Analyse de faille (CVE-2023-30625)

Benjelloun El Kbibi Youssef
El-Salihi Nabil
Prada Mejia Juan Pablo

19 janvier 2024

Table des matières

1 Description de la faille

1.1 Contexte

RudderStack [2] est une plateforme open source de gestion des données clients conçue pour aider les entreprises à collecter, unifier et envoyer leurs données client à divers outils et systèmes tiers.

Les fonctionnalités clés de RudderStack comprennent :

Collecte de données RudderStack permet aux entreprises de collecter des données à partir de diverses sources, telles que des sites Web, des applications mobiles et des applications côté serveur. Il prend en charge plusieurs plates-formes et langages de programmation.

Routage des données RudderStack facilite le routage des données vers différents outils tiers et entrepôts de données, la plateforme permet aussi de transformer et de nettoyer les données avant de les envoyer.

Traitement des données en temps réel RudderStack prend en charge le traitement des données en temps réel, permettant aux entreprises de prendre des décisions rapides et éclairées en fonction d'informations à jour.

Confidentialité et conformité La plateforme est conçue en tenant compte de la confidentialité et de la conformité, aidant les entreprises à respecter les réglementations de protection des données et à assurer le traitement sécurisé des informations des clients.

Open Source RudderStack est un projet open source [5], ce qui signifie que son code source est disponible publiquement. Cela permet aux entreprises de personnaliser et d'étendre la plateforme selon leurs besoins spécifiques.

1.2 La faille

La vulnérabilité permet une injection SQL, ce qui pourrait potentiellement conduire à une exécution de code à distance (RCE). Cela est particulièrement préoccupant car le rôle rudder dans PostgreSQL, utilisé par rudder-server, a par défaut des permissions de superutilisateur. De telles permissions pourraient permettre à un attaquant d'exécuter du code arbitraire sur le serveur hébergeant le rudder-server, menant à une compromission de l'intégrité et de la confidentialité du système. Elle a un score CVSS de 8.8 et a été corrigée dans la version 1.3.0-rc.1.[1]

2 Vulnérabilité

La vulnérabilité est une injection SQL qui permet un utilisateur d'éxécuter du code sur le serveur rudder-stack, voler des données... Rudder-stack offre une sdk permettant au développeurs de l'utiliser dans leur application, par exemple une application de e-commerce. Un attaquant peut à travers le frontend de cette application, injecter du code SQL sur le rudder-server en tant que superuser. Cette faille concerne donc principalement des machines serveurs et aussi les données des utilisateurs.

3 Architecture typique

Dans la figure 1, vous trouverez l'architecture typique d'une application web qui utilise RudderStack [3] pour envoyer différentes données. Cette application pourrait être, par exemple, une application web de *e-commerce*. Lorsqu'un achat a lieu, nous souhaitons envoyer les données d'achat à plusieurs équipes, telles que l'équipe marketing, l'équipe d'analyse des ventes, ou simplement les stocker dans un entrepôt tiers.

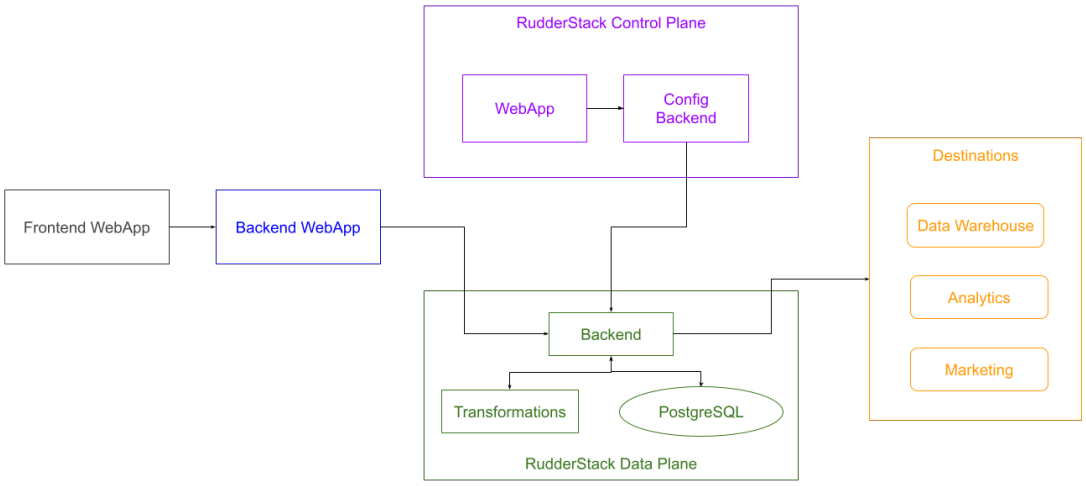

FIGURE 1 – Architecture typique

Quelques explications sur le fonctionnement de l'architecture de RudderStack : nous avons un *frontend* appelé *WebApp* dans la figure 1, capable de modifier les configurations en communiquant avec *ConfigBackend*. Ce *ConfigBackend* communique directement avec le serveur RudderStack, qui utilise une base de données *PostgreSQL* et interagit avec d'autres services pour les transformations de données.

L'application *BackendWebApp* peut communiquer avec le *backend* de RudderStack via un SDK (c'est-à-dire une bibliothèque Node dans notre cas), qui effectuera des appels REST API lorsque des évènements seront déclenchés. Une fois un évènement est déclenché le *backend* reçoit les données et il exécute des transformations sur les données. Les données transformées sont ensuite sauvegardées et envoyées aux destinations désignées.

La base de données *PostgreSQL* est donc principalement utilisée comme base de données en continu pour les données d'évènements. RudderStack stocke temporairement ces données afin de pouvoir les réessayer en cas d'échec de la livraison. Les évènements sont supprimés une fois qu'ils sont livrés avec succès.

4 Limitation de l'impact

4.1 Mettre en place un WAF

Un Web-Application-Firewall [6] pourrait limiter l'impact en filtrant les requêtes malicieuses et bloquer tout ce qui ressemble à une injection SQL. Les logs générés par les WAF peuvent aussi aider à identifier les activités suspicieuse, et par conséquent prendre des précautions.

4.2 Modifications de l'application

Pour limiter l'impact, nous pouvons simplement modifier les droits du rôle `rudder` dans *PostgreSQL* afin de lui retirer les droits de super utilisateur. Aussi RudderStack étant un projet open source [5], nous pouvons facilement modifier l'implémentation de `rudder-server` et le configurer pour effectuer la sanitization des requêtes SQL.

4.3 Observabilité

L'observabilité joue un rôle clé dans l'identification des défaillances d'un système, notamment il faut générer des logs, métriques et traces de tous les services y compris `rudder-server`, et toutes les machines. En ayant un bon système de monitoring, on peut donc identifier l'existence d'un attaquant dans le système, notamment en regardant les logs de `rudder-server` ou du WAF, et remarquer une requête suspicieuse... De cette façon on peut agir plus rapidement afin de stopper l'attaque.

4.4 Isolation de PostgreSQL

Si l'attaquant a accès à Postgres, il ne faut pas qu'il est accès à d'autres parties du système, Postgres doit donc être bien isolé du reste de l'infrastructure.

5 Bonnes pratiques

5.1 Mises à jour régulières

Non seulement pour `rudder-server` mais pour toutes les applications et packages, il faut faire des mises à jour régulière, car ces dernières incluent des correctifs de failles. Dans notre cas, la faille a été résolue dans la version 1.3.0-rc.1.

5.2 Backup de la base de données

Des backup réguliers de la base de données Postgres doivent être faits et stockés dans des machines distantes, ou dans un système de stockage cloud (google cloud storage par exemple). De cette façon si un attaquant exploite la faille en supprimant les données de Postgres, ou en les chiffrants, on aurait un backup déjà prêt.

5.3 Audit de sécurité et sensibilisation

Il faut faire régulièrement des audits de sécurité, soit par l'équipe ou par des intervenants externes, et puis sensibiliser les développeurs et les éduquer sur les meilleures pratiques de sécurité.

6 Cible de sécurité

6.1 Utilisateurs

Les utilisateurs concernés sont principalement les administrateurs de systèmes et les développeurs qui utilisent RudderStack, en particulier ceux qui déploient et gèrent le rudder-server. Les utilisateurs finaux des applications affectées peuvent également être indirectement touchés.

6.2 Biens à protéger

Données sensibles RudderStack informations personnelles, données clients, et autres informations stockées dans les bases de données accessibles par le `rudder-server`.

Intégrité des systèmes RudderStack la sécurité et le bon fonctionnement du `rudder-server` et du reste du système.

Réputation de l'organisation RudderStack ce point est subjectif car celà dépend de l'organisation en question, mais il faut garder la confiance des clients et des utilisateurs dans la sécurité des systèmes.

6.3 Menaces

Les menaces sont principalement des injections SQL, mais aussi en raison des privilèges élevés de l'utilisateur `rudder`, on peut avoir de l'exécution distante de code, et puis selon l'infrastructure et le système d'informations mis en place, on peut potentiellement avoir des attaques secondaires en utilisant le serveur compromis comme tremplin pour d'autres attaques dans le réseau.

7 Démo

7.1 Explication de la démo

Pour reproduire la démo, nous utiliserons `docker-compose` pour exécuter RudderStack, `workspaceConfig.json` qui initialise certains paramètres essentiels au bon fonctionnement du `rudder-server` au démarrage, et un script Python responsable de l'exécution d'une requête HTTP.

L'objectif de cette démo est de mettre en évidence les conséquences de ne pas prendre les mesures nécessaires pour prévenir les injections SQL. Dans la démo, nous faisons une requête `POST` vers le serveur `ruder-server`, et dans le corps de cette requête, nous avons

inséré un *payload* qui videra le contenu de toutes les tables dans la base de données et nous provoquerons également un déni de service par la suite.

7.2 Instructions pour executer la demo

1. `docker-compose -f rudder-stack-exploitable.yaml up -d -build`
   Cela va mettre en place tous les services de RudderStack. Nous allons accorder une attention particulière au serveur Rudder (`rudder-server`) en tant que backend.
2. Révisons les relations de la base des données : Ouvrez un terminal séparé et exécutez, (ne fermez pas ce terminal !) `docker exec -it db psql -U rudder -d jobsdb`
3. Listez tables avec le commande : `\dt`
4. Ouvrez un terminal séparé et exécutez : `python3 sabotage.py`
5. Cela exécute une injection de requête qui aura pour effet :
   — La suppression de toutes les tables ou relations de la base de données.
   — La désactivation de toutes les fonctionnalités fournies par RudderStack.
   — L'arrêt du processus en cours d'exécution dans le serveur backend.
6. Confirmons la bonne exécution de notre attaque, reprenons le terminal de la base des données. Révisez les tables (`\dt`)
7. Révisons les services `docker ps`. Notons que le service backend est arreté.
8. `rudder-server` a été victime d'une injection sql qui a effacé toutes les tables ainsi que d'un déni de service

Références

[1] *CVE details*. URL : https://nvd.nist.gov/vuln/detail/CVE-2023-30625.

[2] *RudderStack*. URL : https://www.rudderstack.com/uxr-20-b/.

[3] *RudderStack Architecture*. URL : https://www.rudderstack.com/docs/resources/rudderstack-architecture/.

[4] *RudderStack Architecture*. URL : https://www.rudderstack.com/docs/get-started/quickstart/server-side-quickstart/.

[5] *ruder-server Github page*. URL : https://github.com/rudderlabs/rudder-server.

[6] *Web-Application-Firewall*. URL : https://en.wikipedia.org/wiki/Web_application_firewall.

Glossaire

RCE Remote Code Execution. 3

SDK Software Development Kit. 4

WAF Web Access Firewall. 5