Java4Shell : un correctif publié n'est pas un correctif reçu

Le 17 novembre 2026, une faille du décodeur Base64 d'OpenJDK (la version libre de Java) a été rendue publique, et Oracle a publié le jour même des versions corrigées. Le récit a été celui d'une course classique entre une faille critique et son correctif. Il a moins montré que, pour Jira, Jenkins ou Keycloak, qui tournent souvent sur un Java qu'ils embarquent eux-mêmes, le correctif d'Oracle ne change rien tant que l'éditeur du produit ne l'a pas intégré. L'épisode rend visible ce que Susan Leigh Star et Karen Ruhleder appelaient l'infrastructure : ce qui reste invisible tant que cela fonctionne, et ne se révèle qu'au moment de la panne.

Par Alexandre Berge · 23 novembre 2026 · 12 min de lecture · Chronique

Une faille Java, des correctifs le soir même

À la date du 21 novembre 2026, date d'arrêt des faits de cette chronique, la chronologie est la suivante. Le 17 novembre, la société américaine de recherche en sécurité Calif divulgue une vulnérabilité qu'elle nomme « Java4Shell », enregistrée sous l'identifiant CVE-2026-114938. Selon Next, elle touche toutes les versions d'OpenJDK depuis la 17.0.6. Elle se loge dans une variante accélérée du décodeur Base64 (le format qui transforme des données binaires en texte, utilisé pour les identifiants et les jetons de session), active sur les processeurs x86-64 dotés des instructions AVX-512 VBMI. Ce décodage intervenant souvent avant toute vérification d'identité, la faille permet d'exécuter du code à distance sans authentification.

Selon The Register, des informations avaient fuité avant la divulgation coordonnée. Oracle publie le jour même les versions corrigées 17.0.21.1, 21.0.13.1, 25.0.5.1 et 27.0.1.1. Le même jour, l'agence américaine de cybersécurité CISA appelle à corriger immédiatement. Le 18 novembre, le CERT-FR (centre gouvernemental de veille, d'alerte et de réponse aux attaques informatiques) émet l'alerte CERTFR-2026-ALE-016. Elle cite, parmi les produits qui utilisent OpenJDK, Keycloak, Jira, Confluence, JFrog Artifactory, Jenkins et Elasticsearch.

Oracle cote la gravité à 8,1 sur 10, Calif à 9,2 et Red Hat à 9,8, selon des versions différentes de l'échelle CVSS (le barème standard de gravité des vulnérabilités). Calif affirme, selon Next, avoir exécuté du code sans identifiant sur Keycloak, Jira, Confluence, Artifactory, Jenkins et PingFederate, chaînes d'exploitation qu'elle garde privées. Le 18 novembre, The Register citait l'équipe Talos de Cisco, qui n'avait pas encore observé d'exploitation en conditions réelles.

Le récit dominant tient en deux temps : une faille sérieuse, découverte selon Calif avec l'aide de l'intelligence artificielle pour 200 dollars de calcul, et un correctif disponible le jour même. Il laisse dans l'ombre la question qui se pose à une direction le 18 novembre au matin : **où ce composant tourne-t-il chez elle, et qui décide du moment où il sera corrigé ?**

Le correctif atteint Java, pas les produits qui l'embarquent

Next le résume : toutes les distributions dérivées d'OpenJDK sont concernées, « comme les logiciels qui embarquent leur propre JVM » (machine virtuelle Java). Le Java qui tourne n'est alors pas celui que les équipes système mettent à jour. Il arrive dans le paquet, l'image de conteneur ou l'installateur de l'éditeur, comme le rappelle le projet Jenkins le 18 novembre pour ses propres images.

Le cas de Keycloak, logiciel libre de gestion des identités, rend la chaîne visible. Dans son avis du 18 novembre, le projet indique que ses images étiquetées 26.8.2-1 et 26.8.2-2 contenaient un OpenJDK 21.0.13 vulnérable. Le 19 novembre, il annonce des images reconstruites sous l'étiquette 26.8.2-3, et met en garde : « Do not use the Keycloak version number alone to determine whether a deployment is fixed » (ne pas se fier au seul numéro de version de Keycloak pour savoir si un déploiement est corrigé).

Le tableau que Red Hat tient pour cette faille, consulté le 21 novembre, montre ce décalage à l'intérieur d'un seul éditeur. Red Hat a publié sa propre distribution d'OpenJDK corrigée le 18 novembre, puis les paquets Java de Red Hat Enterprise Linux 8, 9 et 10 le 19. Le 21 novembre, quatorze des soixante lignes du tableau restent pourtant marquées « Affected », sans correctif publié. Elles concernent des produits de Red Hat qui embarquent Java : sa distribution commerciale de Keycloak, AMQ Broker, streams for Apache Kafka, des images de JBoss EAP, OpenShift Pipelines et les images Jenkins d'OpenShift. **La version communautaire de Keycloak a donc été reconstruite avant la distribution commerciale du même logiciel, chez un éditeur dont le Java était corrigé depuis trois jours.**

Le temps de correction se décompose ainsi en quatre étapes : OpenJDK et Oracle corrigent le composant, chaque distributeur reconstruit son Java, chaque éditeur reconstruit son produit, chaque organisation déploie la nouvelle version. Le bulletin RHSB-2026-018 de Red Hat en tire la conséquence pour les produits qui embarquent leur propre machine virtuelle : passer « via the product vendor's supported path rather than patching the JVM independently » (par le circuit pris en charge par l'éditeur du produit plutôt qu'en corrigeant la JVM de son côté). **Sur ces quatre étapes, la direction n'a la main que sur la dernière, et seulement si elle sait que le produit contient Java.** Entre-temps, il ne reste que la mesure d'attente (une option de démarrage qui désactive le décodeur accéléré), qui suppose elle aussi de savoir où l'appliquer.

L'infrastructure ne se voit qu'au moment où elle casse

> **L'infrastructure selon Star et Ruhleder (1996)**
>
> En étudiant un système collaboratif conçu pour une communauté de généticiens, la sociologue Susan Leigh Star et la chercheuse en systèmes d'information Karen Ruhleder proposent de demander non pas ce qu'est une infrastructure, mais quand quelque chose en devient une (« Steps Toward an Ecology of Infrastructure », *Information Systems Research*, 1996). L'infrastructure est pour elles « a fundamentally relational concept » (un concept fondamentalement relationnel) : un même objet est infrastructure pour l'un et objet de travail pour l'autre. Elles lui reconnaissent plusieurs propriétés. Elle est enfouie dans d'autres structures, transparente à l'usage, construite sur une base installée dont elle hérite les forces et les limites. Elle devient enfin visible lors de la panne : « The normally invisible quality of working infrastructure becomes visible when it breaks » (le caractère normalement invisible d'une infrastructure qui fonctionne devient visible quand elle casse). Ce concept se distingue du *travail invisible* que Star a analysé avec Anselm Strauss : il porte sur les objets techniques, non sur le travail humain.

Le cadre éclaire Java4Shell sur trois points. Le premier est l'enfouissement. Pour l'utilisateur, le service achats et l'exploitant, il n'existe que Jira. Le composant vulnérable se trouve sous chacune de ces couches, et aucune ne le nomme. L'alerte de la CISA le dit en une phrase : les environnements Java « may not appear as separate entries in software inventories » (peuvent ne pas apparaître comme des lignes distinctes dans les inventaires logiciels).

Le deuxième point est le caractère relationnel. Pour l'équipe d'OpenJDK, le décodeur est un objet de travail, corrigé en quelques heures. Pour l'éditeur de Keycloak, c'est une dépendance à reconstruire. Pour une organisation cliente, c'est une infrastructure au sens strict, dont elle dépend sans la voir et qu'elle découvre par l'alerte. Le délai de correction change avec la position de celui qui regarde.

Le troisième point tient à l'absence de responsable. Star le formule dans un texte de 1999 : « Nobody is really in charge of infrastructure » (personne n'a vraiment la charge de l'infrastructure). Dans la chaîne décrite plus haut, chaque acteur répond de son maillon, et aucun n'a la charge de savoir que telle organisation fait tourner tel produit sur tel Java, sur tel processeur. Cette connaissance n'existe que si l'organisation l'a constituée elle-même.

Star reprend à l'historien Geoffrey Bowker l'*inversion infrastructurelle*, qui ramène au premier plan ce qui travaille en arrière-plan. L'alerte du 18 novembre force cette inversion : pendant quelques jours, une ligne de configuration (la version exacte d'un environnement Java) devient une question de direction.

Equifax, 2017 : le correctif existait, l'inventaire non

Le précédent de Log4Shell, en décembre 2021, a été développé dans la chronique consacrée à la faille PHP d'octobre 2026. Le cas d'Equifax isole mieux la question de l'inventaire, parce que le correctif était disponible et que l'organisation n'a pas su où l'appliquer.

En mars 2017, une vulnérabilité critique d'Apache Struts (une bibliothèque Java d'applications web) est rendue publique. Selon le rapport du Government Accountability Office (GAO, l'organe d'audit du Congrès américain) publié le 30 août 2018, des individus non identifiés repèrent dès le 10 mars un serveur d'Equifax qui exécute une version vulnérable, celui de son portail de contestation en ligne. L'intrusion commence le 13 mai et dure environ 76 jours. Elle touche au moins 145,5 millions de consommateurs américains.

Equifax a reconnu devant le GAO que la vulnérabilité « was not properly identified as being present on the online dispute portal » (n'avait pas été correctement identifiée comme présente sur le portail de contestation). Le rapport du sous-comité permanent d'enquête du Sénat américain, rendu public le 6 mars 2019, en établit la raison. Une politique interne imposait de corriger les vulnérabilités critiques en 48 heures. Faute d'inventaire complet des actifs informatiques, les analyses répétées du réseau n'ont jamais trouvé la version vulnérable, et le rapport conclut que cette lacune rendait la politique des 48 heures « useless » (inutile).

Dans les deux cas, le correctif est publié avant que l'organisation sache où le composant se trouve chez elle. Java4Shell y ajoute un étage. Chez Equifax, Struts faisait partie d'une application maison, dont la correction dépendait de l'entreprise seule. Avec un Java embarqué dans un produit tiers, même une organisation dotée d'un inventaire exact attend la livraison de l'éditeur. **L'inventaire reste la condition pour agir, mais il ne fixe plus le délai.**

Le mécanisme, au-delà de Java

Le mécanisme se formule sans nom d'organisation : **lorsqu'un composant partagé est embarqué dans des produits tiers, le délai de correction d'une faille appartient à l'éditeur de chaque produit, et la capacité d'agir de l'organisation cliente dépend d'un inventaire qu'aucun fournisseur ne tient à sa place.**

Il se reproduit à plusieurs conditions. Le composant est assez répandu pour que sa présence aille de soi. Les produits qui l'intègrent le livrent sous leur propre forme (installateur, image de conteneur, appliance), si bien que les mises à jour du système d'exploitation ne l'atteignent pas. Les contrats portent sur le produit et non sur ses composants, sans engagement de délai pour la correction d'une dépendance. L'inventaire, enfin, recense des applications et des licences plutôt que des composants et leurs versions.

Plusieurs signes le rendent observable dans une organisation quelconque. La liste des systèmes concernés par une alerte se construit à la main, en interrogeant les équipes. La version de Java d'un produit n'est connue que d'une personne. Les mesures d'attente sont

inscrites dans les fichiers de configuration sans vérification sur le processus en cours d'exécution, point sur lequel l'avis de Keycloak insiste. Les avis des éditeurs arrivent à des dates différentes, et l'organisation découvre, au fil de ces dates, quels produits elle exploite.

Le cadre de Star et Ruhleder dit aussi pourquoi le mécanisme résiste : une infrastructure ne naît pas de rien, elle hérite de la base installée. Un inventaire des composants ne se décrète pas au lendemain d'une alerte. Il se construit produit par produit, contrat par contrat, et redevient invisible dès que l'alerte est close, comme le montrait déjà la chronique consacrée à l'inventaire cryptographique demandé par l'ANSSI.

> **Grille de lecture**
>
> 1. Pour chaque application exposée sur Internet, un document indique-t-il l'environnement Java qu'elle utilise, son fournisseur et sa version exacte ?
> 2. Parmi ces environnements, combien sont livrés et mis à jour par l'éditeur du produit plutôt que par les équipes internes ?
> 3. Les contrats de maintenance de ces produits prévoient-ils un délai de livraison lorsqu'un composant tiers est vulnérable ?
> 4. Combien de temps a-t-il fallu, après l'alerte du 18 novembre 2026, pour établir la liste des systèmes concernés, et qui l'a établie ?
> 5. Les mesures d'attente ont-elles été vérifiées sur les processus en cours d'exécution, ou seulement dans la configuration ?

Une alerte qui se refermera avant l'inventaire

Java4Shell sera sans doute raconté comme une faille corrigée par Oracle le jour de sa divulgation. Le récit est exact pour le composant, moins pour les organisations qui attendent, depuis le 17 novembre, un correctif publié mais pas encore reçu, parfois de l'éditeur même qui distribue le Java corrigé. L'*infrastructure* au sens de Star et Ruhleder est devenue visible pendant quelques jours parce qu'elle a cassé. Reste à savoir si elle le restera une fois réparée.

Trois indicateurs permettront d'en juger de l'extérieur. Le premier est le tableau de Red Hat pour CVE-2026-114938 : d'ici au 21 mai 2027, le nombre de lignes encore marquées « Affected » dira combien de temps la correction a mis à atteindre les produits d'un seul éditeur. Le deuxième est l'éventuelle inscription de la faille, d'ici au 17 novembre 2027, au catalogue des vulnérabilités exploitées (*Known Exploited Vulnerabilities*) de la CISA, qui dirait si ces délais ont été mis à profit. Le troisième sera la prochaine faille d'un composant embarqué : si les autorités y répètent la consigne de recenser d'abord les environnements livrés avec les produits tiers, l'hypothèse de cette chronique sera confirmée : le correctif circule plus vite que la connaissance de l'endroit où il doit s'appliquer.

Note méthodologique

Les faits reposent sur les documents des autorités et des organisations (CERT-FR, CISA, Red Hat, projets Keycloak et Jenkins) et sur deux titres de référence (Next, The Register). Les affirmations de Calif sur l'exploitation de produits tiers et sur l'usage de l'intelligence artificielle sont attribuées, ses chaînes d'exploitation restant privées. Les scores de gravité divergent selon les émetteurs et sont rapportés sans arbitrage. Le décompte des lignes « Affected » se limite au tableau public de Red Hat consulté le 21 novembre 2026. Pour Equifax, le lien entre l'absence d'inventaire et l'absence de correction est assumé, car il est établi par le rapport du Sénat et rejoint ce qu'Equifax a reconnu devant le GAO. Le rapport de la Chambre des représentants de décembre 2018, dont Equifax a contesté l'exactitude, n'a pas été retenu. L'auteur a préféré le parallèle d'Equifax à celui de Log4Shell, déjà développé dans une chronique précédente. Les faits sont arrêtés au 21 novembre 2026. Recherche de sources, croisement des données et structuration du raisonnement s'appuient sur une assistance par intelligence artificielle. Le choix de l'angle et du cadre théorique, la vérification des sources, les arbitrages et l'établissement du texte final relèvent de l'auteur.

Sources

Documents des autorités et des organisations

CERT-FR (ANSSI), « Vulnérabilité dans Oracle Java SE », alerte CERTFR-2026-ALE-016, 18 novembre 2026, https://cert.cyber.gouv.fr/alerte/CERTFR-2026-ALE-016/

CISA, « Critical Java Vulnerability Requires Immediate Patching », 17 novembre 2026, https://www.cisa.gov/news-events/alerts/2026/11/17/critical-java-vulnerability-requires-immediate-patching

Red Hat, « CVE-2026-114938 », page de vulnérabilité et tableau des produits concernés, consultée le 21 novembre 2026, https://access.redhat.com/security/cve/CVE-2026-114938

Red Hat, « RHSB-2026-018 Remote Code Execution via memory corruption on x86-64 systems - OpenJDK (CVE-2026-114938) - "java4shell" », créé le 17 novembre 2026, mis à jour le 20 novembre 2026, https://access.redhat.com/security/vulnerabilities/RHSB-2026-018

The Keycloak team, « Java4Shell: What Keycloak admins need to do (CVE-2026-114938) », 18 novembre 2026, mis à jour le 19 novembre 2026, https://www.keycloak.org/2026/11/java4shell-vulnerability-cve-2026-114938

Mark Waite, projet Jenkins, « Jenkins and Java4Shell », 18 novembre 2026, https://www.jenkins.io/blog/2026/11/18/mitigating-java4shell/

Presse de référence

Vincent Hermann, Next, « La faille Java4Shell permet d'exécuter du code à distance, notamment dans Jira et Jenkins », 18 novembre 2026, https://next.ink/268262/la-faille-java4shell-permet-dexecuter-du-code-a-distance-notamment-dans-jira-et-jenkins/

Simon Sharwood, The Register, « Oracle reveals fix-this-ASAP-grade Java4Shell bug that allows attacks on many third-party apps », 18 novembre 2026, https://www.theregister.com/security/2026/11/18/oracle-reveals-fix-this-asap-grade-java4shell-bug-that-allows-attacks-on-many-third-party-apps/5310661

Cadre théorique

Susan Leigh Star et Karen Ruhleder, « Steps Toward an Ecology of Infrastructure: Design and Access for Large Information Spaces », *Information Systems Research*, vol. 7, n° 1, mars 1996, p. 111-134, https://doi.org/10.1287/isre.7.1.111

Susan Leigh Star, « The Ethnography of Infrastructure », *American Behavioral Scientist*, vol. 43, n° 3, novembre-décembre 1999, p. 377-391, version en accès libre consultée le 21 novembre 2026, https://ics.uci.edu/~wscacchi/GameLab/Recommended%20Readings/ethnography-infrastructure-Star-1999.pdf

Parallèle historique

U.S. Government Accountability Office, « Data Protection: Actions Taken by Equifax and Federal Agencies in Response to the 2017 Breach », GAO-18-559, 30 août 2018, https://www.gao.gov/assets/700/694347.pdf

United States Senate, Permanent Subcommittee on Investigations, « How Equifax Neglected Cybersecurity and Suffered a Devastating Data Breach », rapport d'enquête, 6 mars 2019, https://www.hsgac.senate.gov/imo/media/doc/FINAL%20Equifax%20Report.pdf