

# Índice

<b>Capítulo 1. Introducción al desarrollo seguro</b> .....	<b>15</b>
1.1 Propiedades del software seguro .....	17
1.2 Principios de diseño seguro de aplicaciones .....	18
1.2.1 Minimizar el área de la superficie de ataque .....	19
1.2.2 Seguridad por defecto .....	19
1.2.3 Privilegios mínimos .....	19
1.2.4 Validación de datos de entrada .....	20
1.2.5 Defensa en profundidad .....	20
1.2.6 Control seguro de errores .....	20
1.2.7 Separación de funciones .....	21
1.2.8 Evitar la seguridad por oscuridad .....	21
1.3 Análisis de requisitos de seguridad .....	21
<b>Capítulo 2. Aspectos fundamentales de desarrollo seguro</b> .....	<b>23</b>
2.1 Controles proactivos .....	24
2.2 OWASP (Open Web Application Security Project) .....	25
2.3. OWASP Mobile Security Project .....	29
2.4 Controles proactivos OWASP .....	31
2.4.1 Verificación de la seguridad desde las primeras etapas de desarrollo .....	32
2.4.2 Validación de las entradas del cliente .....	32
2.4.3 Desbordamientos del búfer .....	35
2.4.4 Gestión de sesiones .....	35
2.4.5 Implementación de controles de acceso .....	36
2.4.6 Implementación de controles de identidad y autenticación .....	36
2.4.7 Autenticación por múltiples factores .....	37
2.4.8 Manejo de errores y excepciones .....	38
2.5 Ataques en aplicaciones web .....	39
2.5.1 Vectores de ataque .....	39
2.5.2 Cross-site scripting (XSS) .....	40
2.5.3 Cross-site request forgery (CSRF) .....	42
2.5.4 Seguridad en las redirecciones .....	44
2.6 SQL Injection: parametrización de las consultas en bases de datos .....	45
2.6.1 Introducción a SQL Injection .....	46
2.6.2 Problemas que pueden causar este tipo de ataques .....	47
2.6.3 Ejemplo de inyección de SQL .....	48
2.6.4 Escapar caracteres especiales utilizados en las consultas SQL .....	50
2.6.5 Delimitación de los valores de las consultas .....	51
2.6.6 Uso de sentencias preparadas parametrizadas .....	52
2.6.7 Uso de procedimientos almacenados .....	53

2.7 Seguridad en AJAX .....	55
<b>Capítulo 3. Herramientas OWASP .....</b>	<b>63</b>
3.1 DefectDojo .....	63
3.2 SonarQube .....	69
3.2.1 El cuadro de mando de SonarQube .....	70
3.2.2 Issues por nivel de criticidad .....	72
3.2.3 Perfiles de calidad .....	73
3.2.4 Reglas SonarQube .....	76
3.2.5 Informes de seguridad en SonarQube .....	78
3.2.6 SonarQube Plugins .....	79
3.2.7 Vulnerabilidades más comunes y explotadas .....	83
3.3 Find Security Bugs .....	83
3.3.1 Inyección potencial de Android SQL .....	85
3.3.2 Abrir un socket sin cifrar .....	85
3.4 LGTM .....	86
3.5 OSS Index .....	89
3.6 Snyk .....	91
3.7 Otras herramientas de análisis estático .....	92
3.8 Checklist de seguridad .....	93
<b>Capítulo 4. Seguridad en aplicaciones Android .....</b>	<b>95</b>
4.1 Introducción al protocolo HTTPS .....	95
4.1.1 Conceptos básicos sobre certificados .....	95
4.1.2 Despliegues en producción .....	97
4.1.3 Certificado de servidor autofirmado .....	100
4.1.4 CA no encontrada dentro de la cadena de certificados .....	100
4.1.5 Configuración de seguridad .....	101
4.1.6 Actualización de proveedores criptográficos .....	102
4.1.7 Android Certificate Pinning .....	102
4.1.8 Cifrado extremo a extremo .....	104
4.1.9 Firmando una aplicación Android .....	104
4.2 Principios fundamentales de desarrollo en Android .....	106
4.2.1 Componentes en Android .....	107
4.2.2 Android Lint .....	109
4.3 Ingeniería inversa en Android .....	113
4.3.1 ADB (Android Debug Bridge) .....	114
4.3.2 Dex2jar .....	114
4.3.3 JD-GUI .....	116
4.3.4 jadx - Dex to Java decompiler .....	116
4.3.5 Apktool .....	118
4.3.6 Código smali y MobyLizer .....	121
4.3.7 Androwarn .....	123
4.3.8 Mobile Security Framework (MobSF) .....	125
4.3.9 ClassyShark .....	127
4.3.10 Drozer .....	128
4.3.11 QARK .....	134

4.3.12	SanDroid	137
4.3.13	Yaazhini	138
4.4	Buenas prácticas de desarrollo seguro en Android	139
4.4.1	Seguridad en AndroidManifest.xml	140
4.4.2	Modelo de permisos en Android	142
4.4.3	Asegurando la capa de aplicación	143
4.4.4	Evitar almacenar datos confidenciales en el dispositivo	144
4.4.5	Uso adecuado del componente WebView	145
4.4.6	Usar método POST para el envío de datos confidenciales	146
4.4.7	Validar los certificados SSL/TLS	147
4.4.8	Restricción de uso de la aplicación a determinados dispositivos	147
4.4.9	Gestión de logs	148
4.4.10	Comprobar la conexión de red	149
4.4.11	Realizar operaciones de red en un hilo separado	149
4.4.12	Permisos de localización	151
4.4.13	Optimizar el código en Android y memoria caché	151
4.4.14	Implementación segura de proveedores de contenido	153
4.4.15	Almacenamiento de preferencias compartidas (SharedPreferences)	154
4.4.16	Almacenamiento seguro de preferencias	157
4.4.17	Almacenamiento en ficheros	160
4.4.18	Almacenamiento externo	160
4.4.19	Implementación segura de Intents	162
4.4.20	Implementación segura de servicios	163
4.4.21	Implementación segura de broadcast receivers	163
4.4.22	Implementación segura de content providers	164
4.4.23	Invocar actividades de forma segura	164
4.4.24	Implementar almacenamiento de datos seguro	165
4.4.25	Algoritmos criptográficos	170
4.4.26	Uso de java.util.String para almacenar información sensible	172
4.4.27	Proteger la configuración de la aplicación	172
4.4.28	Cifrado en base de datos SQLite	173
4.4.29	Optimización y ofuscación del código con ProGuard	174
4.5	Metodología OASAM	177
<b>Capítulo 5. Seguridad en proyectos NodeJS</b>		<b>179</b>
5.1	Introducción a NodeJS	179
5.2	Modelo Event-Loop	181
5.3	Gestión de paquetes	182
5.4	Programación asíncrona	184
5.5	Problema del código piramidal	185
5.6	Módulo para administrar el sistema de archivos	186
5.7	Módulo http	188
5.8	Utilización del Middleware Express	190
5.8.1	Middleware de nivel de aplicación	190
5.8.2	Middleware de nivel de direccionamiento	191
5.8.3	Middleware de terceros	191
5.9	Autenticación en NodeJS	192
5.9.1	Auth0	192
5.9.2	PassportJS	192

<b>5.10 OWASP top 10 en NodeJS</b> .....	<b>193</b>
5.10.1 OWASP NodeGOAT .....	193
5.10.2 Inyección de código .....	196
5.10.3 Función eval .....	197
5.10.4 Ataque de denegación de servicio .....	198
5.10.5 Uso de patrones y expresiones regulares .....	198
5.10.6 Acceso al sistema de ficheros .....	200
5.10.7 Inyección de SQL .....	202
5.10.8 Inyección de NoSQL .....	203
5.10.9 Inyección de logs .....	205
5.10.10 Gestión de la sesión y autenticación .....	205
5.10.11 Protegiendo credenciales de usuario .....	206
5.10.12 Tiempo de espera de sesión y protección de cookies .....	208
5.10.13 Secuestro de sesión (Session hijacking) .....	212
5.10.14 Módulo helmet .....	213
5.10.15 Cross-site scripting (XSS) .....	216
5.10.16 Referencias de objetos directos inseguros .....	218
5.10.17 Mala configuración de seguridad .....	219
5.10.18 Deshabilitar fingerprinting .....	221
5.10.19 Exposición de datos sensibles .....	222
5.10.20 Configurando SSL/TLS .....	222
5.10.21 Forzar peticiones HTTPS .....	225
5.10.22 Falta de control de acceso .....	227
5.10.23 Redirecciones no validadas .....	228
5.10.24 Denegación de servicio mediante expresiones regulares .....	229
5.10.25 Validar datos de entrada con validator .....	230
5.10.26 Validar datos de entrada con express-validator .....	231
5.10.27 Configuración de cabeceras HTTP .....	233
5.10.28 Política de seguridad de contenido (CSP) .....	235
5.10.29 Cross-site request forgery (CSRF) .....	236
5.10.30 Ejecutar código JavaScript de forma aislada .....	238
5.10.31 Uso de componentes con vulnerabilidades conocidas .....	239
5.10.32 NodejsScan .....	242
<b>Capítulo 6. Seguridad en proyectos Python</b> .....	<b>245</b>
6.1 Componentes inseguros en Python .....	248
6.2 Validación incorrecta de entrada/salida .....	249
6.3 Función eval() .....	250
6.4 Serialización y deserialización de datos con pickle .....	253
6.5 Ataques de inyección de entrada .....	257
6.5.1 Inyección de comandos .....	258
6.5.2 Inyección de SQL .....	264
6.6 Acceso seguro al sistema de archivos y ficheros temporales .....	266
6.7 Inyección de XSS .....	268
6.8 Inyección de SSTI .....	270
6.9 Servicios para comprobar la seguridad de proyectos Python .....	273
6.9.1 Pyup .....	273
6.9.2 LGTM en proyectos Python .....	275

6.9.3	Sanitización de las URL	275
6.9.4	Uso de un algoritmo criptográfico roto o débil	276
6.9.5	Peticiones con requests sin validación de certificado	276
6.9.6	Uso de la versión insegura SSL/TLS	277
6.9.7	Deserialización de entrada no confiable	277
6.9.8	Vulnerabilidades de XSS	278
6.9.9	Exposición de información a través de una excepción	279
6.9.10	Conexión con hosts remotos mediante SSH utilizando Paramiko	280
6.10	Análisis estático de código Python	280
6.10.1	Python Taint	281
6.10.2	Bandit	282
6.10.3	Hawkeye	286
6.10.4	DLint	288
6.11	Gestión de dependencias	289
6.11.1	Instalación de dependencias	290
6.11.2	Requires.io	291
6.11.3	Safety	292
6.11.4	Paquetes maliciosos en PyPI	292
6.12	Python code checkers	293
6.12.1	Pyflakes	293
6.12.2	PyLint	294
6.13	Escáner de seguridad de aplicaciones web	295
6.13.1	WAScan	295
6.13.2	SQLmap	296
6.13.3	XSScrapy	297
6.14	Seguridad en Django	298
6.14.1	Protección ante ataques XSS	299
6.14.2	Protección ante ataques CSRF	299
6.14.3	Protección de inyección de SQL	300
6.14.4	Protección de clickjacking	301
6.14.5	SSL/HTTPS	301
6.15	Otras herramientas de seguridad	302
6.15.1	Yosai	302
6.15.2	Flask-Security	303
6.15.3	OWASP Python Security Project	304
<b>Capítulo 7. Análisis estático y dinámico en aplicaciones C/C++</b>		<b>305</b>
7.1	Análisis estático	306
7.1.1	Code Analyzer	307
7.2	Análisis estático de código C/C++	308
7.2.1	Flawfinder	308
7.2.2	Clang	312
7.2.3	Uso de variables sin inicializar	312
7.2.4	Uso inseguro de funciones	314
7.2.5	RATS	315
7.2.6	Vulnerabilidad cadena de formato (format string)	315
7.2.7	Pscan para detectar vulnerabilidades format string	316
7.2.8	Buffer overflow	317

7.2.9	Tipos de heap overflow	320
7.2.10	Vulnerabilidad use after free	321
7.2.11	Dereference after free	322
7.2.12	Vulnerabilidad double free	323
7.2.13	Vulnerabilidad off by one	324
7.2.14	Vulnerabilidades race condition	325
7.2.15	Vulnerabilidad integer overflow	327
7.2.16	Uso de StackOverflow	328
7.3.	Análisis dinámico	329
7.3.1.	Análisis dinámico en C/C++ con Valgrind	330
7.4	Herramientas de análisis	333
<b>Capítulo 8. Metodologías de desarrollo</b>		<b>338</b>
8.1.	Metodologías de desarrollo de software seguro	338
8.1.1	Correctness by Construction (CbyC)	339
8.1.2	Security Development Lifecycle (SDLC)	341
8.1.3	Fases de la metodología SDLC	342
8.1.4	Vulnerabilidades en SDLC	346
8.1.5	Tipos de SDLC	349
8.1.5.1	Microsoft Trustworthy Computing SDL	349
8.1.5.2	CLASP	350
8.1.5.3	TSP-Secure	351
8.1.5.4	Oracle Software Security Assurance	352
8.1.5.5	Propuesta híbrida	352
8.1.6	Tipos de pruebas de seguridad SDLC	353
8.1.7	Conclusiones de ciclo de vida de desarrollo de software (SDLC)	355
8.2	Modelado de amenazas	356
8.2.1	Modelado de amenazas con STRIDE	358
8.3	Perspectiva del atacante	361
8.4	Patrones de ataque	362
8.5	OWASP Testing Framework y perfiles para pruebas de seguridad	363
8.6	OWASP Security Knowledge Framework (SKF)	365
8.7	Seguridad en ingeniería del software	374
8.8	Bibliografía y fuentes de información	376
8.9	Conclusiones	378
<b>Capítulo 9. Glosario de términos</b>		<b>380</b>