Clase N° 15: Introducción a la seguridad en la web

Crombie

Clase N° 15: Agenda

- Que es la seguridad?
- Importancia
- Áreas de la seguridad
- Amenazas más comunes
- DDOS
- SQL Injection
- HTML Injection
- Laboratorio: simulando DDOS y SQL Injection

Crombie

Seguridad en la web

Conjunto de prácticas, tecnologías y estrategias diseñadas para proteger sitios web, aplicaciones web y sus datos contra amenazas maliciosas, vulnerabilidades y accesos no autorizados. Su objetivo principal es garantizar la confidencialidad, integridad y disponibilidad de la información y los servicios web

- Confidencialidad: Garantizar que solo las personas autorizadas puedan acceder a la información.
- Integridad: Asegurar que los datos no sean alterados o manipulados sin autorización.
- Disponibilidad: Asegurar que los servicios y datos estén disponibles para los usuarios legítimos cuando los necesiten.

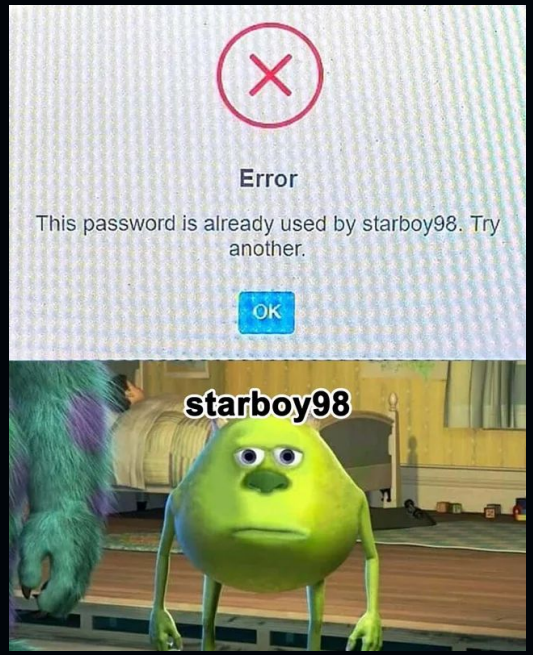
Crombie

Por qué es importante?

- Proteger a los usuarios: Evitar robos de identidad, fraudes o exposición de información personal.
- Preservar la reputación: Un sitio comprometido puede dañar la confianza del cliente y la marca.
- Cumplir con normativas: Como GDPR, CCPA, PCI DSS, entre otras.
- Evitar pérdidas económicas: Un ataque puede causar pérdidas significativas debido a interrupciones del servicio o robo de datos.

Crombie

La seguridad empieza por nosotros



Crombie

La seguridad empieza por nosotros

Tu responsabilidad como desarrollador no se limita a escribir código funcional, sino también a garantizar que el sistema sea resiliente frente a amenazas. Esto implica adoptar buenas prácticas, aprender continuamente y colaborar con otros para proteger a los usuarios y la organización. Recuerda que "una aplicación insegura no solo daña a los usuarios, también afecta tu reputación como desarrollador."

Áreas clave de la seguridad

- Seguridad en las aplicaciones:
  - Prevención de vulnerabilidades como SQL Injection, XSS, y CSRF.
  - Uso de prácticas de desarrollo seguro.

- Seguridad en los datos:
  - Encriptación de datos en tránsito (HTTPS/TLS).
  - Encriptación de contraseñas
  - Protección contra fugas de datos.

- Gestión de accesos:
  - Dar únicamente los permisos necesarios
  - Implementación de controles de autenticación y autorización.
  - Uso de métodos como MFA (Autenticación Multifactor).

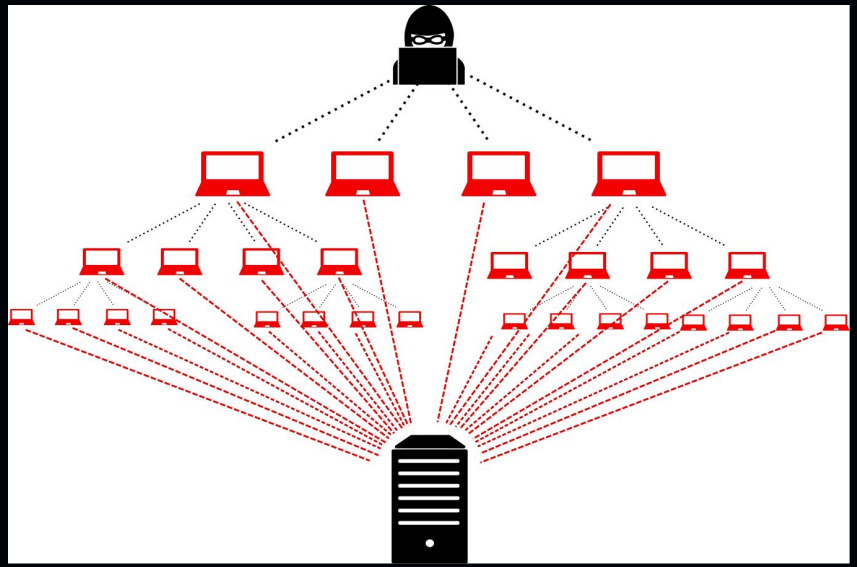
“Esto solo pasa en peliculas”

Crombie

Amenazas más comunes

Amenazas más comunes: DDoS

Un ataque DDoS (Distributed Denial of Service) es un intento malicioso de interrumpir el funcionamiento normal de un servidor, servicio o red al sobrecargarlo con una avalancha de tráfico falso proveniente de múltiples fuentes.



Crombie

Amenazas más comunes: DDoS

¿Cómo funciona?

- Distribución del ataque: Se utiliza una red de computadoras comprometidas (botnet), que puede estar compuesta por miles o incluso millones de dispositivos infectados.
- Inundación de solicitudes: Estas computadoras envían una cantidad masiva de solicitudes al servidor objetivo.
- Saturación: El servidor no puede procesar las solicitudes legítimas porque está ocupado manejando las falsas, lo que genera caídas o lentitud extrema.

Consecuencias

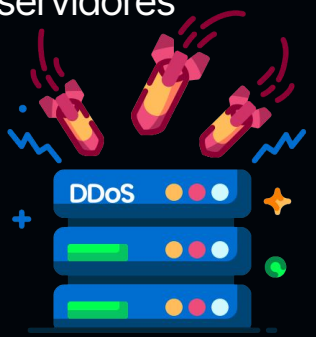
- Pérdida de disponibilidad del servicio.
- Daño a la reputación de la organización.
- Impacto financiero por interrupciones prolongadas.

Crombie

Amenazas más comunes: DDoS

¿Cómo prevenirlos?

- Usar un Firewall de Aplicación Web (WAF):Filtra y bloquea tráfico malicioso antes de que llegue a tu servidor.
  - Ejemplos: AWS WAF, Cloudflare WAF, Azure Application Gateway.
- Implementar una Red de Entrega de Contenidos (CDN): Distribuye el tráfico a través de una red global de servidores, reduciendo la carga en tu servidor principal.
  - Ejemplos: Cloudflare, Akamai, Fastly.
- Habilitar el Rate Limiting para restringir la cantidad de solicitudes que un usuario o IP puede hacer en un período de tiempo.
- Configurar un Balanceador de Carga para distribuir el tráfico entre múltiples servidores para evitar sobrecarga.



Crombie

Amenazas más comunes: Inyección SQL

Un ataque de inyección SQL ocurre cuando un atacante inserta o "inyecta" código SQL malicioso en un formulario o entrada de datos de una aplicación web, con el fin de manipular las consultas SQL que se envían a la base de datos. Esto puede permitir al atacante:

- Acceder a datos confidenciales (como contraseñas o información personal).
- Modificar o eliminar datos en la base de datos.
- Ejecutar comandos arbitrarios en el servidor de base de datos.

Crombie

Amenazas más comunes: Inyección SQL

¿Cómo funciona?

- Entrada maliciosa: El atacante introduce código SQL no esperado a través de un campo de entrada, como un formulario de inicio de sesión o búsqueda
-

Ejemplo: En lugar de introducir un nombre de usuario válido, el atacante escribe:

```
' OR '1'='1'
```

Resultado: La condición '1'='1' siempre es verdadera, lo que permite que el atacante inicie sesión sin una contraseña válida, o incluso obtenga acceso completo a la base de datos.

Crombie

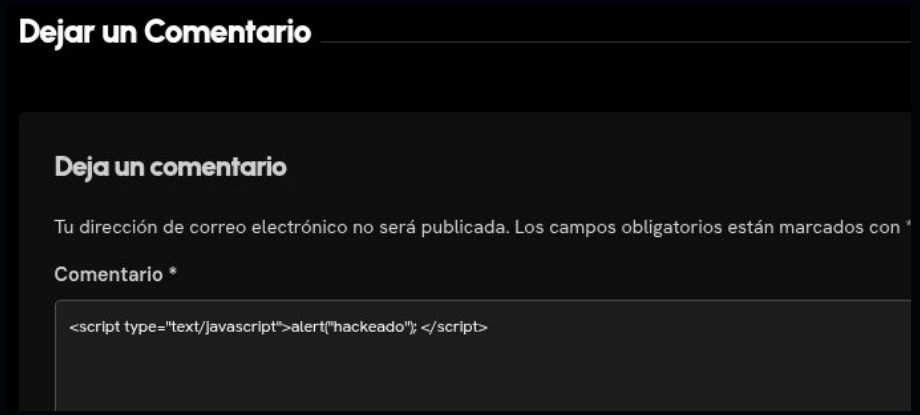
Amenazas más comunes: Inyección SQL

¿Cómo prevenir estos ataques?

- Uso de consultas preparadas: Utilizar parámetros en las consultas SQL en lugar de concatenar directamente los datos del usuario.
- Validación de entradas: Filtrar y validar los datos introducidos por los usuarios.
- Escapar caracteres especiales: Escapar adecuadamente los caracteres especiales en las entradas de los usuarios.
- Principio de menor privilegio: Asegurarse de que las cuentas de base de datos solo tengan los permisos necesarios.

Amenazas más comunes: HTML Injection

Es una vulnerabilidad de seguridad que ocurre cuando un atacante puede inyectar código HTML arbitrario en una página web, haciendo que se muestre contenido no deseado o potencialmente peligroso para los usuarios. Esto sucede cuando las entradas proporcionadas por los usuarios no son correctamente validadas o sanitizadas antes de ser incorporadas al HTML que se envía al navegador.



Crombie

Pongamos en práctica los ataques más conocidos

Como simular un ataque DDoS

Utilicemos las aplicaciones de ecommerce que construyeron como conejillo de india de un ataque DDoS.

- Inicie el servidor local
- Envíe por slack la segunda URL, correspondiente a Network:

```
> api-resegura@0.1.0 dev
> next dev --turbopack

   ▲ Next.js 15.1.2 (Turbopack)
   - Local:        http://localhost:3000
   - Network:      http://192.168.1.32:3000
```

Haremos una prueba de carga para ver cómo reaccionan sus servicios

Crombie

Inyección SQL: práctica

Usaremos una api rest, con un endpoint de login, para intentar saltearnos la seguridad sin usar usuario y contraseña.

Nuestra cuenta con el siguiente endpoint:

- http://localhost:3000/api/login

Para que funcione debemos hacer un POST, con usuario y contraseña. La api validará si el usuario existe usando una base de datos de Mysql.

Si los datos son incorrectos, obtendremos esta respuesta:


```
{
  "error": "Invalid username or password"
}
```


De lo contrario, se iniciará la sesión y veremos esto:


```
{
  "message": "Login successful",
  "user": [
    {
      "id": 1,
      "username": "admin",
      "password": "9928842",
      "email": "lautaro@gmail.com",
      "createdAt": "2024-12-23T00:19:41.943Z",
```

“Ningún sistema es seguro”

“Ningún sistema es seguro”

- **Complejidad del Software:** Más código significa más posibilidades de errores y vulnerabilidades.
- **Factores Humanos:** Errores de configuración, ingeniería social y malas prácticas son puertas abiertas para atacantes.
- **Evolución de Amenazas:** Los atacantes siempre están un paso adelante con nuevas técnicas y herramientas.
- **Dependencia de Terceros:** Sistemas inseguros, bibliotecas desactualizadas y servicios externos amplían los riesgos.
- **Limitaciones Físicas y Hardware:** Vulnerabilidades en dispositivos y ataques físicos comprometen la seguridad.
- **Restricciones de Recursos:** Presiones de tiempo, costos y mantenimiento insuficiente debilitan los sistemas.
- **Proactividad vs. Reactividad:** La seguridad responde a los ataques, pero los atacantes innovan constantemente.

Recomendaciones

1. NUNCA confiar en la entrada del usuario
2. El eslabon mas debil de un sistema es el usuario
3. Es más fácil prevenir que mitigar
4. Aplicar el principio de menor privilegio
5. Cifrar las contraseñas

Mas informacion: https://owasp.org/

OWASP (Open Web Application Security Project) es una fundación sin fines de lucro que se dedica a mejorar la seguridad del software. Es ampliamente reconocida por proporcionar recursos, herramientas y guías que ayudan a los desarrolladores, arquitectos y equipos de seguridad a identificar y mitigar vulnerabilidades en aplicaciones web y sistemas relacionados