC dans le calendrier des audits sécurité de l'organisation. Faire appel à un prestataire externe tous les 2 ans pour un audit indépendant.

Évaluation : ✅ ❌ ⚠️ N/A

S17-C006 — DPS (DNSSEC Practice Statement) rédigé, approuvé et publié

Criticité : 🟡 Moyen
Sources : RFC 6781 §3.1, ANSSI REC-DNSSEC-2021 §4, AFNIC DPS

```
# Le DPS doit couvrir (minimum) :
# 1. Portée : domaines couverts, responsable DNS
# 2. Politique de génération des clés (algo, taille, HSM)
# 3. Durées de vie KSK et ZSK
# 4. Procédures de rollover (KSK et ZSK)
# 5. Paramètres NSEC3
# 6. Procédures de re-signature RRSIG
# 7. Gestion des incidents DNSSEC
# 8. Contacts d'urgence
# 9. Politique de sauvegarde des clés
# 10. Procédure d'audit et de révision du DPS
```

Résultat attendu : DPS rédigé, approuvé par le RSSI et la direction, versionné (contrôle de version), révisé semestriellement. Pour les entités NIS 2, le DPS peut être requis comme preuve de maturité DNSSEC.

Remédiation : Rédiger le DPS en s'inspirant des exemples publiés par l'AFNIC et l'ICANN. Faire valider par un expert DNSSEC externe.

Évaluation : ✅ ❌ ⚠️ N/A

S17-C007 — Cartographie des domaines critiques sous DNSSEC (registre de conformité)

Criticité : 🟡 Moyen
Sources : NIS 2 Art.8 §3, ISO 27001 A.5.9

```
# Générer un rapport de couverture DNSSEC pour tous les domaines de l'entité
#!/bin/bash
DOMAINS=($(cat /etc/dns-domains.txt))  # Liste des domaines de l'organisation
for domain in "${DOMAINS[@]}"; do
  dnskey=$(dig $domain DNSKEY +short 2>/dev/null | wc -l)
  ds=$(dig $domain DS +short 2>/dev/null | wc -l)
  if [ $dnskey -gt 0 ] && [ $ds -gt 0 ]; then
    echo "✅ $domain : DNSSEC actif et chaîne de confiance établie"
  elif [ $dnskey -gt 0 ] && [ $ds -eq 0 ]; then
    echo "⚠️ $domain : zone signée mais sans DS parent (island of security)"
  else
    echo "❌ $domain : DNSSEC non déployé"
  fi
done
```

Résultat attendu : Registre de conformité DNSSEC listant tous les domaines avec leur statut, mis à jour mensuellement.

Remédiation : Créer et maintenir le registre. Inclure dans le rapport RSSI mensuel.

Évaluation : ✅ ❌ ⚠️ N/A

S17-C008 — Intégration DNSSEC dans la politique de sécurité système d'information (PSSI)

Criticité : 🟡 Moyen
Sources : ISO 27001 A.5.1, NIS 2 Art.21

Résultat attendu : La PSSI de l'organisation mentionne explicitement DNSSEC comme contrôle de sécurité obligatoire pour les zones DNS publiques. La politique définit : qui est responsable (équipe DNS), fréquence des rollovers, exigences de monitoring, conformité aux normes RFC 8624, RFC 9276, ANSSI.

Remédiation : Réviser la PSSI pour inclure une section DNSSEC. Valider avec le RSSI et la direction. Diffuser à toutes les équipes concernées (DNS, réseau, sécurité).

Évaluation : ✅ ❌ ⚠️ N/A

S18 — Sécurité Opérationnelle DNSSEC

Contexte

La sécurité opérationnelle DNSSEC concerne les processus, les rôles, et les contrôles organisationnels qui entourent les opérations techniques DNSSEC. Un déploiement DNSSEC techniquement parfait peut être compromis par des défaillances opérationnelles : manque de formation, absence de procédures, single point of failure humain, accès non contrôlés aux clés.

Contrôles S18

S18-C001 — Matrice RACI des opérations DNSSEC (Responsable, Approbateur, Consulté, Informé)

Criticité : 🟡 Moyen
Sources : ISO 27001 A.6.1.1, NIST SP 800-81r2 §7

```
# Matrice RACI recommandée pour les opérations DNSSEC :

| Opération                    | DNS Admin | RSSI | Direction | Registrar |
|------------------------------|-----------|------|-----------|-----------|
| Rollover ZSK routine         | R/A       | I    | -         | -         |
| Rollover KSK planifié        | R         | A    | I         | C         |
| Rollover KSK d'urgence       | R         | A/I  | I         | R/C       |
| Modification paramètres NSEC | R         | A    | -         | -         |
| Activation DNSSEC nouvelle zone| R       | A    | -         | C         |
| Désactivation DNSSEC         | R         | A    | I         | C         |
| Audit annuel DNSSEC          | C         | R/A  | I         | -         |
| Réponse incident zone bogué  | R         | I    | I (si >30min) | C |

R=Responsible, A=Accountable, C=Consulted, I=Informed
```

Résultat attendu : Matrice RACI documentée, approuvée, diffusée à toutes les parties prenantes. Revue annuelle.

Remédiation : Créer la matrice RACI en atelier avec l'équipe DNS et le RSSI. Intégrer dans le DPS.

Évaluation : ✅ ❌ ⚠️ N/A

S18-C002 — Formation obligatoire de l'équipe DNS aux opérations DNSSEC

Criticité : 🟠 Élevé
Sources : NIS 2 Art.21 §3, NIST SP 800-81r2 §7.1

```
# Compétences DNSSEC requises par rôle :
# Administrateur DNS : opérations courantes (rollover ZSK, monitoring, re-signature)
# Opérateur DNSSEC : rollovers KSK, gestion HSM, procédures d'urgence
# RSSI : compréhension risques DNSSEC, validation procédures, conformité NIS 2

# Ressources de formation :
# - ICANN Learning Hub (DNSSEC) : learn.icann.org
# - RIPE NCC Training (DNSSEC) : training.ripe.net
# - DNS-OARC workshops : workshop.dns-oarc.net
# - Certifications : ICANN DNS professional, CompTIA Security+
```

Résultat attendu : Chaque membre de l'équipe DNS a suivi une formation DNSSEC certifiée dans les 12 derniers mois. Au moins 2 personnes maîtrisent les procédures de rollover KSK d'urgence.

Remédiation : Inscrire l'équipe DNS aux formations RIPE NCC DNSSEC (en ligne, gratuites). Organiser des exercices pratiques trimestriels sur l'environnement de staging.

Évaluation : ✅ ❌ ⚠️ N/A

S18-C003 — Journalisation de toutes les opérations sur les clés DNSSEC

Criticité : 🟠 Élevé
Sources : ISO 27001 A.12.4, RFC 6781 §3.1.3

```
# Configurer la journalisation détaillée des opérations DNSSEC dans BIND
grep "logging" /etc/named.conf
# Configuration recommandée :
cat << 'EOF'
logging {
    channel dnssec_log {
        file "/var/log/named/dnssec.log" versions 5 size 50m;
        print-time yes;
        print-severity yes;
        severity info;
    };
    category dnssec { dnssec_log; };
    category dnssec-keys { dnssec_log; };
};
EOF
# Pour HSM : vérifier l'audit log HSM
pkcs11-tool --list-objects 2>/dev/null | grep -i key
```

Résultat attendu : Tous les accès aux clés DNSSEC journalisés (génération, activation, désactivation, suppression, utilisation HSM). Logs conservés ≥ 12 mois. Alertes sur accès non planifiés.

Effet de bord : Les logs de signing peuvent être très volumineux sur les zones dynamiques (milliers de RRSIG re-générées). Configurer la rotation et la rétention des logs pour éviter la saturation disque.

Remédiation : Intégrer les logs DNSSEC dans le SIEM de l'organisation. Créer des règles de détection pour les accès anormaux aux clés (heure inhabituelle, volume anormal de signing).

Évaluation : ✅ ❌ ⚠️ N/A

S18-C004 — MFA obligatoire pour tout accès aux systèmes gérant les clés DNSSEC

Criticité : 🔴 Critique
Sources : NIST SP 800-81r2 §7.4, ISO 27001 A.9.4.2

```
# Vérifier que SSH vers le serveur DNS de signing exige MFA
grep "AuthenticationMethods" /etc/ssh/sshd_config
# Attendu : AuthenticationMethods publickey,keyboard-interactive
# OU utiliser un bastion avec MFA obligatoire
# Vérifier l'accès HSM (PIN + token HSM physique)
# Vérifier l'accès à l'interface registrar (2FA activé)
```

Résultat attendu : MFA obligatoire pour : accès SSH au serveur de signing, accès à l'interface web du registrar, accès au HSM (PIN + token physique), accès au CMDB/DPS.

Remédiation : Activer MFA via TOTP (Google Authenticator, Authy) ou YubiKey pour tous les accès aux systèmes DNSSEC critiques. Documenter les exceptions justifiées.

Évaluation : ✅ ❌ ⚠️ N/A

S18-C005 — Environnement de staging DNSSEC maintenu et synchronisé avec la prod

Criticité : 🟡 Moyen
Sources : RFC 6781 §7, NIST SP 800-81r2

```
# L'environnement de staging doit :
# - Répliquer exactement la configuration DNS de production
# - Avoir ses propres clés (JAMAIS copier les clés de prod en staging)
# - Avoir un resolver de test configuré avec ancre de confiance locale
# - Être utilisé pour tester TOUTE procédure avant exécution en prod

# Vérifier la synchronisation config staging/prod :
diff /etc/named-staging.conf /etc/named-prod.conf | grep -v "secrets\|key"
```

Résultat attendu : Staging opérationnel, configuration synchronisée avec la prod (sauf les clés et zones), resolver de test fonctionnel. Toute procédure DNSSEC testée en staging avant production.

Remédiation : Automatiser la synchronisation des configurations staging/prod (hors secrets). Documenter les différences intentionnelles entre staging et prod.

Évaluation : ✅ ❌ ⚠️ N/A

S18-C006 — Procédure de Change Management pour les opérations DNSSEC

Criticité : 🟡 Moyen
Sources : ISO 27001 A.12.1.2, ITIL Change Management

```
# Toute opération DNSSEC doit passer par le Change Management :
# - Rollover KSK : Change Request approuvée (CAB)
# - Rollover ZSK automatique : exception pré-approuvée (Standard Change)
# - Rollback DNSSEC : Change d'urgence (Emergency Change)
# - Modification paramètres NSEC3 : Change normal
# - Activation DNSSEC nouvelle zone : Change normal

# Critères de classification (suggérés) :
# Standard Change : rollover ZSK automatique, monitoring
# Normal Change : rollover KSK planifié, paramètres NSEC3, activation zone
# Emergency Change : incidents zone bogué, compromission de clé
```

Résultat attendu : Toutes les opérations DNSSEC planifiées sont documentées dans l'outil ITSM (Jira, ServiceNow, etc.). Les Emergency Changes sont signés a posteriori.

Remédiation : Créer des templates de Change Request spécifiques DNSSEC dans l'outil ITSM. Former l'équipe au processus de Change Management DNSSEC.

Évaluation : ✅ ❌ ⚠️ N/A

S18-C007 — Revue semestrielle de la politique DNSSEC (algorithmes, durées, outils)

Criticité : 🟡 Moyen
Sources : RFC 6781 §3.1, ANSSI REC-DNSSEC-2021

```
# Points à réviser semestriellement :
# 1. Conformité algorithmes avec la dernière version RFC 8624
# 2. Mises à jour de sécurité des logiciels DNS (BIND, Unbound, Knot)
# 3. Nouvelles vulnérabilités DNSSEC (CVE)
# 4. Évolutions réglementaires (NIS 2 actes délégués, ANSSI nouvelles rec.)
# 5. Nouvelles RFC DNSSEC (IETF DNSOP, SIDROPS)
# 6. Avancement NSEC5 et post-quantique

# Vérifier les mises à jour des logiciels DNS :
named -v  # Version BIND
unbound -V  # Version Unbound
knotd --version  # Version Knot DNS
```

Résultat attendu : Revue semestrielle documentée. Mises à jour logiciels DNS effectuées dans les 30 jours suivant une release de sécurité. Politique DNSSEC mise à jour si nécessaire.

Remédiation : Planifier la revue semestrielle dans le calendrier RSSI. Souscrire aux listes de diffusion de sécurité BIND (bind-announce@isc.org), Unbound, et Knot DNS.

Évaluation : ✅ ❌ ⚠️ N/A

S18-C008 — Gestion des secrets DNSSEC dans un coffre-fort (vault)

Criticité : 🟠 Élevé
Sources : ISO 27001 A.10.1, NIST SP 800-57

```
# Les secrets DNSSEC à stocker dans un vault sécurisé :
# - Clés privées ZSK (si stockage logiciel, pas HSM)
# - Passphrase de déchiffrement des sauvegardes de clés
# - Credentials de l'interface registrar
# - Token API du registrar
# - Credentials HSM (PIN, code de récupération)

# Vérifier l'utilisation d'un vault (HashiCorp Vault, Bitwarden Teams, etc.)
vault status 2>/dev/null || echo "HashiCorp Vault non détecté"
```

Résultat attendu : Tous les secrets DNSSEC stockés dans un vault avec contrôle d'accès. Accès audités. Rotation des secrets périodique. Sauvegarde du vault chiffrée.

Remédiation : Migrer les secrets DNSSEC vers HashiCorp Vault ou un coffre-fort d'entreprise équivalent. Configurer les politiques d'accès selon la matrice RACI.

Évaluation : ✅ ❌ ⚠️ N/A

S18-C009 — Documentation des procédures DNSSEC accessible hors-ligne (disaster recovery)

Criticité : 🟠 Élevé
Sources : ISO 22301, NIST SP 800-34

```
# Les procédures DNSSEC doivent être disponibles même si :
# - Le wiki interne est inaccessible
# - Le VPN est en panne
# - La connexion Internet est coupée
# Méthodes de disponibilité hors-ligne :
# - Impression physique du runbook DNSSEC (stocké dans le serveur room)
# - PDF chiffré sur clé USB dans le coffre physique
# - Export offline du wiki
```

Résultat attendu : Runbook DNSSEC disponible hors-ligne sous au moins 2 formes (impression + clé USB chiffrée). Mis à jour à chaque révision de procédure. Accessible 24/7 pour l'équipe d'astreinte.

Remédiation : Imprimer et plastifier le runbook DNSSEC. Stocker dans la salle serveur et au bureau du responsable DNS. Mettre à jour à chaque changement de procédure.

Évaluation : ✅ ❌ ⚠️ N/A

S18-C010 — Plan de continuité DNS incluant le composant DNSSEC

Criticité : 🟠 Élevé
Sources : ISO 22301, NIST SP 800-81r2 §9

```
# Plan de continuité DNS — aspects DNSSEC :
# Scénario 1 : Perte du serveur de signing DNSSEC
#   → Bascule vers le serveur de signing de secours (staging prêt)
#   → Durée : < 2h
# Scénario 2 : Perte du HSM
#   → Utilisation du HSM de secours (si redondé) ou des sauvegardes chiffrées
#   → Durée : < 4h
# Scénario 3 : Perte de l'accès au registrar
#   → Utilisation de l'accès API de secours
#   → Durée : < 1h (si API configurée)
# Scénario 4 : Zone bogué sur domaine critique
#   → Procédure de re-signature d'urgence
#   → En dernier recours : désactivation temporaire DNSSEC
#   → Durée : < 30 min
```

Résultat attendu : Plan de continuité DNS intégrant les scénarios DNSSEC, testé annuellement, approuvé par la direction.

Remédiation : Intégrer les scénarios DNSSEC dans le PCA/PRI de l'organisation. Tester les scénarios de reprise lors des exercices Business Continuity.

Évaluation : ✅ ❌ ⚠️ N/A

S19 — Outils et Tests DNSSEC

Contexte

L'écosystème d'outils DNSSEC est riche et permet de tester, valider et monitorer toutes les facettes du déploiement. Cette section recense les outils essentiels avec leurs cas d'usage spécifiques, les commandes pratiques, et les interprétations des résultats.

Contrôles S19

S19-C001 — DNSViz : visualisation graphique de la chaîne de confiance DNSSEC

Criticité : 🟢 Faible
Sources : DNSViz (Sandia National Laboratories)

```
# DNSViz en ligne : https://dnsviz.net
# Installation locale (optionnel) :
pip install dnsviz
# Exécution locale :
dnsviz probe <domaine> | dnsviz graph -Tsvg -o dnssec-graph.svg
# Interprétation :
# - Zones vertes = chaîne valide
# - Zones oranges = avertissements (TTL faibles, etc.)
# - Zones rouges = erreurs (BOGUS, RRSIG expirée, DS orphelin)
# - Lignes pointillées = validation non établie (insecure)
```

Résultat attendu : Graphe DNSViz entièrement vert pour toutes les zones critiques. Aucune zone rouge. Avertissements oranges documentés avec justification.

Remédiation : Pour chaque problème identifié par DNSViz : cliquer sur l'élément en rouge pour obtenir le détail de l'erreur. Corriger en suivant la procédure correspondante dans cette checklist.

Évaluation : ✅ ❌ ⚠️ N/A

S19-C002 — Zonemaster : audit complet de zone DNS et DNSSEC

Criticité : 🟡 Moyen
Sources : AFNIC / SIDN Zonemaster

```
# Zonemaster en ligne : https://zonemaster.net
# Installation CLI :
# apt install zonemaster-cli (Ubuntu/Debian)
# Exécution :
zonemaster-cli <domaine> --test all --level INFO 2>/dev/null | grep -E "ERROR|WARNING|NOTICE"
# Test DNSSEC uniquement :
zonemaster-cli <domaine> --test dnssec --level INFO
# Tests disponibles : basic, connectivity, consistency, dnssec, name_server, zone, address, syntax
```

Résultat attendu : Aucune erreur de niveau ERROR sur les tests DNSSEC. Avertissements (WARNING) documentés. Zonemaster utilisé trimestriellement comme audit complet de zone.

Remédiation : Pour chaque erreur Zonemaster : consulter la documentation du test à https://github.com/zonemaster/zonemaster-engine/blob/develop/docs/specifications/. Corriger et re-tester.

Évaluation : ✅ ❌ ⚠️ N/A

S19-C003 — delv : validation DNSSEC locale et diagnostic

Criticité : 🟠 Élevé
Sources : BIND 9 (ISC), RFC 4035

```
# delv = DNS lookup and validation (remplace dig pour les vérifications DNSSEC)
# Installation : apt install bind9-dnsutils
# Test de validation de base :
delv <domaine> A
# Sortie attendue : "; fully validated"
# Test avec trace détaillée (pour diagnostic) :
delv <domaine> A +vtrace +rtrace 2>&1
# Test avec ancre de confiance explicite :
delv <domaine> A +root=/usr/share/dns/root.ds
# Test zone bogué (doit retourner erreur) :
delv dnssec-failed.org A 2>&1 | head -5
# Tester un sous-domaine délégué :
delv <subdomain>.<domaine> A +vtrace 2>&1 | grep -E "validated|BOGUS"
```

Résultat attendu : `delv <domaine> A` retourne `; fully validated`. Aucun `BOGUS` sur les zones correctement signées.

Remédiation : Si `BOGUS` : utiliser `+vtrace` pour identifier l'étape de validation qui échoue. La trace indique précisément quel DS ou RRSIG pose problème.

Évaluation : ✅ ❌ ⚠️ N/A

S19-C004 — ldns-verify-zone : vérification cryptographique du fichier de zone

Criticité : 🟡 Moyen
Sources : ldnsutils (NLNet Labs)

```
# ldns-verify-zone vérifie la cohérence cryptographique d'un fichier de zone
# Installation : apt install ldnsutils
# Télécharger la zone (si AXFR autorisé) :
dig <domaine> AXFR @<ns-interne> > /tmp/<domaine>.zone 2>/dev/null
# Vérifier la zone :
ldns-verify-zone /tmp/<domaine>.zone
# Résultat attendu : "Zone is verified and complete."
# Vérification avec la clé publique explicite :
ldns-verify-zone -k /tmp/K<domaine>.+013+*.key /tmp/<domaine>.zone
# Test de signature individuelle :
ldns-verify-zone --verbose /tmp/<domaine>.zone 2>&1 | grep -E "ERROR|WARN|OK"
```

Résultat attendu : `Zone is verified and complete.` — toutes les signatures valides, tous les NSEC3 cohérents, aucun RR non signé manquant.

Remédiation : En cas d'erreur : identifier le RR ou la RRSIG problématique depuis la sortie verbose. Forcer une re-signature de la zone.

Évaluation : ✅ ❌ ⚠️ N/A

S19-C005 — check.sidnlabs.nl : test DNSSEC + DANE intégré

Criticité : 🟢 Faible
Sources : SIDN Labs (Pays-Bas)

```
# Test en ligne : https://check.sidnlabs.nl
# Teste simultanément :
# - Validation DNSSEC de la zone
# - Présence et validité des enregistrements TLSA (DANE)
# - Test SMTP DANE si enregistrements TLSA pour port 25

# Via API SIDN Labs :
curl -s "https://check.sidnlabs.nl/api/check?domain=<domaine>" | python3 -m json.tool
```

Résultat attendu : Score DNSSEC "pass", Score DANE "pass" (si TLSA déployé). Aucune erreur critique. Test réalisé après chaque modification DNSSEC et TLSA.

Évaluation : ✅ ❌ ⚠️ N/A

S19-C006 — RIPE Atlas : sondes géodistribuées pour test DNSSEC

Criticité : 🟡 Moyen
Sources : RIPE NCC Atlas

```
# Créer une mesure RIPE Atlas via API
curl -H "Authorization: Key <votre-api-key>" \
     -H "Content-Type: application/json" \
     -d '{
       "definitions": [{
         "type": "dns",
         "query_argument": "<domaine>",
         "query_type": "A",
         "use_probe_resolver": false,
         "set_rd_bit": true,
         "set_do_bit": true,
         "protocol": "UDP",
         "af": 4
       }],
       "probes": [{"type": "area", "value": "WW", "requested": 10}],
       "is_oneoff": true
     }' https://atlas.ripe.net/api/v2/measurements/ | python3 -m json.tool
```

Résultat attendu : 10 sondes géodistribuées testent la validation DNSSEC. Bit AD présent dans les réponses de toutes les sondes avec resolver validant.

Remédiation : En cas d'échec sur certaines sondes : analyser si le problème est géographique (NS secondaire avec problème) ou global (zone bogué).

Évaluation : ✅ ❌ ⚠️ N/A

S19-C007 — Verisign DNSSEC Analyzer : validation complète de chaîne

Criticité : 🟢 Faible
Sources : Verisign Labs

```
# DNSSEC Analyzer en ligne : https://dnssec-analyzer.verisignlabs.com
# Analyse la chaîne de confiance complète depuis la racine
# Vérifie : DNSKEY, RRSIG, DS, NSEC3, délégations

# Alternative CLI : Python dnspython
python3 << 'EOF'
import dns.resolver, dns.dnssec, dns.name, sys

domain = dns.name.from_text("<domaine>.")
print(f"Analyse DNSSEC pour {domain}")

# Vérifier DNSKEY
try:
    dnskeys = dns.resolver.resolve(domain, 'DNSKEY')
    print(f"DNSKEY : {len(list(dnskeys))} clés trouvées")
    for rdata in dnskeys:
        flag = "KSK" if rdata.flags & 0x0100 else "ZSK"
        print(f"  {flag} algo={rdata.algorithm} tag={rdata.key_tag()}")
except Exception as e:
    print(f"DNSKEY : ERREUR — {e}")

# Vérifier DS
try:
    ds_records = dns.resolver.resolve(domain, 'DS')
    print(f"DS : {len(list(ds_records))} enregistrements")
    for rdata in ds_records:
        print(f"  keytag={rdata.key_tag} algo={rdata.algorithm} dtype={rdata.digest_type}")
except Exception as e:
    print(f"DS : {e}")
EOF
```

Résultat attendu : Analyse complète sans erreurs. Résultats cohérents avec DNSViz et Zonemaster.

Évaluation : ✅ ❌ ⚠️ N/A

S19-C008 — Script de rapport d'audit DNSSEC automatisé

Criticité : 🟡 Moyen
Sources : RFC 6781, ANSSI

```
#!/usr/bin/env python3
"""Script d'audit DNSSEC automatisé — rapport complet"""
import dns.resolver, datetime, json, subprocess

DOMAINS = ["<domaine1>", "<domaine2>"]
REPORT = {"date": str(datetime.date.today()), "domains": {}}

for domain in DOMAINS:
    d = {}
    # DNSKEY
    try:
        dnskeys = dns.resolver.resolve(domain, "DNSKEY")
        d["dnskey_count"] = len(list(dnskeys))
        d["algorithms"] = list(set(rdata.algorithm for rdata in dnskeys))
    except: d["dnskey_count"] = 0

    # DS
    try:
        ds = dns.resolver.resolve(domain, "DS")
        d["ds_count"] = len(list(ds))
        d["ds_digest_types"] = list(set(rdata.digest_type for rdata in ds))
    except: d["ds_count"] = 0

    # NSEC3PARAM
    try:
        nsec3 = dns.resolver.resolve(domain, "NSEC3PARAM")
        for r in nsec3:
            d["nsec3_iterations"] = r.iterations
            d["nsec3_salt"] = r.salt.hex() if r.salt else "none"
    except: d["nsec3_iterations"] = "N/A"

    # Validation via delv
    try:
        result = subprocess.run(["delv", domain, "A"], capture_output=True, text=True, timeout=10)
        d["validation"] = "OK" if "fully validated" in result.stdout else "FAIL"
    except: d["validation"] = "ERROR"

    REPORT["domains"][domain] = d
    print(f"{domain}: DNSKEY={d.get('dnskey_count')}, DS={d.get('ds_count')}, NSEC3-
iter={d.get('nsec3_iterations')}, Valid={d.get('validation')}")

# Sauvegarder le rapport
with open(f"/tmp/dnssec-audit-{REPORT['date']}.json", "w") as f:
    json.dump(REPORT, f, indent=2)
print(f"\nRapport sauvegardé : /tmp/dnssec-audit-{REPORT['date']}.json")
```

Résultat attendu : Script d'audit exécuté mensuellement, rapport JSON archivé, résultats comparés avec le mois précédent pour détecter les régressions.

Remédiation : Intégrer dans cron mensuel. Comparer les rapports consécutifs pour détecter des changements non planifiés.

Évaluation : ✅ ❌ ⚠️ N/A

S20 — Post-Quantique : ML-DSA / CRYSTALS-Dilithium pour DNSSEC

Contexte

Les algorithmes DNSSEC actuels (ECDSA P-256, Ed25519) sont vulnérables aux attaques d'un ordinateur quantique suffisamment puissant via l'algorithme de Shor. NIST FIPS 204 (ML-DSA, Module-Lattice Digital Signature Algorithm, standardisé en 2024) définit le successeur post-quantique des signatures numériques,

basé sur les réseaux de modules (CRYSTALS-Dilithium). Des drafts IETF (draft-ietf-dnsop-dnssec-iana-cons) sont en cours pour assigner des identifiants d'algorithme DNSSEC aux algorithmes PQC.

L'horizon d'implémentation est 2027-2030 selon la maturité des implémentations logicielles BIND/Knot/PowerDNS. En 2026, la préparation consiste à : surveiller les drafts IETF, tester les implémentations expérimentales en lab, évaluer l'impact de la taille des clés/signatures PQC sur l'infrastructure DNS, et planifier la migration.

Contrôles S20

S20-C001 — Suivi du statut des drafts IETF DNSSEC post-quantique

Criticité : 🟢 Faible
Sources : draft-ietf-dnsop-dnssec-iana-cons, IETF DNSOP WG

```
# Drafts IETF pertinents à suivre en 2026 :
# - draft-ietf-dnsop-dnssec-iana-cons : assignation algo IDs pour PQC
# - draft-ietf-dnsop-privatedns : usage de SLH-DSA (SPHINCS+) en DNSSEC
# - draft-ietf-dnsop-dnssec-automation : automatisation multi-signer

# Vérifier l'état d'avancement :
# https://datatracker.ietf.org/wg/dnsop/documents/
# https://datatracker.ietf.org/doc/draft-ietf-dnsop-dnssec-iana-cons/

# Implémentations expérimentales en 2026 :
# - BIND 9.20-dev : support partiel ML-DSA (branche experimentale)
# - OQS-Provider (liboqs) : fournisseur PKCS#11 PQC pour HSM
```

Résultat attendu : Veille active sur les drafts IETF DNSSEC PQC. Rapport de veille semestriel intégré dans la revue DNSSEC. Plan de migration PQC documenté à horizon 2028-2030.

Effet de bord : Un déploiement prématuré d'algorithmes PQC non standardisés (avant RFC définitive) peut provoquer des incompatibilités avec les resolvers et registrars qui ne les supportent pas encore.

Remédiation : S'abonner à la liste de diffusion IETF DNSOP (dnsop@ietf.org). Participer aux réunions IETF virtuelles DNSOP. Documenter la veille PQC dans la revue semestrielle DNSSEC.

Évaluation : ✅ ❌ ⚠️ N/A

S20-C002 — Évaluation de l'impact de la taille des clés ML-DSA sur l'infrastructure DNS

Criticité : 🟡 Moyen
Sources : NIST FIPS 204, RFC 8906

```
# Tailles de clés ML-DSA vs algorithmes actuels :
# Ed25519 : clé publique 32 bytes, signature 64 bytes
# ECDSA P-256 : clé publique 64 bytes, signature 64 bytes
# ML-DSA-44 (niveau 2 = NIST security level II) : clé 1312 bytes, sig 2420 bytes
# ML-DSA-65 (niveau 3) : clé 1952 bytes, sig 3309 bytes
# ML-DSA-87 (niveau 5) : clé 2592 bytes, sig 4627 bytes

# Calcul d'impact sur les réponses DNS :
# Réponse DNSKEY ML-DSA-44 : ~2000 bytes (vs ~200 bytes Ed25519)
# Seuil fragmentation UDP IPv6 : 1232 bytes (MTU 1280)
# → TCP OBLIGATOIRE pour toutes les réponses DNSKEY PQC

echo "Impact estimé ML-DSA-44 sur DNS :"
echo "Clé publique : 1312 bytes (vs 32 pour Ed25519 = 41× plus grande)"
echo "Signature : 2420 bytes (vs 64 pour Ed25519 = 38× plus grande)"
echo "→ TOUTES les réponses DNSSEC dépasseront 1232 bytes"
echo "→ TCP/53 et DoT obligatoires"
echo "→ MTU Path Discovery DNS obligatoire"
```

Résultat attendu : Analyse d'impact documentée. Évaluation de la capacité de l'infrastructure DNS à gérer des réponses TCP systématiques (augmentation latence, charge TCP). Plan d'adaptation identifié (upgrade matériel/bande passante si nécessaire).

Effet de bord : ML-DSA génère des signatures 38× plus grandes qu'Ed25519. Cela impacte : taille des réponses DNS (→ TCP obligatoire), performance de validation des resolvers (calcul plus complexe), charge mémoire des caches DNS. Le coût computationnel de vérification ML-DSA est ~100× Ed25519.

Remédiation : Préparer l'infrastructure TCP/53 et DoT (port 853) pour un trafic accru lors de la migration PQC. Évaluer l'upgrade des resolvers et NS autoritatifs.

Évaluation : ✅ ❌ ⚠️ N/A

S20-C003 — Stratégie de signature hybride (classique + PQC) pendant la transition

Criticité : 🟢 Faible
Sources : NIST SP 800-208, ANSSI PQC 2024

```
# Approche hybride : co-signer la zone avec algo classique (Ed25519) + PQC (ML-DSA)
# Avantage : protection dès aujourd'hui (Ed25519) + protection PQC future
# Inconvénient : double la taille de toutes les réponses DNSSEC

# Analogie avec ECDSA+RSA en DNSSEC :
# On peut publier plusieurs DNSKEY avec des algorithmes différents
# Chaque DNSKEY génère ses propres RRSIG
# Le resolver utilise l'algo qu'il supporte

# Commande pour ajouter une ZSK PQC (quand supporté) :
# dnssec-keygen -a ML-DSA-44 <domaine>  (hypothétique, non supporté en 2026)
echo "La co-signature hybride Ed25519 + ML-DSA sera la méthode recommandée"
echo "pour la phase de transition. Horizon estimé : 2027-2029."
```

Résultat attendu : Stratégie de migration hybride documentée dans le DPS, prête à être implémentée dès que le support ML-DSA est disponible dans BIND, Knot, ou PowerDNS en version stable.

Remédiation : Tester la co-signature hybride sur l'environnement de staging dès qu'une implémentation expérimentale stable est disponible.

Évaluation : ✅ ❌ ⚠️ N/A

S20-C004 — Évaluation de SLH-DSA (SPHINCS+, NIST FIPS 205) pour DNSSEC

Criticité : 🟢 Faible
Sources : NIST FIPS 205, draft-ietf-dnsop-privatedns

```
# SLH-DSA = Stateless Hash-Based Digital Signature Scheme
# Alternative à ML-DSA basée sur des fonctions de hachage (pas de réseaux)
# Avantage : sécurité conservative (basée uniquement sur la résistance du hash)
# Inconvénient : signatures encore plus grandes (6500-50000 bytes selon niveau)
# → Non adapté aux réponses UDP DNS
# → Possible usage pour les KSK (offline, pas de contrainte UDP)

# Comparaison pour DNSSEC :
echo "SLH-DSA-SHA2-128s (niveau 1) : sig 7856 bytes"
echo "SLH-DSA-SHA2-256s (niveau 5) : sig 29792 bytes"
echo "→ TCP-only, usage KSK offline uniquement envisageable"
```

Résultat attendu : Évaluation de SLH-DSA documentée. Conclusion : ML-DSA préférable pour les ZSK (signature plus courte). SLH-DSA éventuellement pour les KSK offline (usage rare, pas de contrainte UDP).

Évaluation : ✅ ❌ ⚠️ N/A

S20-C005 — Test en lab de ML-DSA avec liboqs et OQS-Provider

Criticité : 🟢 Faible
Sources : Open Quantum Safe (OQS), liboqs

```
# Installation liboqs et OQS-Provider pour tests PQC
# (environnement de lab uniquement)
git clone https://github.com/open-quantum-safe/liboqs.git
# Compilation et test :
# mkdir build && cd build && cmake .. && make && make test
# OQS-Provider pour OpenSSL (permet de tester les algos PQC)
# Test de génération de clé ML-DSA :
# openssl genpkey -algorithm ml-dsa-65 -out ml-dsa-key.pem (avec OQS-Provider)
# Test de signature :
# openssl dgst -sign ml-dsa-key.pem -out sig.bin message.bin
echo "Tests PQC en lab documentés — résultats archivés dans le rapport de veille"
```

Résultat attendu : Tests PQC réalisés en lab isolé, rapport de résultats documenté (performance, taille des sorties, compatibilité). Résultats partagés avec l'équipe DNS pour préparation.

Remédiation : Allouer 2-3 jours/an à des tests PQC en lab. Documenter les résultats dans la revue semestrielle DNSSEC.

Évaluation : ✅ ❌ ⚠️ N/A

S20-C006 — Plan de migration PQC DNSSEC (horizon 2027-2030)

Criticité : 🟢 Faible
Sources : NIST PQC Standards, ANSSI PQC 2024, IETF DNSOP

```
# Plan de migration PQC DNSSEC recommandé :

# 2026 : Préparation
#   → Veille IETF DNSOP, suivi drafts
#   → Tests PQC en lab (liboqs, OQS-Provider)
#   → Évaluation infrastructure TCP/DoT pour réponses volumineuses
#   → Documentation stratégie migration dans DPS

# 2027 : Pilote
#   → Déploiement expérimental sur une zone de test (si BIND stable)
#   → Co-signature hybride Ed25519 + ML-DSA sur zone pilote
#   → Mesure de l'impact performance et réseau

# 2028-2029 : Migration progressive
#   → Déploiement co-signature hybride sur zones non critiques
#   → Tests de compatibilité avec resolvers principaux
#   → Mise à jour des registrars et chaînes de confiance

# 2030 : ML-DSA seul (ou hybride si compatibilité non garantie)
#   → Retrait progressif des algorithmes classiques
#   → Selon l'état d'avancement des menaces quantiques
```

Résultat attendu : Plan de migration PQC DNSSEC documenté dans le DPS, approuvé par le RSSI, révisé annuellement selon l'évolution des standards.

Remédiation : Intégrer la migration PQC dans la roadmap DNS à 5 ans de l'organisation. Budget alloué pour les tests et la formation.

Évaluation : ✅ ❌ ⚠️ N/A

S20-C007 — Évaluation de la maturité des HSM pour ML-DSA

Criticité : 🟢 Faible
Sources : NIST FIPS 204, FIPS 140-3

```
# Les HSM actuels certifiés FIPS 140-2/3 ne supportent pas ML-DSA en 2026
# Horizon de support HSM ML-DSA : 2026-2027 (firmware updates pour certains)
# Vendeurs à surveiller :
# - Thales Luna HSM : roadmap PQC annoncée
# - AWS CloudHSM : support PQC en cours
# - YubiHSM 2 : ML-DSA pas encore supporté (2026)
# - Entrust nShield : roadmap PQC Q3 2026

echo "Support ML-DSA dans les HSM : planifié 2026-2027 selon vendeur"
echo "Vérifier la roadmap PQC du vendeur de HSM utilisé"
```

Résultat attendu : Évaluation de la roadmap PQC du HSM actuel. Plan de migration ou mise à jour firmware HSM documenté si le vendeur annonce le support ML-DSA.

Remédiation : Contacter le vendeur du HSM pour obtenir la roadmap de support PQC. Budgéter si remplacement/upgrade HSM nécessaire.

Évaluation : ✅ ❌ ⚠️ N/A

S20-C008 — Intégration PQC dans la revue semestrielle DNSSEC

Criticité : 🟢 Faible
Sources : NIST, ANSSI PQC 2024, IETF

Résultat attendu : Chaque revue semestrielle DNSSEC inclut une section "État de l'art PQC DNSSEC" couvrant : avancement des drafts IETF, nouvelles implémentations disponibles, nouvelles publications de recherche sur les vulnérabilités quantiques des algorithmes actuels, mise à jour du plan de migration.

Remédiation : Ajouter une section PQC au template de revue semestrielle DNSSEC. Désigner un responsable de veille PQC dans l'équipe.

Évaluation : ✅ ❌ ⚠️ N/A

Matrices de Conformité

Matrice RFC 8624 (Algorithmes DNSSEC)

Algo	Nom	MUST (signer)	MUST (valider)	Statut 2026
5	RSASHA1	MUST NOT	MUST NOT	Interdit
7	RSASHA1-NSEC3	NOT REC	NOT REC	Déconseillé
8	RSA/SHA-256	MAY	MUST	Acceptable (≥2048b)
10	RSA/SHA-512	NOT REC	NOT REC	Déconseillé
13	ECDSA P-256	**MUST**	**MUST**	✅ Standard
14	ECDSA P-384	MAY	MUST	Gouvernement
15	Ed25519	**RECOMMENDED**	SHOULD	✅ Recommandé
16	Ed448	MAY	SHOULD	Optionnel

Matrice de Conformité Réglementaire

Contrôle	NIS 2 Art.21	ANSSI REC	DORA Art.8	ISO 27001
DNSSEC signing	✅ Requis	R1	Implicite	A.12.4
Algo RFC 8624	✅	R2	✅	A.10.1
KSK HSM	✅	R7	✅	A.10.1
Monitoring RRSIG	✅	R8	✅	A.12.4
DPS	✅	R9	✅	A.5.1
Resolver validant	✅	R10	✅	A.12.4
NSEC3 iter=0	✅	R6 (RFC 9276)	✅	A.10.1
Audit annuel	✅	✅	✅	A.12.7

Matrice RTO par Type d'Incident

Incident	Gravité	RTO Cible	Action
RRSIG expirée	P0	30 min	re-signe zone
Zone bogué (DS orphelin)	P0	2h	retrait DS + DNSKEY
ZSK compromise	P0	1h	rollover ZSK urgence
KSK compromise	P0	4h	rollover KSK + registrar
Resolver validant down	P1	15 min	bascule resolver
TLSA/cert incohérent	P1	1h	màj TLSA
DS non propagé	P2	48h	attente propagation
HSM inaccessible	P1	4h	activation HSM secours

Glossaire DNSSEC

Terme	Définition
AD bit	Authenticated Data : flag DNS indiquant que le resolver a validé DNSSEC
BOGUS	Zone DNSSEC dont la validation a échoué (signatures invalides, DS orphelin...)
CDS	Child DS : enregistrement DNS permettant au parent de récupérer automatiquement le DS
CDNSKEY	Child DNSKEY : similar à CDS mais contient la DNSKEY complète
CSK	Combined Signing Key : clé unique remplissant les rôles KSK et ZSK
DANE	DNS-Based Authentication of Named Entities : utilise DNSSEC pour authentifier les certificats TLS
DNSKEY	Enregistrement DNS contenant la clé publique de signing DNSSEC
DoH	DNS over HTTPS : chiffrement du transport DNS via HTTPS (RFC 8484)
DoQ	DNS over QUIC : chiffrement du transport DNS via QUIC (RFC 9250)

Terme	Définition
DoT	DNS over TLS : chiffrement du transport DNS via TLS (RFC 7858)
DPS	DNSSEC Practice Statement : politique de gestion des clés DNSSEC
DS	Delegation Signer : empreinte de la KSK, publiée dans la zone parente
EDNS0	Extension Mechanisms for DNS : permet des réponses DNS de plus de 512 bytes
HSM	Hardware Security Module : module matériel de sécurité pour le stockage des clés
INSECURE	Zone DNS parente signée DNSSEC mais dont la zone fille ne l'est pas
KSK	Key Signing Key : clé qui signe uniquement les DNSKEY, son empreinte = DS
ML-DSA	Module-Lattice Digital Signature Algorithm : algorithme PQC (NIST FIPS 204)
NSEC	Next Secure : enregistrement prouvant la non-existence d'un nom dans une zone
NSEC3	Next Secure 3 : NSEC avec hash des noms pour éviter le zone-walking
NSEC5	Proposition de NSEC avec VRF post-quantique safe (draft IETF)
RPKI	Resource Public Key Infrastructure : sécurise les annonces BGP des préfixes IP
RRSIG	Resource Record Signature : signature DNSSEC d'un ensemble de RR
SOA	Start of Authority : enregistrement DNS principal d'une zone
TLSA	TLSA Record : enregistrement DNS pour DANE contenant l'empreinte du certificat TLS
TTL	Time To Live : durée de mise en cache d'un enregistrement DNS
ZSK	Zone Signing Key : clé qui signe tous les enregistrements de la zone (sauf DNSKEY)

Sources et Références Normatives

RFCs Fondamentales DNSSEC

RFC	Titre	Pertinence
RFC 4033	DNS Security Introduction and Requirements	Base DNSSEC
RFC 4034	Resource Records for DNS Security Extensions	DNSKEY, RRSIG, DS, NSEC
RFC 4035	Protocol Modifications for DNS Security Extensions	Validation protocol
RFC 5155	DNS Security (DNSSEC) Hashed Authenticated Denial of Existence	NSEC3
RFC 5011	Automated Updates of DNS Security (DNSSEC) Trust Anchors	Rollover ancre confiance
RFC 6781	DNSSEC Operational Practices, Version 2	Guide opérationnel
RFC 6840	Clarifications and Implementation Notes for DNS Security	Précisions pratiques
RFC 7344	Automating DNSSEC Delegation Trust Maintenance	CDS/CDNSKEY
RFC 7583	DNSSEC Key Rollover Timing Considerations	Timing rollovers
RFC 7671	The DANE Protocol	DANE optimisé
RFC 7672	SMTP Security via Opportunistic DANE TLS	SMTP DANE

RFC	Titre	Pertinence
RFC 7958	DNSSEC Trust Anchor Publication for the Root Zone	Ancre root
RFC 8078	Managing DS Records from the Parent via CDS/CDNSKEY	CDS automatisation
RFC 8080	Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC	Ed25519/Ed448
RFC 8205	BGPsec Protocol Specification	BGPsec
RFC 8484	DNS Queries over HTTPS (DoH)	DoH
RFC 8624	Algorithm Implementation Requirements and Usage Guidance for DNSSEC	Algorithmes 2026
RFC 8901	Multi-Signer DNSSEC Models	Multi-signer
RFC 8906	A Common Operational Problem in DNS Servers	TCP fallback
RFC 9250	DNS over Dedicated QUIC Connections	DoQ
RFC 9276	Guidance for NSEC3 Parameter Settings	NSEC3 iterations=0
RFC 9364	DNSSEC Roadmap	Vue d'ensemble

Normes et Recommandations

Référence	Organisme	Contenu
NIST SP 800-81r2	NIST	Secure DNS Deployment Guide
NIST FIPS 204	NIST	ML-DSA (Module-Lattice Digital Signature)
NIST FIPS 205	NIST	SLH-DSA (Stateless Hash-Based Signatures)
ANSSI REC-DNSSEC-2021	ANSSI	Recommandations DNSSEC pour la France
ANSSI PQC 2024	ANSSI	Migration post-quantique : recommandations
NIS 2 Art.21	UE	Obligations sécurité DNS pour entités NIS 2
DORA Art.8	UE	Résilience opérationnelle DNS secteur financier
AFNIC DPS	AFNIC	DNSSEC Practice Statement TLD .fr
ICANN DNSSEC Guide	ICANN	Guide déploiement DNSSEC global
CIS Controls v8	CIS	Contrôle 9 (DNS Filtering)
