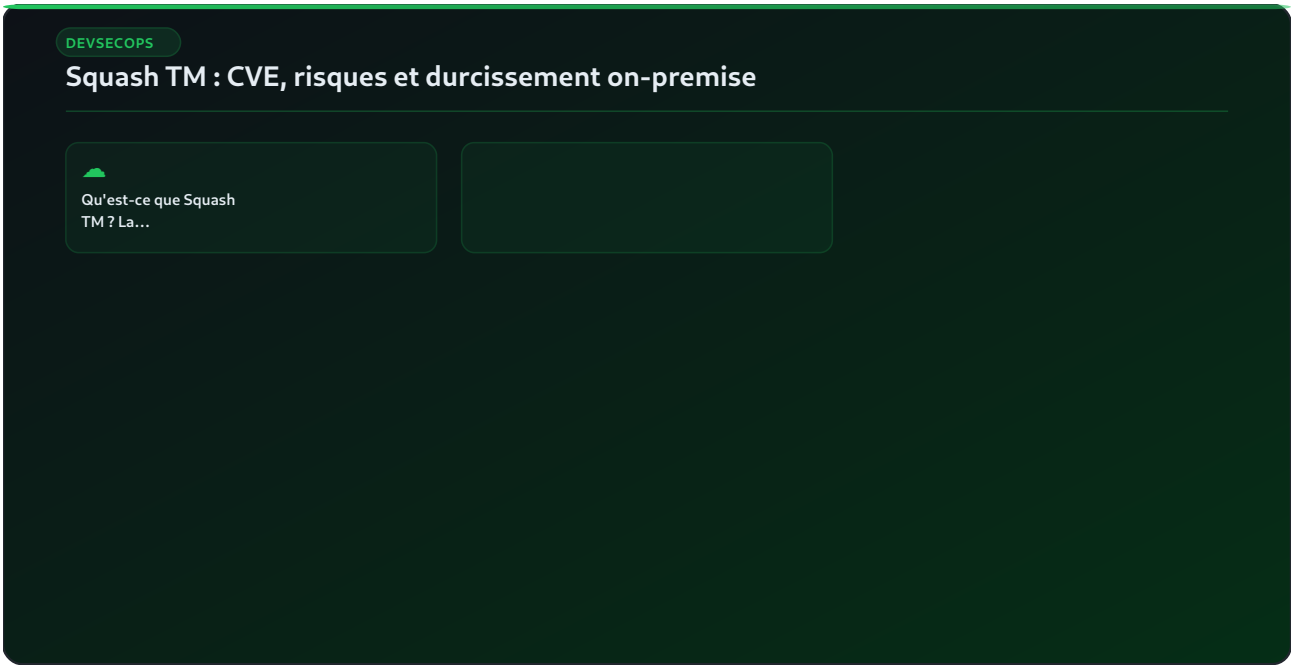
Squash TM : CVE, risques de sécurité et durcissement

> Guide de sécurisation Squash TM on-premise : CVE Log4Shell, CVE-2022-34213, durcissement HTTPS, LDAP/SSO, scan SCA avec Trivy et conformité NIS 2.

Squash TM est la principale suite open-source européenne de gestion des tests logiciels. Éditée par Henix, elle constitue une alternative sérieuse aux outils propriétaires comme HP ALM/ Quality Center, TestRail, Xray ou Zephyr Scale. Déployée on-premise dans des contextes aussi variés que la banque, la santé, la défense ou l'administration, Squash TM gère les exigences, les cas de test, les campagnes d'exécution et les anomalies, en cycle en V comme en contexte agile. Si ses fonctionnalités de test management sont largement documentées, sa posture de sécurité, ses vulnérabilités connues et les bonnes pratiques de durcissement post-installation le sont beaucoup moins. Cet article présente Squash TM dans sa dimension technique complète, passe en revue les CVE publiées (Log4Shell, CVE-2022-34213, CVE-2018-16987), détaille les risques inhérents à un déploiement non durci, et fournit un guide actionnable pour les équipes DevSecOps, les administrateurs système et les RSSI qui découvrent ou héritent de cet outil dans leur parc applicatif.



Qu'est-ce que Squash TM ? La suite open-source de test management

Squash TM est une application web open-source de gestion du référentiel de tests, publiée sous licence LGPL v3 par la société française Henix. Le projet est disponible en téléchargement libre sur Docker Hub et sur le site officiel, ce qui en fait un outil accessible à toute organisation souhaitant maîtriser ses tests logiciels sans dépendance à un éditeur propriétaire. Henix propose également une offre commerciale (Squash Premium / Ultimate) donnant accès au support, aux correctifs prioritaires et à des fonctionnalités avancées.

La suite Squash se compose de plusieurs modules complémentaires :

Squash TM : le cœur — application web de gestion des référentiels (exigences, cas de test, campagnes d'exécution, rapports, anomalies).

Squash AUTOM / DEVOPS : plugins d'automatisation des tests et d'intégration CI/CD (Jenkins, GitLab CI, GitHub Actions).

Squash Orchestrator : moteur d'orchestration des tests automatisés, pilotant l'exécution distribuée sur les agents de test.

Xsquash : plugins d'intégration bidirectionnelle avec Jira Server, Jira Data Center et Jira Cloud.

Les fonctionnalités principales couvrent la création et la structuration des exigences, la rédaction des cas de test avec pas à pas détaillés, la traçabilité exigences/tests, la planification et l'exécution des campagnes, la gestion des anomalies avec connexion aux bug trackers (Jira, GitLab, Mantis, Bugzilla, Azure DevOps), des rapports personnalisés, des champs personnalisés configurables, des tableaux de bord dynamiques et la gestion des tests automatisés.

Squash TM est écrit en Java (Spring Boot / Spring MVC), avec un frontend basé sur des technologies web standard. L'application embarque un serveur Tomcat et peut fonctionner de manière autonome ou être exposée derrière un reverse proxy. Cette stack Java/Spring l'expose mécaniquement aux CVE des dépendances tierces — un point central de la stratégie de sécurisation.

Modes de déploiement : tarball, Docker et SaaS

Squash TM a été conçu dès l'origine pour le déploiement on-premise, ce qui en fait un choix naturel pour les organisations soumises à des contraintes réglementaires ou de souveraineté des données. Trois modes coexistent.

> ⚡ **Retour terrain**
>
> La dette de sécurité dans les dépendances npm/pip/Maven est le vecteur sous-estimé numéro un dans les projets DevSecOps que j'accompagne. Pour un projet Node.js moyen, `npm audit` remonte systématiquement entre 50 et 300 vulnérabilités — dont la grande majorité sont dans des dépendances transitives, pas dans les dépendances directes. La règle que je recommande : 0 CVE critique dans les dépendances directes, et un processus de revue mensuelle des dépendances transitives critiques.

Installation tarball Linux et Windows

Le mode historique repose sur une archive .zip ou .tar.gz à décompresser sur le serveur cible. La configuration s'effectue dans `startup.sh` (Linux) ou `startup.bat` (Windows), en renseignant les paramètres JDBC, le port d'écoute et les options JVM. Le démarrage se fait via `nohup ./startup.sh &`. L'accès par défaut se fait sur `http://localhost:8080/squash` avec les identifiants `admin / admin`.

Sur Windows, Squash TM peut être installé en service via `squash-tm.exe install`. L'installateur .jar Windows est réservé aux environnements de dev/test — jamais en production. Les anciens packages Red Hat (.rpm) et Debian (.deb) ne sont plus supportés depuis Squash TM 2.0 ; les organisations les utilisant encore doivent migrer vers le mode tarball ou Docker.

Déploiement Docker (mode recommandé)

Henix maintient une image officielle et signée sur Docker Hub (`squashtest/squash` , ~476 Mo) avec des mises à jour très fréquentes. Le déploiement Docker est le mode recommandé pour les nouvelles installations pour trois raisons : isolation de l'application vis-à-vis du système hôte, mises à jour simplifiées par remplacement de l'image, et — fait notable — les installations Docker n'étaient pas impactées par Log4Shell lors de la crise de décembre 2021. En pratique, le déploiement s'effectue via Docker Compose avec un service Squash TM, un service base de données et les volumes de persistance. Les credentials s'injectent via des variables d'environnement ou Docker Secrets.

Base de données : MySQL/MariaDB ou PostgreSQL

Deux moteurs sont supportés. Pour MySQL/MariaDB, le moteur InnoDB est obligatoire (MyISAM est explicitement déconseillé — risque de corruption de données). Pour PostgreSQL, l'extension `uuid-ossp` est requise. Dans les deux cas, l'encodage UTF-8 / `utf8mb4` est obligatoire et un compte applicatif dédié `squash-tm` avec droits restreints à la base `squashtm` doit être utilisé.

Offre SaaS Henix

Henix propose une option SaaS hébergée en France, avec données et sauvegardes physiquement localisées en France, hébergeurs sous droit français et conformité RGPD documentée. Cette option élimine la charge opérationnelle des mises à jour, mais introduit une dépendance à un tiers — un arbitrage à évaluer selon le niveau de classification des projets hébergés.

Architecture technique, prérequis JVM et surface d'attaque

La version minimale de JVM requise pour Squash TM 5.x et suivantes est Java 11 LTS. Les versions anciennes (branches 1.x et 2.x) fonctionnaient avec Java 8. Le passage à Java 11 LTS

réduit la surface liée aux vulnérabilités Java 8 sur la cryptographie et la désérialisation. Spring Boot embarque un serveur Tomcat, ce qui signifie que les CVE Tomcat et Spring Framework sont directement concernées si les versions embarquées ne sont pas à jour.

Le frontend utilise historiquement jQuery, ce qui expose l'application aux CVE jQuery (notamment les failles XSS dans les versions antérieures à 3.5.0). Les versions récentes ont progressivement modernisé leur frontend, mais les installations anciennes peuvent encore utiliser des versions jQuery vulnérables.

L'API REST de Squash TM expose les fonctionnalités de l'application via des endpoints HTTP. L'authentification s'effectue par Basic Auth ou par token. Sans contrôle d'accès réseau, cette API permet à un attaquant ayant obtenu des identifiants de manipuler l'ensemble des données de test à distance, d'exporter les cas de test, et d'interagir avec les bug trackers connectés.

CVE publiées : bilan complet des vulnérabilités Squash TM

Le périmètre CVE publié pour Squash TM est relativement limité sur les bases publiques (NVD, CVE Details). Henix publie ses correctifs principalement via les release notes et un fil d'annonces sur son site officiel plutôt qu'en assignant systématiquement un identifiant CVE. Les bases NVD sous-estiment donc probablement le nombre réel de correctifs de sécurité appliqués. Voici les vulnérabilités référencées :

CVE-2018-16987 — Exposition de mots de passe en clair

Cette vulnérabilité découverte en 2018 expose les mots de passe en clair de services externes configurés dans Squash TM (par exemple le champ `ta-server-password` pour les serveurs de tests automatisés) dans le panneau d'administration, via le code source HTML de la page. Un utilisateur disposant d'un accès au panneau d'administration et capable de visualiser le code source peut récupérer les credentials de connexion aux systèmes tiers intégrés.

Cet anti-pattern classique (retour des secrets dans le DOM plutôt que masquage systématique) illustre pourquoi l'accès au panneau d'administration de Squash TM doit être strictement

contrôlé. La mitigation passe par la mise à jour vers une version corrigée et l'audit des comptes de service configurés dans Squash TM pour vérifier qu'ils n'ont pas été exfiltrés.

CVE-2021-44228, CVE-2021-45046, CVE-2021-45105 — Log4Shell

La crise Log4Shell de décembre 2021 a constitué l'épisode de sécurité le plus grave ayant affecté l'écosystème Java depuis des années. Squash TM utilisait Apache Log4j dans certaines de ses versions et configurations, le rendant vulnérable à l'exécution de code à distance non authentifiée (RCE) via le mécanisme de lookup JNDI.

Henix a publié des correctifs couvrant Log4Shell et ses variantes, incluant la mise à niveau de Log4j vers la version 2.17. Ces correctifs sont iso-fonctionnels et couvrent également les plugins Squash AUTOM et DEVOPS. Pour les versions entre 1.22.0 et 1.22.5, une mise à jour de la base de données est également nécessaire. Point notable : les plugins Xsquash pour Jira Server et Jira Data Center n'étaient pas impactés, et les installations Docker de l'époque n'étaient pas affectées selon l'annonce officielle Henix.

Toute installation Squash TM en version antérieure aux correctifs Log4Shell de fin 2021 doit être considérée comme **compromise potentielle** et mise à jour immédiatement. Dans les procédures de réponse à incident, un serveur Squash TM non patché Log4Shell doit être traité comme un vecteur d'intrusion potentiel.

CVE-2022-34213 — Plugin Jenkins Squash TM Publisher

Le plugin Jenkins *Squash TM Publisher* (Squash4Jenkins) stocke le mot de passe de connexion à Squash TM en clair dans le fichier de configuration globale du contrôleur Jenkins (`config.xml`), accessible à tout utilisateur disposant d'un accès au système de fichiers du contrôleur. En pratique, tout administrateur Jenkins, tout script ou agent compromis peut récupérer ce mot de passe en clair.

La mitigation passe par la mise à jour du plugin, l'utilisation d'un compte de service Squash TM dédié aux intégrations Jenkins avec des droits minimaux, et la rotation du password après mise à jour.

Avis Jenkins 2021-11-12 — Squash4Jenkins sans CVE

Antérieure à la CVE-2022-34213, cette vulnérabilité (publiée le 12 novembre 2021) affecte le plugin Squash TM Publisher ≤ 1.0.0. Le plugin implémente un message agent-vers-contrôleur sans validation des entrées. Un attaquant contrôlant un processus agent Jenkins peut remplacer des fichiers arbitraires sur le système de fichiers du contrôleur par une chaîne JSON contrôlée. Aucun correctif n'était disponible au moment de la publication — critique dans les environnements où des agents sont partagés entre plusieurs équipes.

Tableau de synthèse — Posture de sécurité Squash TM

Risque	Sévérité	Mesure de mitigation	Effort
Identifiants admin/admin par défaut	Critique	Changer immédiatement après installation	Faible
HTTP non chiffré (port 8080)	Critique	Reverse proxy Nginx + TLS	Faible
Log4Shell (CVE-2021-44228)	Critique	Mise à jour vers version corrigée (post-déc. 2021)	Moyen
CVE-2022-34213 (Jenkins plugin)	Haute	Mise à jour Squash4Jenkins + rotation credentials	Faible
CVE-2018-16987 (passwords HTML)	Haute	Mise à jour + audit credentials stockés	Faible
Dépendances Java/Spring non patchées	Variable	Scan SCA (Trivy/Grype) + mise à jour mensuelle	Moyen
API REST exposée sans contrôle réseau	Haute	Restriction IP, WAF, tokens à durée limitée	Faible
Credentials plugins en clair/bas droits	Haute	Comptes de service minimaux, tokens OAuth	Moyen
Base de données avec compte root	Haute	Compte dédié squash-tm, droits restreints	Faible
Sauvegardes non chiffrées	Moyenne	Chiffrement GPG/AES-256 des dumps	Faible
Gestion locale des comptes utilisateurs	Moyenne	Intégration LDAP/SAML/OAuth2	Moyen

Risques de sécurité en déploiement on-premise non durci

Au-delà des CVE répertoriées, les déploiements Squash TM on-premise exposent à un ensemble de risques opérationnels récurrents, souvent liés à une configuration par défaut maintenue telle quelle faute de procédure de durcissement formalisée.

Identifiants par défaut non changés. Le risque numéro un des déploiements internes est la non-modification du couple `admin / admin`. Un attaquant ayant accès au réseau interne et connaissant l'URL de l'instance prend le contrôle de l'application en quelques secondes. L'outil de gestion des tests contient typiquement des informations sur les architectures système, les anomalies connues des applications (une mine pour un attaquant) et les credentials des connecteurs vers les systèmes de développement.

Absence de HTTPS. Tout le trafic entre les navigateurs et le serveur transit en clair sur le réseau, exposant les sessions utilisateurs à une interception. Dans un environnement interne segmenté le risque semble limité, mais une MITM sur le LAN, un agent malveillant ou une mauvaise configuration de VLAN peut compromettre les authentifications.

Surface des dépendances Java/Spring. Sans processus de mise à jour régulier, les CVE des bibliothèques embarquées s'accumulent. Un scan SCA sur une installation vieillissante révèle souvent des dizaines de CVE de sévérité variée — dont certaines à fort impact comme Log4Shell.

Plugins et connecteurs tiers. Chaque connecteur activé stocke des credentials vers les systèmes externes (Jira, GitLab, Jenkins, Azure DevOps). Ces credentials sont une cible de valeur pour un attaquant ayant compromis l'interface d'administration, permettant un mouvement latéral vers les systèmes de développement et d'intégration continue. Pour aller plus loin sur les risques de mouvement latéral, consultez nos checklist de sécurité et les techniques documentées dans la catégorie actualités CVE.

Guide de durcissement : les 12 mesures prioritaires

Les mesures suivantes constituent le socle minimum de sécurisation. Elles doivent être appliquées dès le déploiement initial, avant toute mise en production.

1. Changer immédiatement les identifiants par défaut

La première action post-installation est la modification du mot de passe `admin`. Utiliser un mot de passe fort (16+ caractères). Si possible, renommer ou désactiver le compte `admin` générique et créer un compte nominatif pour l'administration.

2. Activer HTTPS via reverse proxy Nginx

Placer Squash TM derrière un reverse proxy avec TLS est la méthode recommandée. La configuration Nginx minimale :

```
server {
    listen 443 ssl http2;
    server_name squash.exemple.fr;
    ssl_certificate /etc/letsencrypt/live/squash.exemple.fr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/squash.exemple.fr/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
    add_header X-Frame-Options DENY always;
    add_header X-Content-Type-Options nosniff always;
    location /api/ {
        allow 10.0.0.0/8;
        deny all;
        proxy_pass http://127.0.0.1:8080;
    }
    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Interdire l'accès direct au port 8080 depuis le réseau via `iptables` ou `nftables`.

3. Isoler le réseau

Squash TM ne doit pas être exposé directement sur Internet sauf besoin métier explicite (dans ce cas, intégrer un WAF). En interne, restreindre l'accès au VLAN des équipes de développement et de test. L'API REST doit être accessible uniquement aux serveurs CI/CD qui en ont besoin.

4. Sécuriser la base de données

Créer un compte dédié `squash-tm` avec les seuls droits nécessaires (`SELECT` , `INSERT` , `UPDATE` , `DELETE`) sur la base `squashtm` . Ne jamais utiliser root/postgres pour le runtime applicatif. Restreindre la connexion à `localhost` ou à l'IP du serveur Squash TM :

```
CREATE USER 'squash-tm'@'127.0.0.1' IDENTIFIED BY 'MotDePasseFort!123';
GRANT SELECT, INSERT, UPDATE, DELETE ON squashtm.* TO 'squash-tm'@'127.0.0.1';
```

5. Préférer le déploiement Docker

L'image officielle `squashtest/squash` est maintenue activement, signée, et bénéficie de mises à jour régulières. Docker facilite les mises à jour (remplacement de l'image), améliore l'isolation et permet d'utiliser Docker Secrets pour les credentials. Utiliser un fichier `.env` non versionné pour les secrets, ou un gestionnaire (Vault, Docker Secrets).

6. Mettre en place un processus de mise à jour régulier

Squash TM publie des versions correctives régulières. Définir un cycle mensuel ou trimestriel selon la criticité. S'abonner aux release notes officielles et au fil d'annonces Henix. Les correctifs critiques (comme Log4Shell) exigent une intervention sous 24-72h.

7. Intégrer l'authentification d'entreprise (LDAP / SAML / OAuth2)

Squash TM supporte LDAP/Active Directory, SAML 2.0 et OAuth2. L'intégration avec l'annuaire d'entreprise permet d'appliquer les politiques de mots de passe centrales, d'activer le

MFA via le fournisseur d'identité, et d'automatiser le déprovisionnement lors des départs. La gestion locale des comptes est à réserver aux comptes techniques.

8. Scanner les dépendances Java (SCA)

Mettre en place un scan SCA (Software Composition Analysis) sur l'image Docker ou les binaires :

Trivy : `trivy image squashtest/squash:latest`

Grype : `grype squashtest/squash:latest`

OWASP Dependency-Check : analyse des JARs embarqués

Intégrer ce scan dans la pipeline CI/CD ou planifier une exécution hebdomadaire avec remontée d'alertes.

9. Limiter les plugins aux strictement nécessaires

Désactiver les connecteurs inutilisés. Pour les plugins actifs : comptes de service dédiés avec droits minimaux, tokens OAuth à durée limitée plutôt que mots de passe pérennes, rotation trimestrielle des credentials, audit régulier des secrets stockés dans l'interface d'administration.

10. Chiffrer les sauvegardes

La base contient des anomalies connues des applications testées, des credentials des connecteurs et des informations utilisateurs. Chiffrer les dumps (GPG ou AES-256), tester la restauration trimestriellement, stocker hors site avec rétention 30 jours minimum.

11. Configurer les en-têtes HTTP de sécurité

Via le reverse proxy : `Strict-Transport-Security` (HSTS), `X-Frame-Options: DENY`, `X-Content-Type-Options: nosniff`, `Content-Security-Policy` adapté, `Referrer-Policy: strict-origin-when-cross-origin`.

12. Journalisation et supervision

Centraliser les logs (Wazuh, Graylog, Elastic). Alertes sur : échecs d'authentification répétés, modifications d'administration, accès API depuis sources inhabituelles. Activer les logs d'accès du reverse proxy pour traçabilité complète.

Scan SCA et veille active sur les dépendances

La gestion des vulnérabilités dans les dépendances est un défi permanent pour les applications Java. La pratique recommandée est d'intégrer un scan Trivy dans la pipeline CI/CD :

```
 # Scan de l'image Docker officielle
trivy image --severity HIGH,CRITICAL squashtest/squash:latest

# Scan des JARs extraits localement
trivy fs --security-checks vuln /opt/squash-tm/lib/
```

OWASP Dependency-Check complète Trivy en analysant les fichiers Maven POM et en produisant un rapport HTML détaillé avec les scores CVSS. Ces scans doivent être exécutés avant chaque mise en production et de manière hebdomadaire sur les instances de production.

La veille active sur les release notes Squash TM est indispensable. Pour les environnements critiques, le support commercial Squash Premium / Ultimate donne accès à des alertes de sécurité prioritaires et à un SLA de correction. Les résultats des scans SCA doivent s'intégrer dans votre procédure de gestion des vulnérabilités.

Intégration LDAP, SAML 2.0 et OAuth2 en environnement entreprise

L'intégration à l'authentification d'entreprise est une mesure essentielle en contexte professionnel. Elle évite la gestion locale des mots de passe Squash TM, applique les politiques d'entreprise (complexité, rotation, MFA) et automatise le déprovisionnement.

LDAP / Active Directory : supporté depuis les premières versions. La configuration renseigne l'URL du serveur LDAP, le DN de bind, le filtre de recherche et les groupes LDAP mappés sur les rôles Squash TM. Utiliser un compte LDAP en lecture seule pour les recherches. Les ressources de notre guide Tiering Model Active Directory documentent les bonnes pratiques de structuration des comptes de service dans l'AD.

SAML 2.0 : disponible sur les versions récentes, intégrable avec Keycloak, ADFS, Okta, Azure AD. L'utilisateur est redirigé vers le fournisseur d'identité, permettant l'activation du MFA sans modification de Squash TM.

OAuth2 : adapté aux environnements cloud ou hybrides. Tester l'intégration en staging avant production, car les configurations SSO peuvent présenter des subtilités sur les scopes et les redirections.

Sauvegardes, chiffrement et plan de reprise d'activité

La base Squash TM contient des informations sensibles qui justifient une politique de sauvegarde rigoureuse : cas de test révélateurs d'architectures internes sensibles, anomalies connues des applications testées, credentials des connecteurs vers les systèmes de développement, données utilisateurs. Sans chiffrement des dumps, un vol de sauvegarde expose l'ensemble de ce patrimoine informationnel.

Recommandations opérationnelles :

Fréquence : sauvegarde quotidienne en production, hebdomadaire minimum en préproduction.

Chiffrement : `mysqldump squashtm | gzip | gpg --encrypt --recipient backup@exemple.fr > squashtm_$(date +%Y%m%d).sql.gz.gpg`

Rétention : 30 jours de sauvegardes quotidiennes, 12 mois de sauvegardes mensuelles.

Test de restauration : trimestriel minimum, résultat documenté.

Stockage hors site : sur un système distinct du serveur de production.

Conformité NIS 2, ISO 27001 et secteurs réglementés

Pour les organisations soumises à la directive NIS 2, Squash TM en tant qu'outil du processus de développement logiciel entre dans le périmètre de la sécurité de la chaîne d'approvisionnement numérique (article 21, §2.d). Les mesures décrites contribuent à la conformité sur les volets : gestion des vulnérabilités (§2.e), contrôle d'accès et authentification forte (§2.i), chiffrement (§2.h), sécurité de la chaîne d'approvisionnement logicielle (§2.d).

Pour les organisations certifiées ou en démarche ISO 27001, la sécurisation de Squash TM relève des contrôles A.8 (gestion des actifs), A.9 (contrôle d'accès), A.12 (sécurité de l'exploitation) et A.14 (sécurité du développement). Un inventaire des instances déployées, documentant version, configuration de sécurité et plan de mise à jour, est une preuve attendue lors des audits.

Pour les secteurs réglementés (santé, finance, défense, aérospatiale), des exigences supplémentaires s'ajoutent : hébergement souverain des données de test, traçabilité des accès, séparation des environnements. L'offre SaaS Henix, avec ses données hébergées en France sous droit français, peut répondre à ces exigences. Notre guide de sécurisation Active Directory couvre les environnements réglementés dans leur ensemble.

FAQ — Sécurité Squash TM

Squash TM est-il adapté aux environnements classifiés ?

En configuration on-premise durcie (HTTPS, LDAP/SSO, réseau isolé, sauvegardes chiffrées, scan SCA), Squash TM peut être déployé dans des environnements à fort niveau d'exigence. Pour les environnements DR ou les OIV/OSE, un accompagnement avec Henix (support Premium/Ultimate) et une analyse de risques formelle (EBIOS RM) sont recommandés.

Log4Shell est-il encore un risque sur les versions récentes ?

Non, si l'installation est à jour. Toutes les versions publiées après décembre 2021 intègrent un Log4j ≥ 2.17 non vulnérable. Le risque subsiste uniquement sur des installations datant d'avant fin 2021.

Comment être alerté des nouvelles CVE Squash TM ?

S'abonner aux release notes sur tm-fr.doc.squashtest.com, au fil d'annonces squashtm.com, aux CVE Details pour "Squash TM", et aux avis Jenkins pour le plugin Squash4Jenkins. Un scan SCA hebdomadaire avec Trivy complète cette veille.

Faut-il préférer la version open-source ou commerciale ?

La version open-source (LGPL) inclut toutes les fonctionnalités de base et reçoit les corrections de sécurité. La version commerciale (Premium/Ultimate) ajoute un SLA de support et des correctifs prioritaires sur les vulnérabilités critiques — recommandé pour les environnements de production critiques.

Conclusion : une sécurisation explicite, pas optionnelle

Squash TM est un outil puissant et légitime pour la gestion des tests logiciels, mais son déploiement on-premise requiert une sécurisation explicite que l'installation par défaut ne fournit pas. Les deux risques immédiats — identifiants `admin / admin` et HTTP non chiffré — se corrigent en quelques minutes. Les risques structurels — gestion des dépendances Java, contrôle d'accès, intégration SSO — demandent un investissement opérationnel continu, proportionnel à la criticité des données hébergées dans l'outil.

La mise en œuvre du guide de durcissement présenté dans cet article, combinée à un cycle de mise à jour régulier et à un scan SCA périodique, permet d'atteindre un niveau de sécurité compatible avec les exigences NIS 2 et ISO 27001. Pour aller plus loin sur la sécurisation de vos

outils DevSecOps et de votre chaîne de développement, contactez nos experts via le formulaire de contact ou découvrez nos livres blancs gratuits sur la sécurisation des infrastructures critiques.

À RETENIR

Points clés à retenir

Squash TM est une application Java/Spring Boot : les CVE des dépendances embarquées se surveillent via Trivy ou Grype.

Les identifiants par défaut `admin / admin` et l'absence de HTTPS sont les deux risques immédiats à éliminer avant mise en production.

Log4Shell (CVE-2021-44228) a affecté les versions on-premise ; les installations Docker n'étaient pas impactées.

Le plugin Jenkins Squash4Jenkins a fait l'objet de deux avis de sécurité distincts ; sa version et ses credentials doivent être audités.

L'intégration LDAP/SAML/OAuth2 est indispensable en contexte entreprise pour centraliser le contrôle d'accès et activer le MFA.

Pour NIS 2 et ISO 27001 : documenter la configuration de sécurité de Squash TM et l'inclure dans les audits annuels.

Sources et références

OWASP — DevSecOps Guidelines

NIST SP 800-190 — Application Container Security Guide

ANSSI — Recommandations de sécurité pour le développement