PROGRAMACIÓN ORIENTADA A PRUEBAS

Revisado por: Ing. Miguel Cardona.

Aarón Hernández García.

Actividad: OWASP Cheat Sheet Series

09/04/2025

1. Authentication Cheat Sheet

Objetivo Principal

Este Cheat Sheet está enfocado en asegurar que el proceso de autenticación de una aplicación se realice de forma segura, evitando vulnerabilidades asociadas a la gestión de identidades y credenciales.

Aspectos y Recomendaciones Clave

- **Gestión de Contraseñas:**
    - **Hashing Seguro:** Se recomienda utilizar algoritmos de hashing fuertes (por ejemplo, bcrypt, Argon2 o PBKDF2) para almacenar las contraseñas de los usuarios, de modo que, en caso de una filtración, el atacante no tenga acceso directo a las contraseñas.
    - **Salting:** Incorporar una "sal" única para cada contraseña ayuda a mitigar ataques basados en tablas de hash precalculadas (rainbow tables).
- **Políticas de Contraseñas:**
    - **Requisitos de complejidad:** Establecer reglas mínimas (longitud, combinación de caracteres, etc.) para asegurar que las contraseñas tengan suficiente fortaleza.
    - **Caducidad y reutilización:** Implementar políticas sobre el tiempo de uso y la reutilización de contraseñas para incentivar a los usuarios a cambiarlas periódicamente.
- **Autenticación Multifactor (MFA):**
    - **Adicional a la contraseña:** Utilizar un segundo factor (como un token, SMS, aplicación de autenticación o biometría) para validar la identidad del usuario, reduciendo el riesgo de acceso no autorizado.
- **Gestión de Sesiones:**
    - **Expiración y revocación:** Las sesiones deben tener una expiración automática y mecanismos para revocarlas (por ejemplo, en caso de que se sospeche un acceso indebido).
    - **Cookies Seguras:** Usar atributos como HttpOnly, Secure y SameSite en las cookies para prevenir el robo de sesiones mediante ataques XSS o CS

2. API Security Cheat Sheet

Objetivo Principal

Este Cheat Sheet aborda la seguridad en las APIs, especialmente las RESTful, proporcionando lineamientos para proteger la comunicación, el acceso y la integridad de los datos que se transmiten y procesan.

Aspectos y Recomendaciones Clave

- **Transmisión Segura:**
    - **HTTPS/TLS:** Es fundamental cifrar todas las comunicaciones entre el cliente y la API usando HTTPS para prevenir la interceptación de datos sensibles.
- **Autorización y Autenticación:**
    - **Tokens Seguros:** Emplear tokens (por ejemplo, JWT) para autenticar las peticiones. Es importante validar y gestionar estos tokens de manera adecuada.
    - **Scopes y Roles:** Definir niveles de acceso y roles para limitar la exposición de datos y funcionalidades según el tipo de usuario o aplicación.
- **Validación y Saneamiento de Entradas:**
    - **Filtrado de Parámetros:** Todas las entradas (ya sea por query strings, headers o el cuerpo del mensaje) deben ser validadas y saneadas para prevenir inyecciones de código o datos maliciosos.
- **Rate Limiting y Protección Contra Abuso:**
    - **Límites de Solicitudes:** Implementar límites en la cantidad de peticiones por unidad de tiempo para evitar ataques de denegación de servicio (DoS) o abusos.
- **Buenas Prácticas de Diseño:**
    - **Versionado de API:** Permite evolucionar la API sin interrumpir el servicio a los consumidores.
    - **Documentación y Error Handling:** Proveer respuestas de error claras y mantener documentadas todas las funciones de la API para facilitar la identificación y resolución de problemas de seguridad.

3. Input Validation Cheat Sheet

Objetivo Principal

Este Cheat Sheet se centra en asegurar que todos los datos que ingresa un usuario sean validados, transformados o filtrados antes de ser utilizados en el sistema. De esta forma se busca prevenir ataques basados en la manipulación de la entrada, como inyecciones SQL, XSS, y otros tipos de explotación.

Aspectos y Recomendaciones Clave

- **Validación del Lado del Servidor:**
    - **No Confiar en el Lado del Cliente:** Toda la validación que se realice en el cliente debe ser replicada o reforzada en el servidor, ya que el cliente puede ser manipulado.
- **Listas Blancas vs. Listas Negras:**
    - **Listas Blancas (Whitelist):** Es preferible definir exactamente qué tipo de datos se espera (por ejemplo, solo letras y números) en vez de intentar bloquear ciertos patrones maliciosos.
    - **Rechazo de Datos Mal Formados:** Si la entrada no se ajusta a los criterios predefinidos, debe ser rechazada inmediatamente.
- **Saneamiento de Datos:**
    - **Escape de Caracteres Especiales:** Aplicar técnicas de escape a los caracteres especiales que puedan tener significados particulares en consultas SQL, HTML o scripts.
    - **Uso de ORM o Librerías de Saneamiento:** En el caso de trabajar con bases de datos, utilizar herramientas que ayuden a prevenir inyecciones (como parámetros en consultas o ORMs) puede reducir significativamente el riesgo.
- **Validación de Formato y Longitud:**
    - **Restricciones de Entrada:** Limitar el tamaño de las entradas y el formato (por ejemplo, expresiones regulares para validar emails o números de teléfono) para evitar la entrada de datos excesivamente largos o maliciosos.
- **Feedback Adecuado:**
    - **Mensajes de Error Seguros:** Proveer mensajes que sean útiles para el usuario, sin revelar detalles técnicos que puedan ayudar a un atacante a comprender la lógica interna del sistema.

Documentación de Recomendaciones OWASP Aplicadas

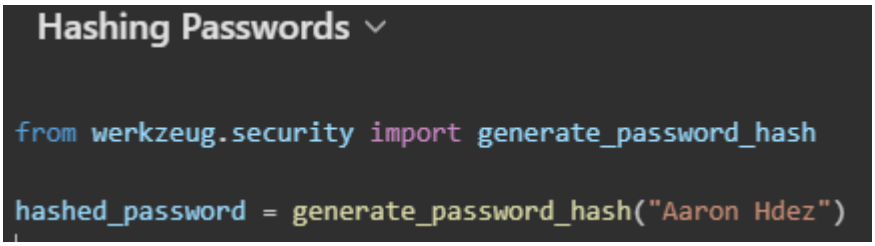
1. **Almacenamiento Seguro de Contraseñas (Authentication Cheat Sheet)**

Descripción:

Implementamos hashing seguro con bcrypt para almacenar contraseñas de los usuarios en la base de datos.

Importancia:

El almacenamiento en texto plano representa un riesgo grave: si un atacante accede a la base de datos, tendría acceso directo a las contraseñas. bcrypt permite aplicar un hash lento y seguro, lo que hace que ataques de fuerza bruta o diccionario sean poco prácticos.



2. **Autenticación con JWT y Expiración (Authentication + API Security Cheat Sheet)**

Descripción:

Se implementó autenticación con JSON Web Tokens (JWT), incluyendo tiempo de expiración.

Importancia:

El uso de JWT asegura que cada solicitud proviene de un usuario autenticado. Además, el tiempo de expiración impide que tokens comprometidos tengan validez indefinida.

JWT Token Generation

```
import jwt
import datetime

token = jwt.encode({
    'user': 'usuario_aaronhdez',
    'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=30)
}, secret_key, algorithm="HS256")
```

3. **Validación de Entrada del Usuario (Input Validation Cheat Sheet)**

Descripción:

Se aplicó validación de entradas en todos los formularios del sistema. Solo se permiten los valores esperados (listas blancas).

Importancia:

Los ataques como XSS, SQL Injection y otras inyecciones comienzan cuando el sistema acepta entrada maliciosa. Validar las entradas evita que el atacante manipule el comportamiento del sistema.