AYI NEDJIMI Consultants

Attaques sur Metadata Services Cloud : SSRF et IMDS

Catégorie : Cloud Security Lecture : 8 min Publié le : 10/03/2026 Auteur : Ayi NEDJIMI

Analyse des attaques SSRF ciblant les metadata services cloud IMDS : techniques d'exploitation AWS, Azure et GCP, protections IMDSv2 et détection.

Résumé exécutif

Les attaques SSRF ciblant les metadata services cloud (IMDS) restent l'un des vecteurs de compromission les plus exploités. Cette analyse technique couvre les techniques d'exploitation sur AWS, Azure et GCP, les protections IMDSv2 et les stratégies de détection.
Le metadata service est le talon d'Achille des instances cloud. Accessible à l'adresse magique 169.254.169.254 depuis toute instance EC2, VM Azure ou Compute Engine GCP, il fournit des informations sensibles incluant les credentials temporaires des rôles IAM associés à l'instance. Une vulnérabilité SSRF (Server-Side Request Forgery) dans n'importe quelle application hébergée sur l'instance permet à un attaquant distant de requêter ce metadata service et d'exfiltrer ces credentials pour pivoter vers l'ensemble de l'infrastructure cloud. L'affaire Capital One en 2019, qui a exposé les données de 106 millions de clients via une SSRF exploitant le metadata service AWS, a mis en lumière ce vecteur d'attaque, mais six ans plus tard, des millions d'instances cloud restent vulnérables. Ce guide technique analyse les mécanismes d'exploitation sur chaque provider, les protections natives disponibles et les stratégies de détection avancées pour cette catégorie de menaces persistantes et particulièrement dévastatrices en termes d'impact.

Comment fonctionne le metadata service IMDS ?

L'*Instance Metadata Service* (IMDS) est un service HTTP local accessible à l'adresse `http://169.254.169.254` depuis chaque instance cloud. Il fournit des métadonnées sur l'instance : identifiant, type, région, réseau, et surtout les **credentials IAM temporaires** du rôle associé. Sur AWS, le chemin `/latest/meta-data/iam/security-credentials/{role-name}` retourne un AccessKeyId, SecretAccessKey et Token valides pendant plusieurs heures. Sur Azure, le chemin `/metadata/identity/oauth2/token` retourne un access token pour la Managed Identity. Sur GCP, `/computeMetadata/v1/instance/service-accounts/default/token` retourne un token OAuth2.

Le risque est direct : toute application capable de faire des requêtes HTTP internes (proxy, webhook, image fetcher, PDF generator, API avec paramètre URL) est potentiellement un vecteur SSRF vers l'IMDS. Les techniques d'escalade de privilèges IAM détaillées dans escalades de privilèges AWS commencent souvent par une exfiltration de credentials via l'IMDS.

Provider	IMDS URL	Protection native	Credentials type
AWS	169.254.169.254	IMDSv2 (token PUT)	STS temporary credentials
Azure	169.254.169.254	Header Metadata:true	Managed Identity token
GCP	metadata.google.internal	Header Metadata-Flavor	OAuth2 access token
AWS EKS	169.254.169.254	Pod Identity / IRSA	Web Identity token
GCP GKE	metadata.google.internal	Workload Identity	Federated token

Mon avis : IMDSv2 d'AWS est la meilleure protection native mais son adoption reste trop lente. Beaucoup d'organisations laissent IMDSv1 activé pour ne pas casser la compatibilité avec des applications anciennes. C'est inacceptable en 2026. Chaque instance qui supporte encore IMDSv1 est une cible SSRF exploitable en quelques secondes par un attaquant motivé.

Quelles techniques SSRF exploitent l'IMDS ?

Les techniques SSRF exploitant l'IMDS varient en sophistication. La **SSRF directe** via un paramètre URL non validé est la plus simple : `?url=http://169.254.169.254/latest/meta-data/`. La **SSRF via redirection** contourne les filtres basiques : l'attaquant contrôle un serveur qui redirige (HTTP 301/302) vers l'IMDS. La **SSRF via DNS rebinding** exploite un domaine qui résout alternativement vers une IP publique puis vers 169.254.169.254. La **SSRF via parsing de protocoles** utilise des schémas alternatifs : `gopher://`, `dict://`, ou des encodages URL comme `http://169.254.169.254` encodé en decimal `http://2852039166/`.

Sur **IMDSv1 AWS**, une simple requête GET suffit pour exfiltrer les credentials. **IMDSv2** impose une requête PUT préalable pour obtenir un token, avec un header `X-aws-ec2-metadata-token-ttl-seconds`. Ce token doit ensuite être inclus dans les requêtes GET suivantes. Cette protection bloque la majorité des SSRF simples car les applications vulnérables ne peuvent généralement pas effectuer de requêtes PUT avec des headers custom. Cependant, certaines SSRF avancées (via cURL avec options custom ou via des proxies HTTP complets) peuvent contourner IMDSv2. Les techniques d'exploitation GCP sont détaillées dans sécurité offensive GCP.

Consultez les ressources officielles d'AWS Security et de GCP Security pour les configurations de sécurité recommandées contre les attaques SSRF sur chaque provider.

Comment configurer IMDSv2 sur AWS ?

La migration vers **IMDSv2** se fait en deux étapes. D'abord, passez l'instance en mode `HttpTokens: required` qui bloque les requêtes IMDSv1. Vérifiez au préalable que toutes les applications et agents sur l'instance supportent IMDSv2 : les AWS SDKs récents (post-2019) le supportent nativement, mais les scripts custom utilisant `curl` directement doivent être mis à jour. Configurez `HttpPutResponseHopLimit: 1` pour empêcher les requêtes IMDS depuis les conteneurs Docker sur l'instance (le hop limit 1 empêche le trafic traversant un réseau additionnel).

Au niveau organisationnel, utilisez une **SCP** pour interdire le lancement d'instances avec IMDSv1 : `"Condition": {{"StringNotEquals": {{"ec2:MetadataHttpTokens": "required"}}}}` avec un Deny sur `ec2:RunInstances`. Pour le parc existant, utilisez **AWS Config** avec la règle `ec2-imdsv2-check` pour détecter les instances non conformes et une remediation action SSM pour forcer la migration. L'audit IaC via audit Terraform compliance garantit que les nouvelles instances sont déployées en IMDSv2 par défaut dans vos templates Terraform.

Lors d'un pentest pour un éditeur SaaS, nous avons exploité une SSRF dans un service de génération de PDF qui acceptait des URLs d'images. En envoyant `http://169.254.169.254/latest/meta-data/iam/security-credentials/prod-api-role` comme URL d'image, le service a requêté l'IMDS et inclus les credentials IAM dans le message d'erreur (l'image n'étant pas une image valide). Ces credentials donnaient accès en lecture/écriture à tous les buckets S3 du compte de production. IMDSv2 aurait bloqué cette attaque car le service de génération de PDF ne supporte que les requêtes GET.

Quelles protections applicatives déployer ?

Au-delà d'IMDSv2, plusieurs protections applicatives réduisent le risque SSRF. Le **filtrage d'URL en whitelist** bloque les requêtes vers les plages d'adresses internes (169.254.0.0/16, 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16). Les **WAF rules** détectent les tentatives SSRF dans les paramètres HTTP. Les **Security Groups** peuvent bloquer le trafic vers l'IMDS au niveau réseau (bien que cela casse certaines fonctionnalités AWS). Sur **Kubernetes**, les Network Policies peuvent bloquer l'accès à 169.254.169.254 depuis les pods applicatifs, et **IRSA** (IAM Roles for Service Accounts) élimine le besoin d'IMDS pour les pods EKS.

La protection réseau évoquée dans segmentation réseau VLAN firewall complète ces défenses en limitant le trafic des instances vers les plages d'adresses internes non nécessaires. Pour la détection des tentatives SSRF, les logs CloudTrail montrent l'utilisation des credentials IMDS avec l'user agent et le source IP, permettant de détecter les exfiltrations — consultez escalade de privilèges IAM cloud pour les techniques de détection IAM.

À retenir : La protection contre les attaques SSRF ciblant l'IMDS repose sur trois couches complémentaires : IMDSv2 obligatoire sur toutes les instances (protection infrastructure), validation stricte des URLs dans le code applicatif (protection application), et monitoring des accès IMDS anormaux dans les logs CloudTrail (détection). Aucune couche seule ne suffit car les techniques de contournement évoluent constamment.

Peut-on détecter les attaques IMDS en temps réel ?

La détection repose sur l'analyse des logs **CloudTrail** et des **VPC Flow Logs**. Dans CloudTrail, surveillez les événements `sts:GetCallerIdentity` et les actions IAM effectuées avec des credentials temporaires de rôles d'instance depuis des adresses IP externes à votre VPC — c'est un indicateur fort d'exfiltration de credentials IMDS. **GuardDuty** détecte nativement ce pattern via le finding type `UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS` et `...OutsideAWS`. Dans les VPC Flow Logs, des volumes de requêtes anormaux vers 169.254.169.254 depuis une instance spécifique peuvent indiquer un scan automatisé.

Les techniques avancées de contournement d'IMDSv2 évoluent constamment et nécessitent une veille active. Les contournements documentés incluent : l'exploitation de **proxies HTTP complets** (comme Squid ou HAProxy) qui supportent les requêtes PUT avec headers custom, permettant d'obtenir le token IMDSv2 via le proxy puis de l'utiliser pour les requêtes GET suivantes. L'exploitation de **bibliothèques HTTP configurables** dans les applications vulnérables (cURL avec options avancées, requests Python avec session persistante) qui peuvent être manipulées pour effectuer la séquence PUT puis GET requise par IMDSv2. Le **container escape** suivi d'un accès IMDS depuis le host, où le hop limit de 1 ne s'applique plus car le trafic ne traverse plus le réseau Docker bridge.

Pour contrer ces techniques avancées, les protections additionnelles incluent : le **blocage réseau complet** de l'accès au metadata service via des iptables rules sur l'instance ou des Network Policies Kubernetes pour les pods qui n'ont aucun besoin légitime d'accéder à l'IMDS, l'utilisation exclusive d'**IRSA** (IAM Roles for Service Accounts) sur EKS qui fournit des credentials via le token projector Kubernetes sans passer par l'IMDS, et le monitoring des accès IMDS via des scripts custom qui loggent chaque requête au metadata service avec le PID et le user du processus appelant pour la forensique.

Êtes-vous certain que toutes vos instances EC2, VMs Azure et instances GCP Compute Engine ont le metadata service version 2 activé et que la version 1 est complètement désactivée ?

Comment tester les protections IMDS dans votre environnement ?

Le test des protections IMDS doit faire partie de vos exercices de sécurité réguliers. Commencez par un **audit de configuration** : utilisez AWS Config avec la règle `ec2-imdsv2-check` pour identifier toutes les instances qui supportent encore IMDSv1. Sur Azure, vérifiez les metadata service headers obligatoires avec Azure Policy. Sur GCP, confirmez que le header `Metadata-Flavor: Google` est requis pour tous les accès au metadata service. Documentez chaque exception avec une justification business et une date de remédiation planifiée.

Ensuite, conduisez des **tests de pénétration ciblés** sur vos applications web pour vérifier la résistance aux SSRF. Testez les paramètres d'URL de chaque endpoint qui accepte des URLs : tentez d'accéder à http://169.254.169.254, aux variantes encodées (hex, decimal, octal), aux redirections via des domaines que vous contrôlez, et au DNS rebinding. Pour les applications conteneurisées, vérifiez que les Network Policies Kubernetes ou les iptables rules bloquent effectivement le trafic vers le metadata service depuis les pods applicatifs. Documentez les résultats et corrigez immédiatement toute SSRF découverte.

Enfin, testez la **détection** : simulez une exfiltration de credentials IMDS et vérifiez que GuardDuty génère le finding attendu dans le délai configuré, que l'alerte est correctement routée vers le SOC, et que le playbook de réponse automatique fonctionne (révocation des credentials, notification de l'équipe, création d'un incident). Ces tests end-to-end valident l'ensemble de la chaîne de protection, de la prévention à la détection et à la réponse, garantissant que vos protections IMDS ne sont pas simplement configurées mais réellement opérationnelles et efficaces contre les techniques d'attaque actuelles.

Les attaques SSRF ciblant l IMDS restent parmi les vecteurs de compromission cloud les plus exploites malgre la disponibilite de protections natives depuis 2019. Cette persistance s explique par la dette technique des applications anciennes non mises a jour et par la meconnaissance des equipes de developpement qui ne percoivent pas le metadata service comme un risque. La sensibilisation des developpeurs au risque SSRF et IMDS doit faire partie integrante de votre programme de formation securite applicative car c est un vecteur qui mene directement a la compromission complete de l infrastructure cloud avec un impact catastrophique potentiel.

Sources et références : CISA · Cloud Security Alliance

Conclusion : plan de protection IMDS

Protégez-vous contre les attaques IMDS en quatre étapes. Étape 1 : auditez toutes les instances existantes et identifiez celles qui utilisent encore IMDSv1 (AWS Config rule ec2-imdsv2-check). Étape 2 : migrez vers IMDSv2 obligatoire avec hop limit 1 sur toutes les instances, en commençant par la production. Étape 3 : déployez des protections applicatives (filtrage d'URL, WAF rules) et Kubernetes (Network Policies, IRSA/Workload Identity). Étape 4 : configurez la détection GuardDuty et les alertes CloudTrail pour les exfiltrations de credentials IMDS. Cette approche multicouche réduit drastiquement le risque tout en maintenant la fonctionnalité du metadata service pour les cas d'usage légitimes. La mise en conformité IMDSv2 doit être priorisée comme un chantier de sécurité à part entière avec un sponsor exécutif, un planning de migration par lots d'instances, des tests de non-régression applicative, et un suivi hebdomadaire de l'avancement via les dashboards AWS Config. Les équipes de développement doivent être formées aux bonnes pratiques d'utilisation de l'IMDS et aux alternatives comme les variables d'environnement ECS task definitions ou les projections de tokens Kubernetes IRSA qui éliminent complètement la dépendance au metadata service pour l'obtention de credentials cloud temporaires dans les architectures conteneurisées modernes déployées sur les principaux orchestrateurs cloud managés en environnement de production cloud sécurisé.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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