Determinación del nivel de seguridad requerido por aplicaciones (Puesta en producción segura)

Fuentes abiertas para el desarrollo seguro

Inteligencia de fuentes abiertas u «Open Source Intelligence» (OSINT) hace referencia al conocimiento recopilado a partir de fuentes de acceso público. El proceso incluye la búsqueda, selección y adquisición de la información, así como un posterior procesado y análisis de la misma con el fin de obtener conocimiento útil y aplicable en distintos ámbitos. (Información obtenida de INCIBE).

Existen multitud de fuentes abiertas a partir de las cuales se puede obtener información relevante, entre las que destacan:

- Medios de comunicación: revistas, periódicos, radio, etc.
- Información pública de fuentes gubernamentales.
- Foros, redes sociales, blogs, wikis, etc.
- Conferencias, simposios, «papers», bibliotecas online, etc.

Listas de riesgos de seguridad habituales: OWASP Top Ten (web y móvil)

OWASP es un proyecto de código abierto dedicado a determinar y

combatir las causas que hacen que el software sea inseguro. La Fundación OWASP es un organismo sin ánimo de lucro que apoya y gestiona los proyectos e infraestructura de OWASP.

Top 10 Web Application Security Risks (https://owasp.org/www-project-top-ten/):

1. **Injection**. Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
2. **Broken Authentication**. Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
3. **Sensitive Data Exposure**. Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
4. **XML External Entities (XXE)**. Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
5. **Broken Access Control**. Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as

access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

6. **Security Misconfiguration**. Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.
7. **Cross-Site Scripting (XSS)**. XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
8. **Insecure Deserialization**. Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.
9. **Using Components with Known Vulnerabilities**. Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
10. **Insufficient Logging & Monitoring**. Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most

breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

Top 10 Mobile Risks – Final List 2016 (https://owasp.org/www-project-mobile-top-10/)

- M1: Improper Platform Usage
- M2: Insecure Data Storage
- M3: Insecure Communication
- M4: Insecure Authentication
- M5: Insufficient Cryptography
- M6: Insecure Authorization
- M7: Client Code Quality
- M8: Code Tampering
- M9: Reverse Engineering
- M10: Extraneous Functionality

Requisitos de verificación necesarios asociados al nivel de seguridad establecido

Requisitos de verificación detallada (https://owasp.org/www-pdf-archive/Est%C3%A1ndar_de_Verificaci%C3%B3n_de_Seguridad_en_Aplicaciones_3.0.1.pdf):

- V1. Arquitectura, diseño y modelado de amenazas
- V2. Autenticación
- V3. Gestión de sesiones
- V4. Control de acceso
- V5. Manejo de entrada de datos maliciosos
- V7. Criptografía en el almacenamiento
- V8. Gestión y registro de errores
- V9. Protección de datos
- V10. Comunicaciones
- V11. Configuración de seguridad HTTP
- V13. Controles Maliciosos

- V15. Lógica de negocio
- V16. Archivos y recursos
- V17. Móvil
- V18. Servicios Web (Nuevo en 3.0)
- V19. Configuración (nuevo en 3.0)

Comprobaciones de seguridad a nivel de aplicación: ASVS (Application Security Verification Standard)

El proyecto del estándar de verificación de seguridad de aplicaciones (ASVS) de OWASP proporciona información para probar los controles técnicos de seguridad de las aplicaciones web y también proporciona a los desarrolladores una lista de requisitos para un desarrollo seguro.

ASVS tiene dos objetivos principales:

- ayudar a las organizaciones en el desarrollo y mantenimiento aplicaciones seguras
- permitir la alineación entre las necesidades y ofertas de los servicios de seguridad, proveedores de herramientas de seguridad y consumidores

Los requisitos se desarrollaron con los siguientes objetivos en mente:

- Usarlo como métrica: proporcione a los desarrolladores y propietarios de aplicaciones un criterio con el que evaluar el grado de confianza que se puede depositar en sus aplicaciones web,
- Utilizarlo como guía: proporcionar orientación a los desarrolladores de controles de seguridad sobre qué incorporar en los controles de seguridad para satisfacer los requisitos de seguridad de las aplicaciones
- Usarlo durante la adquisición: proporciona una base para especificar los requisitos de verificación de seguridad

de la aplicación en los contratos.