Programmation sécurisée (Secure coding)
Common Weakness Enumeration (CWE)
Répertoire des Faiblesses Communes

Halim Djerroud

LISV
Laboratoire d'ingénierie
des systèmes de Versailles

UVSQ
université PARIS-SACLAY

révision : 0.1

Plan

MITRE

MITRE (ce n'est pas acronyme) est une organisation à but non lucratif basée aux États-Unis, spécialisée dans des activités visant l'intérêt public (américaines). Ses principaux champs d'expertise :

- L'aviation
- La défense
- La santé
- La sécurité intérieure
- **La cybersécurité**

MITRE : Production pour la cybersécurité

MITRE développe plusieurs cadres de référence essentiels en cybersécurité, notamment :

- **Common Weakness Enumeration (CWE) :** un système de catégorisation des faiblesses et vulnérabilités logicielles, aidant les développeurs et les analystes en sécurité à identifier et à atténuer les vulnérabilités potentielles dans les applications.
- **MITRE ATTCK :** une base de connaissances accessible mondialement qui décrit les tactiques et techniques employées par les cyberadversaires, basée sur des observations réelles. Elle est utilisée pour développer des modèles de menace et des méthodologies pour les professionnels de la cybersécurité et les organisations. MITRE ATTCK
- **Common Vulnerabilities and Exposures (CVE) :** une liste de vulnérabilités et d'expositions liées à la sécurité de l'information, fournissant des identifiants communs pour les vulnérabilités connues afin de faciliter le partage des données entre les outils de sécurité et les services.

CWE

- Common Weakness Enumeration (CWE), (*Répertoire des Faiblesses Communes*), est une classification standardisée des faiblesses de sécurité des logiciels. Il est maintenu par l'organisation MITRE et est largement utilisé dans le domaine de la sécurité informatique pour identifier, documenter et partager des informations sur les vulnérabilités des logiciels et des systèmes

`https://cwe.mitre.org/`

Objectifs principaux du CWE

CWE Traite des catégories de vulnérabilités logicielles et matérielles, elle a pour objectif :

- **Standardisation :** Fournir un langage commun pour décrire les faiblesses de sécurité des logiciels.
- **Éducation :** Aider les développeurs, les architectes et les testeurs à comprendre et à prévenir les vulnérabilités.
- **Évaluation :** Servir de base pour les outils d'analyse de la sécurité et les audits.
- **Priorisation :** Permettre aux organisations de se concentrer sur les faiblesses les plus critiques.

Site web du CWE

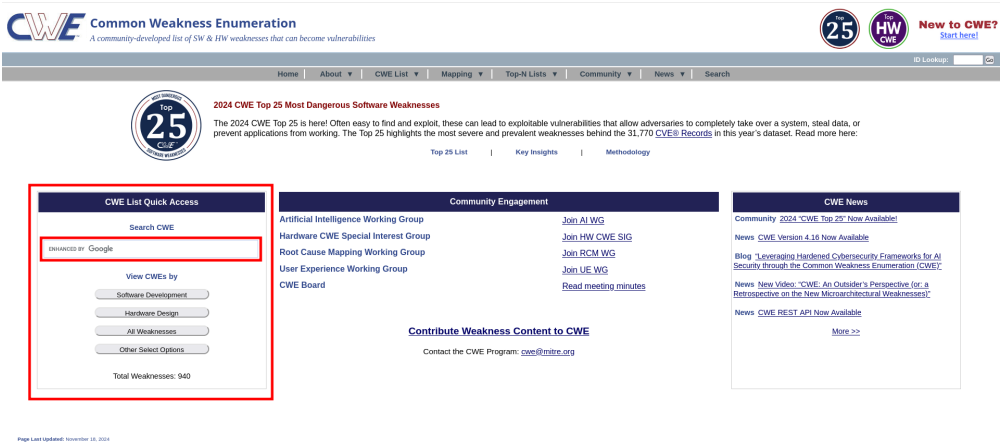


Figure { https://cwe.mitre.org/

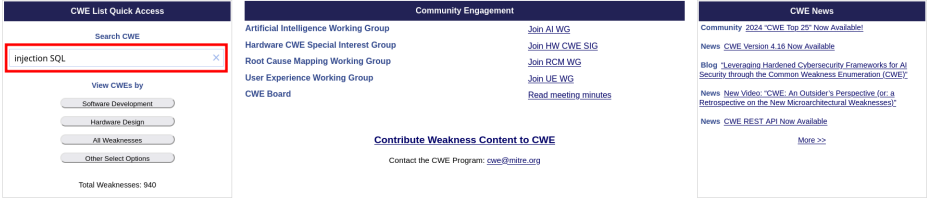
Organisation

- **Piliers :** représentent les grands domaines ou dimensions de classification des faiblesses dans le CWE. Ils servent à regrouper les faiblesses en fonction de leur contexte global.
- **Classes :** des regroupements logiques de faiblesses qui partagent des caractéristiques fondamentales. Elles définissent des types de problèmes récurrents rencontrés dans les logiciels et les systèmes.
- **Catégories :** regroupements spécifiques de faiblesses au sein d'une classe ou d'un pilier. Elles fournissent une granularité plus fine pour classer des types de problèmes courants, souvent en fonction de leur impact ou de leur exploitation.
- **Variantes :** formes spécifiques d'une faiblesse donnée. Elles décrivent comment une faiblesse particulière peut se manifester ou être exploitée dans des contextes précis. Ce niveau détaille les cas particuliers ou les sous-types d'une vulnérabilité.

Exemple : Injection SQL

Exemple : Injection SQL

Exemple : Injection SQL



About 334 results (0.17 seconds)

CWE-89: Improper Neutralization of Special Elements used ... - MITRE
cwe.mitre.org › CWE List
Since **SQL** databases generally hold sensitive data, loss of confidentiality is a frequent problem with **SQL injection** vulnerabilities. Authentication. Technical ...

CWE-564: **SQL Injection**: Hibernate (4.16) - MITRE
cwe.mitre.org › CWE List
CWE-564: **SQL Injection**: Hibernate ... Abstraction: Variant Variant - a weakness that is linked to a certain type of product, typically involving a specific ...

CWE Top 25 Most Dangerous Software Weaknesses - MITRE
cwe.mitre.org › top25
Nov 18, 2024 ... ... **injection**); Cost Savings – Fewer vulnerabilities in product development mean fewer issues to manage post-deployment, ultimately saving money ...

CWE-566: Authorization Bypass Through User-Controlled **SQL** ...
cwe.mitre.org › CWE List
The following code uses a parameterized statement, which escapes metacharacters and prevents **SQL injection** vulnerabilities, to construct and execute a **SQL** query ...

Exemple : Injection SQL

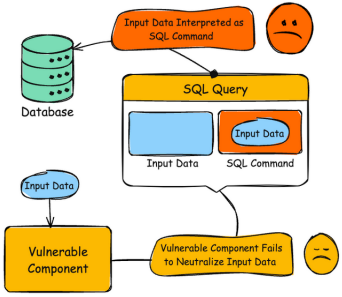
CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Weakness ID: 89
Vulnerability Mapping: ALLOWED
Abstraction: Base

View customized information: Conceptual | Operational | Mapping Friendly | Complete | Custom

Description

The product constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component. Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data.



Introduction à CWE-89

CWE ID : CWE-89
Nom : Injection SQL
Description :
Une injection SQL se produit lorsqu'un attaquant insère des données malveillantes dans une requête SQL. Cela permet d'exécuter des commandes non prévues, affectant potentiellement la *confidentialité*, l'*intégrité* ou la *disponibilité*.

Causes principales

Les injections SQL surviennent en raison de :

- Validation insuffisante des entrées utilisateur.
- Construction de requêtes dynamiques via la concaténation de chaînes.
- Absence de requêtes paramétrées.
- Utilisation directe de données non filtrées.

Conséquences potentielles

Une injection SQL peut entraîner :

- **Confidentialité :** Lecture non autorisée de données sensibles.
- **Intégrité :** Modification ou suppression de données critiques.
- **Disponibilité :** Perturbation ou destruction des bases de données.
- **Privilèges :** Prise de contrôle du serveur de base de données.

Exemple vulnérable (Python)

Code vulnérable utilisant une concaténation dangereuse :

```
user_input = "admin' OR '1'='1"
query = f"SELECT * FROM users WHERE username = '{user_input}'"
cursor.execute(query)
```

Impact : Un attaquant peut contourner l'authentification en utilisant des entrées comme : ' OR '1'='1

Techniques d'atténuation

1. **Utiliser des requêtes paramétrées :**
   - Exemple en Python :

```
query = "SELECT * FROM users WHERE username = ?"
cursor.execute(query, (username,))
```

2. **Sanitisation des entrées utilisateur.**
3. **Adopter un ORM** (e.g., SQLAlchemy).
4. **Limiter les privilèges utilisateur** dans la base de données.
5. **Configurer un pare-feu d'application web (WAF).**

Relations avec d'autres CWE

- **CWE-20 :** Validation insuffisante des entrées utilisateur.
- **CWE-116 :** Encodeur de sortie incorrect ou absent.
- **CWE-98 :** Inclusion dynamique non sécurisée.

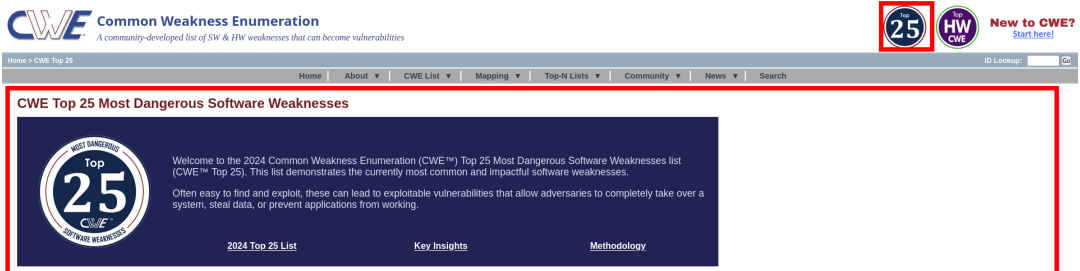
Conclusion sur CWE-89

CWE-89 (Injection SQL) est l'une des failles les plus critiques dans les applications. Elle peut être évitée en :

- Validant systématiquement les entrées utilisateur.
- Utilisant des requêtes paramétrées.
- Adoptant des bonnes pratiques de sécurité.

Protégez vos applications en intégrant la sécurité dès le développement !

Top 25 des Vulnérabilités



Qu'est-ce que le Top 25 des Vulnérabilités ?

- Une liste des failles logicielles les plus courantes et critiques publiée par le CWE (Common Weakness Enumeration).
- Inclut des vulnérabilités comme les dépassements de tampon, les injections SQL, les failles d'authentification, etc.

Les Intérêts Principaux

Amélioration de la Sécurité :

- Réduit les vulnérabilités critiques dans les logiciels.
- Protège contre des failles exploitables courantes.

Réduction des Coûts :

- Correction en phase de développement = moins coûteuse.
- Réduction des coûts liés aux audits, correctifs et incidents.

Conformité et Réputation :

- Respect des normes (ISO 26262, DO-178C, etc.).
- Améliore la fiabilité et la confiance des utilisateurs.

Autres Bénéfices

- **Amélioration des compétences techniques :** Meilleures pratiques de codage.
- **Protection contre les cyberattaques :** Réduction des points d'entrée pour les attaquants.
- **Augmentation de la qualité :** Code plus stable et performant.

Pour quoi maîtriser le Top 25 des vulnérabilités

- Maîtriser le Top 25 des vulnérabilités est essentiel pour développer des logiciels sécurisés et fiables.
- Cela réduit les risques, garantit la conformité et renforce la confiance des utilisateurs.
- Adoptez les meilleures pratiques dès aujourd'hui pour un développement sécurisé !

Top 25 des vulnérabilités (1-12)

Rang	ID CWE	Nom	Score	vs 2023
1	CWE-79	Neutralisation incorrecte des entrées lors de la génération de pages web (‘Cross-site Scripting’)	56,92	+1
2	CWE-787	Écriture hors limites (Out-of-bounds Write)	45,20	-1
3	CWE-89	Neutralisation incorrecte des éléments spéciaux utilisés dans une commande SQL (‘Injection SQL’)	35,88	0
4	CWE-352	Falsification de requête intersite (CSRF)	19,57	+5
5	CWE-22	Limitation incorrecte d’un nom de chemin à un répertoire restreint (‘Traversal de chemin’)	12,74	+3
6	CWE-125	Lecture hors limites	11,42	+1
7	CWE-78	Neutralisation incorrecte des éléments spéciaux utilisés dans une commande OS (‘Injection de commande OS’)	11,30	-2
8	CWE-416	Utilisation après libération	10,19	-4
9	CWE-862	Autorisation manquante	10,11	+2
10	CWE-434	Téléchargement non restreint de fichiers de type dangereux	10,03	0
11	CWE-94	Contrôle incorrect de la génération de code (‘Injection de code’)	7,13	+12
12	CWE-20	Validation incorrecte des entrées	6,78	-6

Top 25 des vulnérabilités (13-15)

Rang	ID CWE	Nom	Score	vs 2023
13	CWE-77	Neutralisation incorrecte des éléments spéciaux utilisés dans une commande (‘Injection de commande’)	6,74	+3
14	CWE-287	Authentification incorrecte	5,94	-1
15	CWE-269	Gestion incorrecte des privilèges	5,22	+7
16	CWE-502	Désérialisation de données non fiables	5,07	-1
17	CWE-200	Exposition d’informations sensibles à un acteur non autorisé	5,07	+13
18	CWE-863	Autorisation incorrecte	4,05	+6
19	CWE-918	Requête côté serveur forgée (SSRF)	4,05	0
20	CWE-119	Restriction incorrecte des opérations dans les limites d’un tampon mémoire	3,69	-3
21	CWE-476	Déréférencement de pointeur NULL	3,58	-9
22	CWE-798	Utilisation de mots de passe codés en dur	3,46	-4
23	CWE-190	Dépassement d’entier ou retour à zéro	3,37	-9
24	CWE-400	Consommation non contrôlée de ressources	3,23	+13
25	CWE-306	Authentification manquante pour une fonction critique	2,73	-5

Open Source Security Foundation (OpenSSF)

Open Source Security Foundation (OpenSSF), une organisation créée pour renforcer la sécurité des logiciels open source. Les travaux et ressources produits par le Working Group (WG) dédié aux meilleures pratiques pour les développeurs de logiciels open source.

Le dépot GitHub

https://github.com/ossf/wg-best-practices-os-developers/tree/main/docs/Secure-Coding-Guide-for-Python

Liste des vulnérabilités (TP)

- CWE-134
- CWE-197
- CWE-754
- CWE-1109
- CWE-1339
- CWE-392
- CWE-230
- CWE-838

CWE-134

CWE-134

Use of Externally-Controlled Format String

Utilisation d'une chaîne de formatage contrôlée par une source externe

`https://cwe.mitre.org/data/definitions/134`

- Cette vulnérabilité survient lorsque des chaînes de formatage influencées par l'utilisateur sont passées à des fonctions de formatage, ce qui peut permettre à un attaquant d'exécuter du code non autorisé ou de provoquer des comportements inattendus.

https ://github.com/ossf/wg-best-practices-os-developers/blob/main/docs/Secure-Coding-Guide-for-Python/CWE-664/CWE-134/README.md

Introduction à CWE-134

- **CWE-134 : Utilisation d'une chaîne de formatage contrôlée par une source externe**.
- Cette vulnérabilité survient lorsque des données externes sont utilisées dans des chaînes de formatage sans validation adéquate.
- Cela peut entraîner :
  - L'exécution de code arbitraire.
  - La divulgation de données sensibles.
  - Des plantages d'application.

Exemple de code non sécurisé

Code problématique avec entrée utilisateur :

Code Python

```
user_input = "{0} + {1}".format(5, 10)
result = eval(user_input)   # Permet une injection de code
```

Problème :

- L'attaquant peut injecter du code malveillant via l'entrée utilisateur.
- Exemple : `5.__class__.__base__`.

Exemple sécurisé

Solution corrigée :

Code Python

```python
user_input = "World"
safe_string = "Hello {name}!".format(name=user_input)
print(safe_string)
```

Améliorations :

- Validation des entrées utilisateur.
- Utilisation de chaînes statiques.

Impact technique de CWE-134

- **Exécution de code arbitraire :** Permet l'injection de code Python malveillant.
- **Plantages :** Les erreurs dans le formatage peuvent provoquer des exceptions.
- **Vol de données sensibles :** Possibilité d'exfiltrer des données système.

Recommandations pour la sécurité

1. Évitez d'utiliser des fonctions comme `eval`.
2. Utilisez des chaînes statiques ou des f-strings pour le formatage.
3. Validez systématiquement les entrées utilisateur.
4. Faites appel à des bibliothèques tierces pour gérer les données sensibles.

Conclusion

- CWE-134 est une vulnérabilité critique à prendre en compte dans le développement Python.
- En suivant les bonnes pratiques de codage sécurisé, il est possible de réduire significativement les risques.
- Pour plus de détails, consultez le guide sur GitHub :

Lien GitHub

https://github.com/ossf/wg-best-practices-os-developers/blob/main/docs/Secure-Coding-Guide-for-Python/CWE-664/CWE-134/README.md