22 Mars 2022

OWASP API Security Top 10: Comprendre les menaces qui ciblent les APIs

Olivier Melis

Directeur Commercial France BeNeLux

Pierrick Prevert

Responsable Avant Vente EMEA

Intervenant

Pierrick Prévert

Responsable Avant Vente EMEA

42Crunch

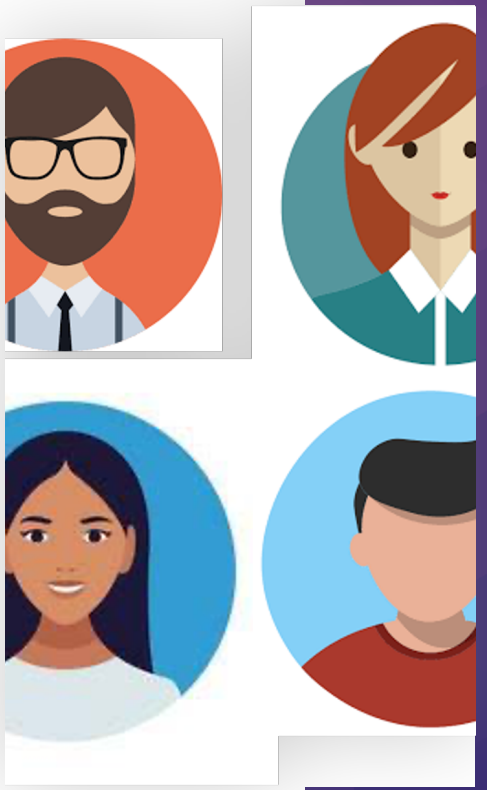
Règles du Webinaire

- Tous les participants sont en sourdine
- Les questions sont posées via le module Q&A
- L'enregistrement du webinaire sera partagé
- Il y aura 2 questions de sondage

Sondage - Question 1:

Quelle est la description qui correspond le mieux à votre métier actuel?

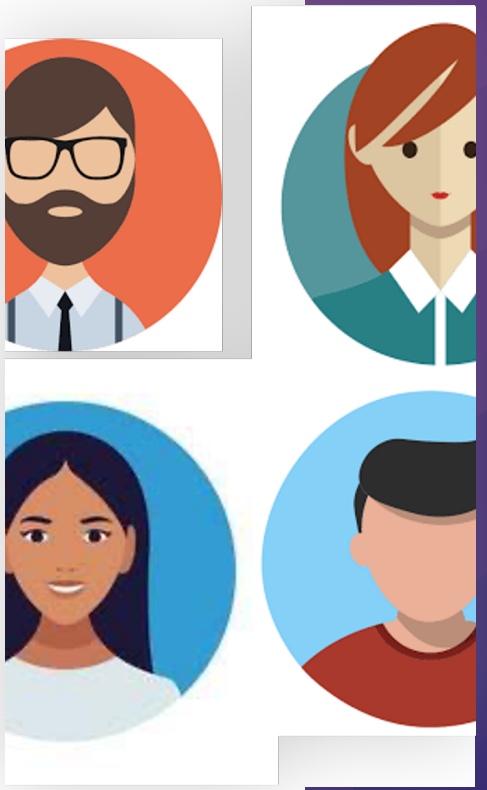
1. Développeur d'API
2. Ingénieur Sécurité
3. Architecte d'Entreprise
4. Responsable de la Sécurité
5. Développeur Front ou Back End
6. Autre



Sondage - Question 1:

Quelle est la description qui correspond le mieux à votre métier actuel?

Développeur d'API	14%
Ingénieur Sécurité	14%
Architecte d'Entreprise	14%
Responsable de la Sécurité	0%
Développeur Front ou Back End	21%
Autre	36%



OWASP API Security Top 10

API1:2019 Broken Object Level Authorization
API2:2019 Broken User Authentication
API5:2019 Broken Function Level Authorization

} Problèmes d'authentication/ autorisation (AuthN/AuthZ)

API3:2019 Excessive Data Exposure
API6:2019 Mass Assignment
API8:2019 Injection

} Problèmes de protection des données

API4:2019 Lack of Resources & Rate Limiting
API7:2019 Security Misconfiguration
API9:2019 Improper Assets Management
API10:2019 Insufficient Logging & Monitoring

} Problèmes de gouvernance/ d'opération

OWASP API SECURITY TOP 10

Problèmes d’authentification et d’autorisation

API1:2019 Broken Object Level Authorization

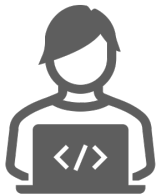
Un attaquant substitue à l'ID (identifiant) d'une de ses ressources l'ID d'une ressource d'un autre utilisateur. L'absence de contrôle de droits effectif permet à l'attaquant d'y accéder.

Deux mécanismes oeuvrent à BOLA:

- *Contrôle d'autorisation défaillant*
- *Prédictibilité des IDs et la possibilité de réutilisation*

API1:2019 Broken Object Level Authorization

Requête



POST /pet/12

Authorization: **af23b**

{species: “Chat”,
name: “Tigrou”}

Backend API

Décision AuthN

Est-ce que le token **af23b** est valide?
Qui utilise le token **af23b**?
=> **user56**

Décision AuthZ

Est-ce que **user56** peut effectuer un **POST** sur les adresses **/pet/{petId}**?

Est-ce que **user56** a le droit de modifier le pet **12**?

API1:2019 Broken Object Level Authorization

Prévenir BOLA :

- Définir une matrice de droits d'accès, lier les ressources à cette matrice
- Utiliser des autorisations fines dans chaque couche de contrôle
- Ne pas utiliser d'IDs devinables (123, 124, 125) mais des UUIDs
- Idéalement, lier cryptographiquement les IDs à la session, afin de les rendre éphémères
- Quand cela est possible, ne pas exposer d'ID (e.g. /info/me plutôt que /info/1234)
- Les scopes OAuth ne permettent *pas* d'adresser BOLA !

BOLA: Uber (2019)

Source

https://www.appsecure.security/blog/how-i-could-have-hacked-your-uber-account

- **Deux problèmes**

- Capacité de récupération de l'UUID privé d'un conducteur via un message d'erreur (**API3:2019**)

Récupération avec cet ID de l'ensemble des informations (dont la session) d'un utilisateur via un endpoint (**API1:2019**)

Adresser BOLA avec 42Crunch

Deux approches possible :

1. Remédiation

- Créer une matrice de droits d'accès
- Utiliser un gestionnaire de policies granulaire (e.g. Open Policy Agent)
- Le Firewall s'intègre avec ce PDP (Policy Decision Point) pour vérifier les droits à chaque appel

2. Atténuation

- Recommandation OWASP: virtualisation d'ID
- Générer ID aléatoires, non devinables, liés à la session
- Le Firewall substitue les IDs faibles d'une API par ces IDs virtualisés

API2:2019 Broken User Authentication

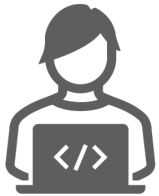
Les mécanismes d'authentification sont souvent mal implémentés et permettent à un attaquant de compromettre des jetons d'authentification ou d'utiliser l'identité d'un autre utilisateur.

Deux mécanismes principaux

- *Abus du processus d'authentification pour obtenir un jeton d'authentification*
- *Modification d'un jeton d'authentification valide*

API2:2019 Broken User Authentication

Requête



POST /pet/12

Authorization: **af23b**

{species: “Chat”,
name: “Tigrou”}

Backend
API

Décision AuthN

Est-ce que le token **af23b** est valide?
Qui utilise le token **af23b**?
=> **user56**

Décision AuthZ

Est-ce que **user56** peut effectuer un **POST** sur les adresses **/pet/{petId}**?

Est-ce que **user56** a le droit de modifier le pet **12**?

API2:2019 Broken User Authentication

Prévenir Broken User Authentication :

- Par défaut : aucun endpoint sans authentification !
- Utiliser des tokens à durée de vie courte
- Bien choisir son grant type OAuth (auth_code + PKCE, bon défaut)
- Valider les tokens (claims, header *et* signature)
  - None algorithm attack
  - JWT RFC8725
  - Vérifier les signature ! Utiliser un stockage sécurisé des secrets !
- Scanner régulièrement ses repositories (e.g. Github secret scanning)

Broken User Authentication: Auth0 (2020)

Source

https://insomniasec.com/blog/auth0-jwt-validation-bypass

Problème

- Attaque « alg: none ». Auth0 empêchait bien l'utilisation de l'algorithme « none », mais ne vérifiait pas la casse. « alg : nonE »

Adresser Broken User Authentication avec 42Crunch

Audit
- Détection des endpoints non sécurisés (intégration IDE et CI/CD)
- Détection des mécanismes faibles d'authenfication

Scan
- Test des endpoints authentifiés avec et sans token
- Injection de tokens malformés/expirés (Q4)

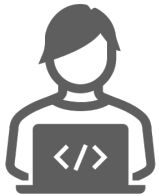
Protection
- Bloque les requêtes non authentifiés sur les endpoints authentifiés
- Validation des tokens JWT (RFC8725)
- Protége les endpoints OAuth (token/authorize)

API5:2019 Broken Function Level Authorization

Le contrôle d'accès est mal effectué au niveau d'une opération. Cela peut être dû à une gestion des droits d'accès confuse, ou à l'absence de restriction d'accès aux fonctions administrateur.

API5:2019 Broken Function Level Authorization

Requête



POST /pet/12

Authorization: **af23b**

{species: “Chat”,
name: “Tigrou”}

Backend API

Décision AuthN

Est-ce que le token **af23b** est valide?
Qui utilise le token **af23b**?
=> **user56**

Décision AuthZ

Est-ce que **user56** peut effectuer un **POST** sur les adresses **/pet/{petId}**?

Est-ce que **user56** a le droit de modifier le pet **12**?

API5:2019 Broken Function Level Authorization

Adresser Broken Function Level Authorization :

- Interdire tout accès par défaut
- Utiliser une authentification fine dans chaque couche de contrôle
- Si utilisation de OAuth : se servir les scopes !
- Ecrire des tests de validation sur chaque opération pour chaque rôle

Adresser Broken Function Level Auth avec 42Crunch

Audit
- Détection des opérations non sécurisées
- Détection des scopes inconnus

Scan
- Appelle l'API avec des verbes inconnus
- Teste les endpoints avec et sans token

Protection
- Bloque les verbes (GET, POST…) non définis
- Bloque les chemins non définis (dont les attaques « path traversal »)
- Bloque les requêtes vers les APIs non définies

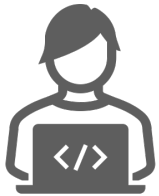
Problèmes de protection des données

API3:2019 Excessive data exposure

Afin de rendre leurs implémentations génériques, les développeurs ont tendance à exposer toutes les propriétés d'un objet sans considérer la sensibilité, laissant le soin aux clients API de filtrer les informations avant de, par exemple, les afficher à un utilisateur.

API3:2019 Excessive data exposure

Requête



GET /user/12

Authorization: **af23b**

Backend API

Décision AuthN

Est-ce que le token **af23b** est valide?
Qui utilise le token **af23b**?
=> **user56**

Décision AuthZ

Est-ce que **user56** peut effectuer un **GET** sur les adresses **/user/{userId}**?

Est-ce que **user56** a le droit de voir les informations de l'utilisateur **12**?

Réponse

{name: "Jean Doh",
session_id:
12341543543543}

API3:2019 Excessive data exposure

Adresser Excessive data exposure

- Par défaut, considérer tout client comme hostile
- Définir précisément chaque réponse et chaque propriété autorisée
- Valider que l'implémentation ne renvoie que les champs autorisés
- Attention à ne jamais retourner d'exceptions !

Excessive data exposure: Chess.com (2020)

Source

https://samcurry.net/hacking-chesscom/

Problème

- Fonction communautaire pour obtenir des informations sur un utilisateur
- L'endpoint renvoyait des PIIs ainsi que la session de l'utilisateur !

Adresser Excessive data exposure avec 42Crunch

Audit
- Valide les schémas de réponse et encourage à leur spécification complète
- Toutes les réponses sont spécifiées et ont un schéma

Scan
- Alerte dès qu'une réponse contient un schéma non connu / non valide

Protection
- Bloque les réponses non prévues (code d'erreur non défini)
- Bloque les réponses invalides d'après le schéma de réponse

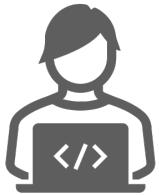
API6:2019 Mass Assignment

Lier des données utilisateur à un modèle de données sans vérifier que l'ensemble des propriétés sont admises amène à un problème de Mass Assignment.

En devinant les propriétés ou en les trouvant dans d'autres endpoints, dans la documentation etc. un attaquant peut modifier des propriétés qu'il ne devrait pas pouvoir modifier.

API6:2019 Mass Assignment

Requête



POST /user/12

Authorization: af23b

{name: “Jean Doh”,
email: “jeandoh@etc.etc”,
admin: true}

Backend API

Décision AuthN

Est-ce que le token **af23b** est valide?
Qui utilise le token **af23b**?
=> **user56**

Décision AuthZ

Est-ce que **user56** peut effectuer un **POST** sur les adresses **/user/{userId}**?

Est-ce que **user56** a le droit de voir les informations de l’utilisateur **12**?

API6:2019 Mass Assignment

Adresser Mass Assignment

- Passer par une représentation intermédiaire restrictive entre les données client et le modèle de données
- Valider que chaque propriété transmise dans un objet/array/dict est autorisée
- Par défaut, utiliser dans ses contrats d'API additionalProperties=false !

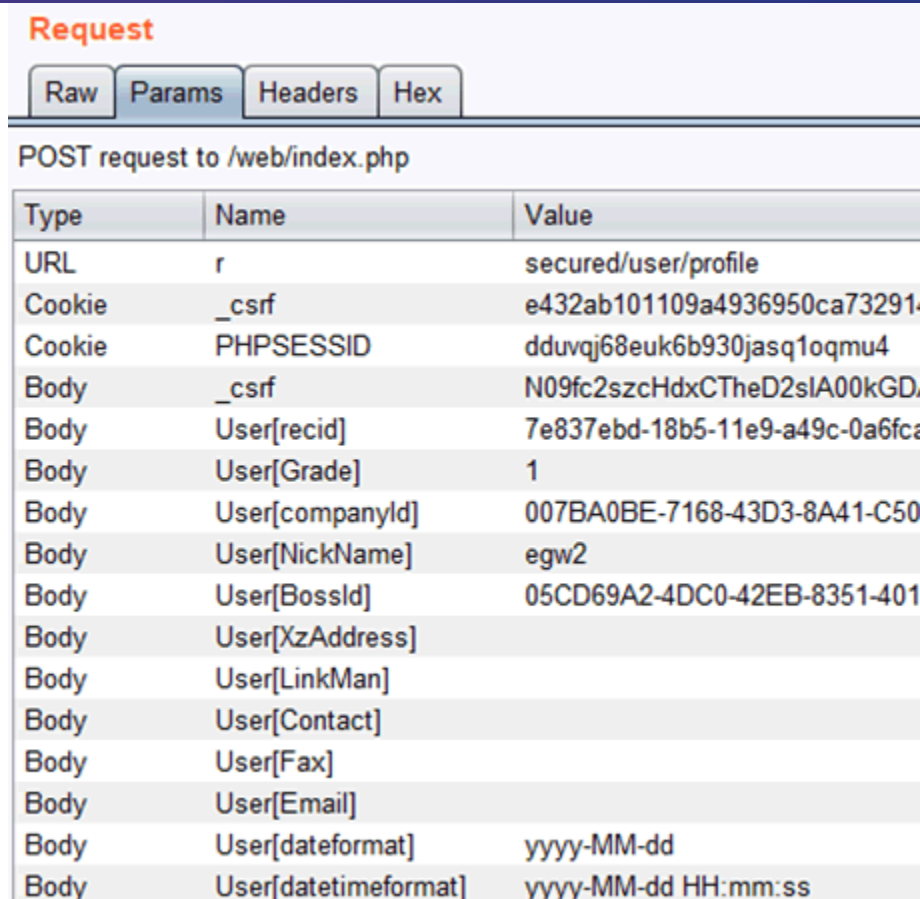
Mass Assignment: Gator Watches (2019)

Source

https://www.pentestpartners.com/security-blog/gps-watch-issues-again/

Problème

- User[Grade] définit le rôle de l'utilisateur
- Transmettre un autre User[Grade] lors d'une édition du profil utilisateur modifiait son rôle. 0 = admin.

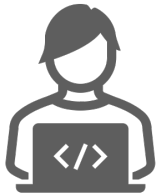


API8:2019 Injection

Les failles d'injection, telles que SQL, NoSQL, Injection de commande, Log4Shell, etc., surviennent quand des données non validées sont transmises à un interpréteur au sein d'une commande. Un attaquant peut modifier les données afin de piéger l'interpréteur et exécuter une commande sans autorisation.

API8:2019 Injection

Requête



POST /user/12

Authorization: af23b

{name: “Jean Doh”,
email: “**${jndi:ldap://xliic/willnotwork}@etc.etc**”}

Backend API

Décision AuthN

Est-ce que le token **af23b** est valide?
Qui utilise le token **af23b**?
=> **user56**

Décision AuthZ

Est-ce que **user56** peut effectuer un **POST** sur les adresses **/user/{userId}**?

Est-ce que **user56** a le droit de voir les informations de l’utilisateur **12**?

API8:2019 Injection

Adresser Injection

- Valider le type de chaque paramètre (nombre, chaîne de caractères, tableau etc.)
- Valider le format de chaque paramètre (e.g. int32, int64 etc.)
- Faire la même chose pour les headers et les tokens
- Mettre des contraintes sur chaque paramètre !
  - Min/Max pour les nombres
  - Pattern et taille des chaînes
  - Nombre d'éléments d'un tableau.

Injection: Log4Shell (2021)

Protecting your APIs against Log4Shell with 42crunch:

https://42crunch.com/protecting-your-apis-against-log4shell-with-42crunch/

Adresser Injection avec 42Crunch

Audit
- Valide les schémas de requête et encourage à leur spécification complète
- Alerte sur les paramètres non contraints (requête, chemin, header, etc.)

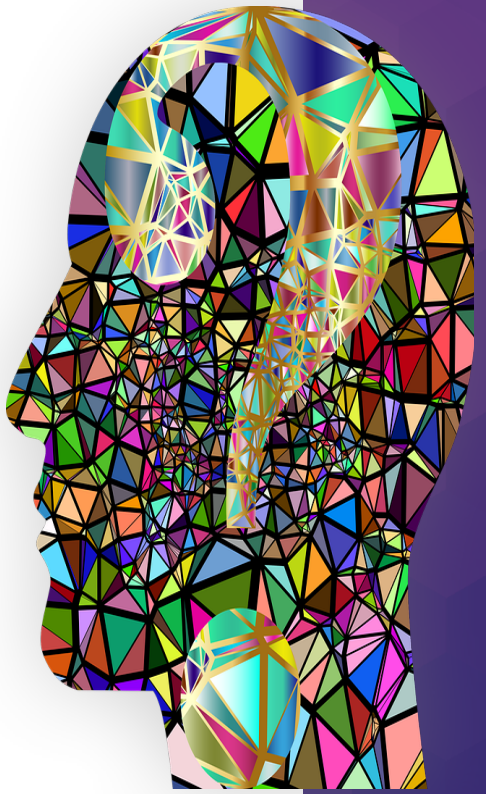
Scan
- Teste l'API avec des formats de données qui dévie des contraintes pour valider la couche de validation

Protection
- Validation systématique de chaque donnée contre le contrat
- Bloquage systématique des requêtes qui ne respectent pas les contraintes

Sondage - Question 2:

Comment protégez-vous vos APIs?

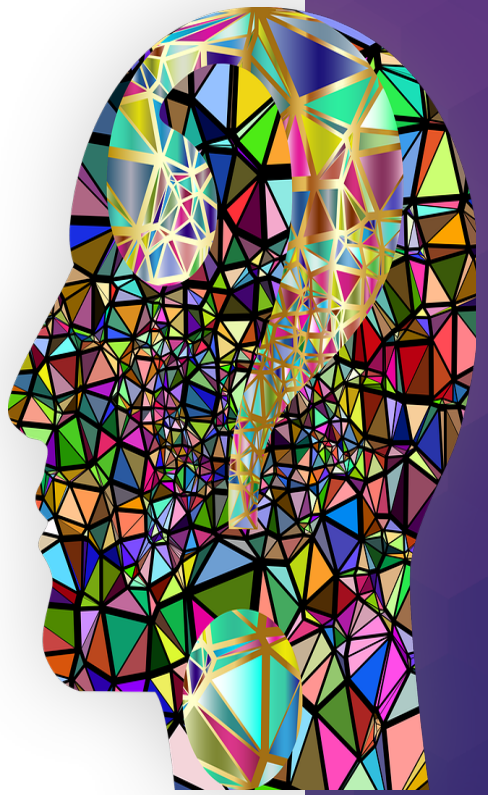
1. WAF
2. API Gateway
3. API micro-firewall dédié
4. Protection de type Runtime (RASP, IAST)



Sondage - Question 2:

Comment protégez-vous vos APIs?

WAF	17%
API Gateway	83%
API micro-firewall dédié	0%
Protection de type Runtime (RASP, IAST)	0%



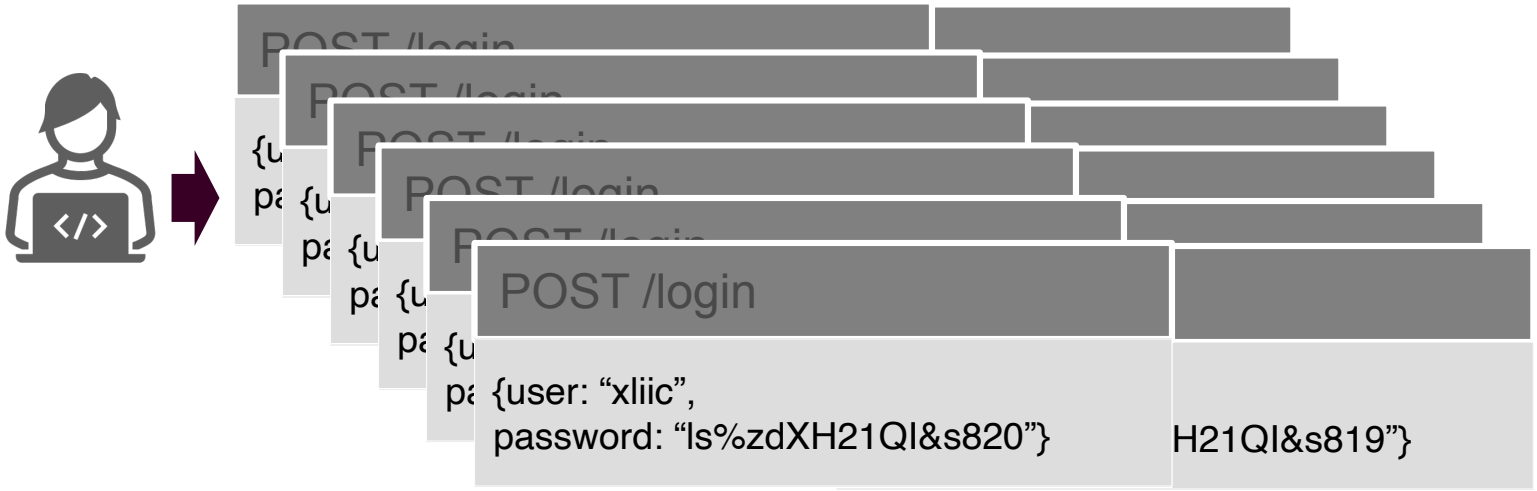
Problèmes de gouvernance et d’opération

API4:2019 Lack of Resources & Rate Limiting

Trop souvent les APIs n'imposent pas de restriction sur la taille ou le nombre de ressources qui peuvent être demandées par un utilisateur.

Cela peut avoir un impact sur la performance qui peut mener à un Déni de Service, mais laisse aussi la porte ouverte à de l'exfiltration massive de données ou du brute force.

API4:2019 Lack of Resources & Rate Limiting



API4:2019 Lack of Resources & Rate Limiting

Adresser Lack of Resources & Rate Limiting

- Quel que soit le problème que l'API a, l'absence de rate limiting l'aggrave
- Définir des réponses HTTP 429 (Too Many Requests)
  - Spéficier précisément les endpoints, clients, ou méthodes qui doivent être rate-limitées
  - *Porter une attention particulière à ceux d'authentification et ceux coûteux en resources !*

Lack of Resources & Rate Limiting : Facebook (2018)

Source

https://appsecure.security/blog/we-figured-out-a-way-to-hack...

Problème

- Facebook.com est rate-limité
- Beta.facebook.com ne l'était pas, permettant une attaque « brute force » sur la page de code de réinitialisation du mot de passe

Adresser Lack of Resources&Rate Limiting avec 42Crunch

Audit
- Valide la définition de réponses HTTP 429 (« Too Many Requests »)
- Pousse à la définition des contraintes (taille des chaînes, nombre d'éléments dans un tableau etc.)

Scan
- Envoi de données incorrectes, hors de ce qui est attendu

Protection
- Bloquage systématique des requêtes qui ne respectent pas les contraintes
- Possibilité d'ajouter du rate-limiting en Security-as-Code
- Validation du Content-Type

API7:2019 Security Misconfiguration

Souvent le résultat de configurations par défaut soit non sécurisées ou incomplète. Cela va du stockage cloud en libre accès, aux headers sécurisés HTTP mal configurés, en passant par des messages d'erreur verbeux, etc.

Adresser Security Misconfiguration

- Patcher régulièrement les systèmes des APIs
- Mettre en place du TLS (1.2 minimum, 1.3 idéal)
- CORS mal mis ou manquant
- Gérer toutes les exceptions au niveau de l'API
- Limiter aux verbes importants

API9:2019 Improper Assets Management

Un attaquant trouve un autre environnement d'API (par exemple, staging, beta, alpha etc.) qui ne sont pas aussi bien protégées qu'une API de production et s'en sert pour lancer une attaque.

Adresser Improper Assets Management

- Garder un inventaire de toutes les APIs et de leur définition
- Limiter l'accès de tout ce qui ne doit pas être public
- Limiter l'accès aux données de production et bien ségréguer données de QA/UAT/dev et production
- Utiliser un firewall pour bloquer l'accès à des destinations non autorisées
- Décommissionner les vieilles APIs, ou s'assurer qu'elles sont régulièrement patchées

API10:2019 Insufficient Logging & Monitoring

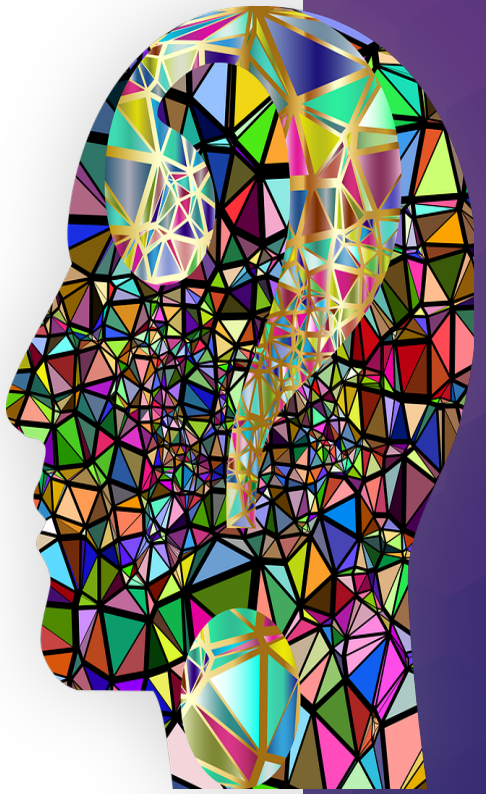
L’absence de logging ou de monitoring d’événements, couplé au défaut d’intégration avec un SIEM permet aux attaquants de mener des attaques durables et de pivoter vers d’autres systèmes

Adresser Insufficient Logging & Monitoring

- Penser aux logs dès la phase de conception de l’API
- Utiliser différents niveaux de verbosité (FATAL, ERROR, WARN...)
- Contextualiser les messages de log (et pas seulement « IntegerBoundaryException ») pour qu’ils peuvent être compris par un humain, tout en étant parsables
- Ne *jamais* logger de PII.
- Ne *jamais* logger de donnée non validée !
- Les événements doivent être envoyés à un SIEM

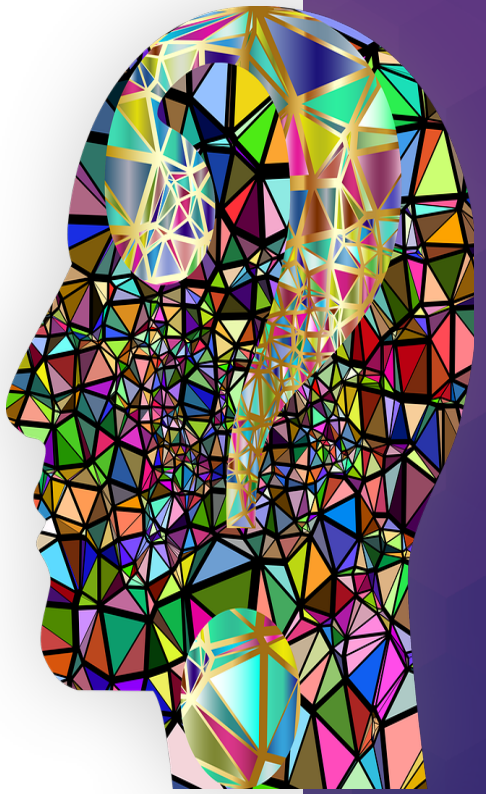
Question de l'audience:

Quelle est la différence entre l'OWASP Top 10 et l'OWASP API Security Top 10?



Question de l'audience:

C’est le serveur de resources qui doit se protéger contre BOLA et non la Gateway?



Plus d'Informations

OWASP API Security Top 10 https://42crunch.com/owasp-api-security-top-10-webinar-series/

Lille, Grand Palais 7-9 Juin

FIC
International Cybersecurity Forum

France API, Paris 14 Juin

FRANCE API

OpenAPI Editor - Free Download
https://42crunch.com/resources-free-tools/

APIsecurity.io Newsletter
https://apisecurity.io/

APIsecurity.io