Fundación SADOSKY

Plan de estudios
Curso virtual para docentes de introducción al desarrollo seguro

---

El Plan de Estudios consta de los siguientes módulos.

1. Introducción a la Seguridad Informática

Objetivos

Introducir los conceptos y vocabulario básicos de la Seguridad Informática haciendo énfasis en los temas desde “Taxonomía de vulnerabilidades” en adelante.

Contenidos

Definición de seguridad informática. Definición de confidencialidad, integridad y disponibilidad. Sujetos (agentes) y objetos. Entorno o universo benigno y hostil (malicioso). Adversarios, amenazas, vulnerabilidades y ataques. Diferencias y semejanzas entre error (de programación o configuración) y vulnerabilidad. *Safety* y *security*. Seguridad interna y seguridad externa. Frontera del sistema, perímetro de seguridad y superficie de ataque. Política de seguridad (definición y ejemplos). Autenticación, autorización y auditoría. Credencial de acceso. Factores de autenticación. Permisos, modos o derechos de acceso. Control de acceso basado en identidad y roles. Programa confiable, benigno y hostil. Taxonomía de vulnerabilidades; base de vulnerabilidades *Common Vulnerabilities and Exposures* (CVE); *Common Weakness Enumeration* (CWE); *Common Vulnerability Scoring System* (CVSS). Análisis de amenazas. Caballos de Troya.

Bibliografía de referencia

- Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley 2002, ISBN 0-201-44099-7
- Morrie Gasser. *Building a Secure Computer System*. Van Nostrand Reinhold Co., 1988, ISBN 0-442-23022-2. Disponible en:

  https://www.researchgate.net/publication/242363259_Building_a_Secure_Computer_System

- Ross Anderson. *Security engineering - a guide to building dependable distributed systems* (2. ed.). Wiley 2008, ISBN 978-0-470-06852-6
- Chris Salter, O. Sami Saydjari, Bruce Schneier & Jim Wallner (1998): Toward a secure system engineering methodology. In: Proceedings of the New Security Paradigms Workshop, Charlottesville, VA, pp. 2–10.

- Lawrence A. Gordon & Martin P. Loeb (2002): The economics of information security investment. ACM Transactions on Information and System Security 5(4), pp. 438–457.
- Shawn A. Butler (2002): Security attribute evaluation method: a cost-benefit approach. In Will Tracz, Michal Young & Jeff Magee, editors: Proceedings of the 24th International Conference on Software Engineering, ICSE 2002, 19-25 May 2002, Orlando, Florida, USA, ACM, pp. 232–240, doi:10.1145/581339.581370. Available at http://doi.acm.org/10.1145/581339.581370.
- Maximiliano Cristiá: Seguridad Informática, apunte de clase. Disponible en http://www.fceia.unr.edu.ar/~mcristia/apunte-si.pdf

2. Arquitectura y diseño de software para Seguridad Informática

Objetivos

Se espera que el docente explique claramente las dificultades que involucra el desarrollo de aplicaciones seguras (que además deben cumplir con su especificación funcional). El módulo debe poner énfasis en los principios de seguridad mostrando ejemplos concretos donde cada principio se cumple y no se cumple (y en ese caso cuáles son las consecuencias).

Contenidos

Estructura de un sistema de cómputo (hardware, sistema operativo, software de base, comunicaciones, aplicaciones). Principio de seguridad desde el inicio. Principio de mediación total (*complete mediation*, B. Lampson). Principio de mínimo privilegio (*least privilege*). Principio de valores seguros por defecto (*failsafe defaults*). Principio de separación de privilegios. Principio de defensa en profundidad. Principio de economía de mecanismo. Principio del diseño abierto. Principio de mecanismo común mínimo (*least common mechanism*). Principio de aceptación psicológica (de los mecanismos de seguridad). Ejemplos donde cada principio no se cumple y donde se cumple; mostrar cómo los componentes o funciones del sistema verifican o no los principios estudiados. La seguridad es un proceso, no un producto. Monitor de referencias (*reference monitor*) y núcleo de seguridad. Minimización del núcleo de seguridad. Base de cómputo confiable (TCB, *trusted computing base*). Aplicaciones confiables. Camino confiable (*trusted path*). Aislamiento: máquinas virtuales y *sandboxes*. Seguridad en redes de computadoras (sistemas distribuidos): cliente, servidor, comunicaciones, sujetos, objetos, control de acceso, perímetro de la red, camino protegido, servidores de autenticación y autorización, núcleo de seguridad distribuido.

Bibliografía de referencia

- Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley 2002, ISBN 0-201-44099-7
- Morrie Gasser. *Building a Secure Computer System*. Van Nostrand Reinhold Co., 1988, ISBN 0-442-23022-2. Disponible en:

  https://www.researchgate.net/publication/242363259_Building_a_Secure_Computer_System

- Ross Anderson. *Security engineering - a guide to building dependable distributed systems* (2. ed.). Wiley 2008, ISBN 978-0-470-06852-6
- Iván Arce y otros. Avoiding the top 10 software security design flaws. IEEE Cybersecurity, 2015.

  https://cybersecurity.ieee.org/blog/2015/11/13/avoiding-the-top-10-security-flaws/

- Dougherty, Chad; Sayre, Kirk; Seacord, Robert; Svoboda, David; & Togashi, Kazuya. Secure Design Patterns. CMU/SEI-2009-TR-010. Software Engineering Institute, Carnegie Mellon University. 2009.

  http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9115

3. Seguridad en sistemas operativos y redes

Objetivos

Se espera que se presenten con cierta profundidad los mecanismos de control de acceso presentes en sistemas operativos de uso masivo. Luego, se deberían mostrar otros modelos de control de acceso y su implementación en sistemas operativos menos difundidos. No se espera que los temas sobre seguridad en redes sean tratados en profundidad.

Contenidos

Control de acceso en sistemas operativos. Control de acceso discrecional (DAC, *discretionary access control*). Matriz de control de acceso. Listas de control de acceso. Aplicación a UNIX/Linux/Windows. Llamadas al sistema para control de acceso. Ataque a la confidencialidad con caballo de Troya en sistemas con DAC. Control de acceso obligatorio (*mandatory access control*, MAC). Seguridad multi-nivel (*multi-level security*, MLS). Ataque de caballo de Troya en sistemas con MAC. Canales encubiertos. Mención de modelos MLS. Hardening de sistemas operativos. Configuración de servicios de red (hardening, aplicación de mínimo privilegio, separación de privilegios, etc.). Distribuciones, sistemas operativos, y mecanismos de sistemas operativos orientados a seguridad (Security-Enhanced Linux, Qubes OS, Whonix, GEMSOS (comercial), Docker, Linux namespaces, AppArmor, CGroups, seccomp). *Switches, firewalls y proxies*. Zona demilitarizada (DMZ). Protocolos de comunicación seguros (sin detalles de criptografía). Redes privadas virtuales (VPN). Disponibilidad: redundancia, balanceo de carga.

Bibliografía de referencia

- Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley 2002, ISBN 0-201-44099-7
- Morrie Gasser. *Building a Secure Computer System*. Van Nostrand Reinhold Co., 1988, ISBN 0-442-23022-2. Disponible en:

  https://www.researchgate.net/publication/242363259_Building_a_Secure_Computer_System

- Ross Anderson. *Security engineering - a guide to building dependable distributed systems* (2. ed.). Wiley 2008, ISBN 978-0-470-06852-6
- Maximiliano Cristiá: Seguridad Informática, apunte de clase. Disponible en http://www.fceia.unr.edu.ar/~mcristia/apunte-si.pdf
- *Hardening* de sistemas operativos: https://www.cisecurity.org/cis-benchmarks/

4. Desborde de arreglos: ataques y contramedidas

Objetivos

Este módulo está orientado a mostrar: a) el ataque por desborde de arreglo asumiendo que no hay ninguna protección activa; y b) contramedidas. Se proponen como tópicos avanzados ataques cuando hay una o más contramedidas en funcionamiento.

Contenidos

Estructura de un proceso: segmentos de texto, datos y pila. Funcionamiento de la pila. Estructura de un marco de pila. Ataque por desborde de arreglo. Programación del *shellcode* o *payload*. Ejecución del *shellcode*. Herramientas para programación y ejecución de *shellcode*. Programas que usualmente son objeto de ataques por desborde de arreglo (servicios de red, programas SUID, librerías de renderización de imágenes, etc.). Contramedidas: protecciones a nivel de sistema operativo; lenguajes de programación tipados y no tipados; protecciones a nivel de compilador; protecciones a nivel de librerías; analizadores estáticos de código (alcance, falsos positivos y negativos); analizadores en tiempo de ejecución (alcance, falsos positivos y negativos); reglas de la programación segura; revisiones de código (sin detalles, se trata en Módulo 6). Ejercicios. Desborde de arreglo en perspectiva: el ataque por desborde de arreglo en relación al ataque a la confidencialidad con caballo de Troya. Tópicos avanzados (descripción superficial orientada a mostrar las falencias de las contramedidas; estos tópicos se pueden profundizar en Módulo 7): *buffer underflow* de la pila; desbordamiento del montículo (*heap*); corrupción de cadenas de formato (*format strings*); programación basada en retornos (*return-oriented programming*, ROP); programación basada en retornos sin retornos.

Bibliografía de referencia

- Aleph One: Smashing The Stack For Fun And Profit. Available at http://insecure.org/stf/smashstack.html.
- Maximiliano Cristiá: Seguridad Informática, apunte de clase. Disponible en http://www.fceia.unr.edu.ar/~mcristia/apunte-si.pdf
- Fundación Sadosky. Guía de auto-estudio para la escritura de exploits. Disponible en https://fundacion-sadosky.github.io/guia-escritura-exploits/.
- Matt Bishop: A Clinic for "Secure" Programming. IEEE Secur. Priv. 8(2): 54-56 (2010). Ver también: http://nob.cs.ucdavis.edu/bishop/papers/2006-cisse-2/clinic.pdf
- Matt Bishop, Deborah A. Frincke: Teaching Robust Programming. IEEE Secur. Priv. 2(2): 54-57 (2004).
- Matt Bishop, Chip Elliott: Robust Programming by Example. World Conference on Information Security Education 2011: 140-147
- Matt Bishop: Some "Secure Programming" Exercises for an Introductory Programming Class. World Conference on Information Security Education 2009: 226-232. Disponible en https://hal.inria.fr/hal-01463642/document
- M. Bishop, Robust Programming, handout for ECS 153, Computer Security (Jan. 2006); available at

  http://nob.cs.ucdavis.edu/classes/ecs153-2006-01/handouts/robust.pdf

- Mark G. Graff, Kenneth R. van Wyk. *Secure coding - principles and practices: designing and implementing secure applications*. O'Reilly 2003, ISBN 978-0-596-00242-8, pp. 1-203
- John Viega, Matt Messier. *Secure programming cookbook for C and C++ - recipes for cryptography, authentication, networking, input validation and more*. O'Reilly 2003, ISBN 978-0-596-00394-4, pp. I-XXV, 1-762

5. Vulnerabilidades y programación segura en aplicaciones Web

Objetivos

Este módulo se enfoca en mostrar: a) el ataque por inyección de sentencias SQL cuando no hay protecciones activas; y b) contramedidas. Se proponen como tópicos avanzados ataques cuando hay una o más contramedidas en funcionamiento y otros ataques en aplicaciones Web.

Contenidos

Estructura básica de una aplicación Web que accede a una base de datos relacional. Definición del ataque por inyección de sentencias SQL (ISQL). Daños que pueden causar estos ataques. Detalles técnicos del ataque. Ejemplos. Tecnologías Web vulnerables a ISQL. Herramientas para detección de aplicaciones Web vulnerables a ISQL. Contramedidas: protecciones a nivel de sistema operativo; lenguajes de programación tipados y no tipados; protecciones a nivel de compilador; protecciones a nivel de librerías; analizadores estáticos de código (alcance, falsos positivos y negativos); analizadores en tiempo de ejecución (alcance, falsos positivos y negativos); reglas de la programación segura para aplicaciones Web; revisiones de código (sin detalles, se trata en Módulo 6). Ejercicios. ISQL en perspectiva: el ataque por ISQL en relación al ataque a la confidencialidad con caballo de Troya. Tópicos avanzados (descripción superficial orientada a mostrar otras problemáticas del desarrollo Web seguro; estos tópicos se pueden profundizar en Módulo 7): Software Assurance Maturity Model; *firewalls* para aplicaciones Web (e.g. ModSecurity); OWASP Application Security Verification Standard; programación Web segura en Java; programación Web segura en PHP; programación Web segura en Javascript; Cross Site Scripting (XSS); XPATH Injection; CSV Injection; otros ataques específicos de aplicaciones Web indicados por OWASP.

Bibliografía de referencia

- Recursos disponibles en OWASP (https://owasp.org/)
- Michael Howard, David LeBlanc and John Viega. *24 Deadly Sins of Software Security*. McGraw-Hill 2010, 978-0-07-162676-7. Disponible en:

  https://vdoc.pub/download/24-deadly-sins-of-software-security-programming-flaws-and-how-to-fix-them-3ges8rqic82g

- Oracle. Secure Coding Guidelines for Java SE. 2022. Disponible en: https://www.oracle.com/java/technologies/javase/seccodeguide.html
- Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland, David Svoboda. *The CERT Oracle Secure Coding Standard for Java.* Addison-Wesley Professional 2011, ISBN: 978-0-321-80395-5.

- Chris Snyder, Thomas Myer and Michael Southwell. *Pro PHP Security*. Apress 2010, 978-1-4302-3318-3.
- Chris Shiflett. *Essential PHP Security*. O'Reilly 2005, ISBN: 0-596-00656-X.

6. Revisiones de código orientadas a seguridad a nivel de aplicación

Objetivos

Este módulo está orientado a profundizar en las revisiones de código como herramienta para detectar y depurar vulnerabilidades presentes en programas. El instructor deberá mostrar ejemplos y proponer ejercicios donde las revisiones de código ayuden a detectar y eliminar vulnerabilidades. En las revisiones se podrá usar una combinación de herramientas y listas de control (*checklists*) con la simple lectura del código, poniendo énfasis en esta última. El temario que se propone es abierto de forma tal que el instructor pueda fijar los contenidos específicos. No se espera cubrir todas las vulnerabilidades existentes sino por el contrario se recomienda hacer foco en unas pocas (3 o 4) que incluyan desborde de arreglos e inyección de sentencias SQL.

Contenidos

Listas de control y herramientas para la asistencia en revisiones de código orientadas a seguridad en aplicaciones de sistemas (*system programming*); por ejemplo en lenguaje C sobre Linux. Ejemplos y ejercicios de revisiones de código orientadas a seguridad en aplicaciones de sistemas. Listas de control y herramientas para la asistencia en revisiones de código orientadas a seguridad en aplicaciones Web; por ejemplo en lenguaje Java o PHP. Ejemplos y ejercicios de revisiones de código orientadas a seguridad en aplicaciones Web.

Bibliografía de referencia

Por la especificidad del temario se espera que los docentes a cargo del módulo dispongan de los recursos bibliográficos más adecuados. De todas formas sugerimos la siguiente bibliografía.

- Michael Howard, David LeBlanc and John Viega. *24 Deadly Sins of Software Security*. McGraw-Hill 2010, 978-0-07-162676-7. Disponible en:

  https://vdoc.pub/download/24-deadly-sins-of-software-security-programming-flaws-and-how-to-fix-them-3ges8rqic82g

- Oracle. Secure Coding Guidelines for Java SE. 2022. Disponible en: https://www.oracle.com/java/technologies/javase/seccodeguide.html
- Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland, David Svoboda. *The CERT Oracle Secure Coding Standard for Java*. Addison-Wesley Professional 2011, ISBN: 978-0-321-80395-5.
- Chris Snyder, Thomas Myer and Michael Southwell. *Pro PHP Security*. Apress 2010, 978-1-4302-3318-3.
- Chris Shiflett. *Essential PHP Security*. O'Reilly 2005, ISBN: 0-596-00656-X.

7. Tópicos complementarios o avanzados

Objetivos

Este módulo tiene como finalidad proveer de un menú de temas para que cada estudiante elija uno con el fin de profundizarlo. El estudiante deberá seleccionar uno o dos de los recursos bibliográficos que se proponen para cada tema. Si el estudiante lo requiere el coordinador puede asistirlo en la elección del tema y de los recursos bibliográficos. La idea es que el estudiante pueda estudiar los recursos seleccionados en las 24 horas previstas para el módulo. Los temas que se proponen constituyen posibles continuaciones de los temas desarrollados en los módulos anteriores.

Contenidos

- **Módulo 2**. Aplicación de patrones de diseño para estructurar código de seguridad a nivel de aplicación de forma tal de mantener separadas la implementación de los requisitos funcionales de la implementación de la política de seguridad. Posibles ejemplos: Decorator (Wrapper) para control de acceso; Chain of Responsability para generación de bitácoras de auditoría; Secure Factory para crear objetos en base a las credenciales del usuario.

  Bibliografía sugerida

  i. Dougherty, Chad; Sayre, Kirk; Seacord, Robert; Svoboda, David; and Togashi, Kazuya. Secure Design Patterns. CMU/SEI-2009-TR-010. Software Engineering Institute, Carnegie Mellon University. 2009. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9115
  ii. Leonard J. LaPadula: A Rule-Set Approach to Formal Modeling of a Trusted Computer System. Comput. Syst. 7(1): 113-167 (1994). Disponible en: https://www.usenix.org/legacy/publications/compsystems/1994/win_lapadula.pdf

- **Módulo 3**. Modelos de confidencialidad: modelo de Bell-LaPadula; flujo de información seguro; no-interferencia; multi-ejecución segura. Flujo de información seguro basado en lenguajes; sistemas de tipos para flujo de información; aplicaciones en programación funcional. *Proof-carrying code*.

  Bibliografía sugerida

  i. Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley 2002, ISBN 0-201-44099-7. Capítulos: 4, 5, 8.
  ii. Morrie Gasser. *Building a Secure Computer System*. Van Nostrand Reinhold Co., 1988, ISBN 0-442-23022-2. Disponible en:

  https://www.researchgate.net/publication/242363259_Building_a_Secure_Computer_System

  Capítulos: 6, 7.

  iii. Maximiliano Cristiá: Seguridad Informática, apunte de clase. Disponible en http://www.fceia.unr.edu.ar/~mcristia/apunte-si.pdf. Unidad II.
  iv. D. Elliot Bell & Leonard LaPadula (1973): Secure computer systems: Mathematical foundations. MTR 2547, The MITRE Corporation.

v. D. Elliot Bell & Leonard LaPadula (1973): Secure computer systems: Mathematical model. ESD-TR 73-278, The MITRE Corporation.

vi. Maximiliano Cristiá & Gianfranco Rossi (2021): Automated Proof of Bell-LaPadula Security Properties. J. Autom. Reason. 65(4), pp. 463–478, doi:10.1007/s10817-020-09577-6. Available at:

https://doi.org/10.1007/s10817-020-09577-6.

vii. Dorothy E. Denning: A Lattice Model of Secure Information Flow. Commun. ACM 19(5): 236-243 (1976). Disponible en:

https://courses.cs.washington.edu/courses/cse590s/02sp/secure-information-flow.pdf

viii. Joseph A. Goguen & José Meseguer (1982): Security Policies and Security Models. In: 1982 IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 26-28, 1982, IEEE Computer Society, pp. 11–20, doi:10.1109/SP.1982.10014. Available at:

https://doi.org/10.1109/SP.1982.10014.

ix. Daryl McCullough (1987): Specifications for Multi-Level Security and a Hook-Up Property. In: Proceedings of the 1987 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 27-29, 1987, IEEE Computer Society, pp. 161–166, doi:10.1109/SP.1987.10009. Available at: https://doi.org/10.1109/SP.1987.10009.

x. Daryl McCullough (1988): Noninterference and the composability of security properties. In: IEEE Symposium on Security and Privacy, Oakland, CA, pp. 177–186.

xi. Ian Sutherland & et. al (1991): Romulus: A Computer Security Properties Modeling Environment. The Theory of Security. Technical Report RL-TR-91-36 Vol IIa, Rome Laboratory. Disponible en:

https://www.researchgate.net/publication/235029068_Romulus_A_Computer_Security_Properties_Modeling_Environment_Volume_2B_MATHESIS

xii. Michael R. Clarkson and Fred B. Schneider (2010): Hyperproperties. Journal of Computer Security 18(6), pp. 1157–1210, doi:10.3233/JCS-2009-0393. Available at https://doi.org/10.3233/JCS-2009-0393.

xiii. Andrei Sabelfeld, Andrew C. Myers: Language-based information-flow security. IEEE J. Sel. Areas Commun. 21(1): 5-19 (2003)

xiv. Dennis M. Volpano, Cynthia E. Irvine and Geoffrey Smith (1996): A Sound Type System for Secure Flow Analysis. Journal of Computer Security 4(2/3), pp. 167–188, doi:10.3233/JCS-1996-42-304. Available at: https://doi.org/10.3233/JCS-1996-42-304.

xv. Maximiliano Cristiá & Pablo Mata (2009): Runtime Enforcement of Noninterference by Duplicating Processes and their Memories. In: Workshop de Seguridad Informática WSEGI 2009, SADIO, Mar del Plata, Argentina. Available at:

http://www.fceia.unr.edu.ar/%7Emcristia/publicaciones/flowx-abs-model.pdf

xvi. Dominique Devriese & Frank Piessens (2010): Noninterference through Secure Multi-execution. In: 31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA, IEEE Computer Society, pp. 109–124, doi:10.1109/SP.2010.15. Available at:

https://pdfs.semanticscholar.org/4e73/34db18da606f0ddb85caab476a026337aa1e.pdf.

xvii. Mauro Jaskelioff, Alejandro Russo: Secure Multi-execution in Haskell. Ershov Memorial Conference 2011: 170-178

xviii. George C. Necula (1997): Proof-Carrying Code. In Peter Lee, Fritz Henglein & Neil D. Jones, editors: Conference Record of POP'97: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Papers Presented at the Symposium, Paris, France, 15-17 January 1997, ACM Press, pp. 106–119, doi:10.1145/263699.263712. Available at https://www.utdallas.edu/~hamlen/Papers/necula97proofcarrying.pdf.

xix. Dexter Kozen: Language-Based Security. MFCS 1999: 284-298. Disponible en: http://www.cs.cornell.edu/~kozen/Papers/lbs.pdf

- **Módulo 4**. *Buffer underflow* (*underrun* o *underwrite*) de la pila. Desbordamiento del montículo (*heap*). Corrupción de cadenas de formato (*format strings*). Desbordamiento de enteros. Programación basada en retornos (*return-oriented programming*, ROP). Programación basada en retornos sin retornos.

  Bibliografía sugerida

i. Michael Howard, David LeBlanc and John Viega. *24 Deadly Sins of Software Security*. McGraw-Hill 2010, 978-0-07-162676-7. Disponible en:

https://vdoc.pub/download/24-deadly-sins-of-software-security-programming-flaws-and-how-to-fix-them-3ges8rqic82g

ii. CWE-124: Buffer Underwrite ('Buffer Underflow'). Common Weakness Enumeration. MITRE. 2006-07-19 Disponible en: https://cwe.mitre.org/data/definitions/124.html

iii. Michel "MaXX" Kaempf. Vudo - An object superstitiously believed to embody magical powers. Phrack Volume 0x0b, Issue 0x39, Phile #0x08 of 0x12 (2001). Disponible en http://phrack.org/issues/57/8.html#article.

iv. Phantasmal Phantasmagoria. Exploiting the Wilderness. Bugtraq mailing list archives (February 2004). Disponible en:

https://seclists.org/vuln-dev/2004/Feb/25

v. Phantasmal Phantasmagoria. The Malloc Maleficarum. Bugtraq mailing list archives (October 2005). Disponible en:

https://seclists.org/bugtraq/2005/Oct/118

vi. blackngel. MALLOC DES-MALEFICARUM. Phrack Volume 0x0d, Issue 0x42, Phile #0x0A of 0x11 (2009). Disponible en http://phrack.org/issues/66/10.html

vii. scut. Exploiting Format String Vulnerabilities. 2001. Disponible en: https://julianor.tripod.com/bc/formatstring-1.2.pdf
viii. CWE-134: Uncontrolled Format String. Common Weakness Enumeration. MITRE. 2010-12-13. Retrieved 2011-03-05. Disponible en: http://cwe.mitre.org/data/definitions/134.html
ix. gera and riq. Advances in format string exploitation. Phrack 0x0b, Issue 0x3b, Phile #0x07 of 0x12 (2002). Disponible en: http://phrack.org/issues/59/7.html
x. blexim. Basic Integer Overflows. Phrack Volume 0x0b, Issue 0x3c, Phile #0x0a of 0x10 (2002). Disponible en: http://phrack.org/issues/60/10.html#article
xi. Saif El-Sherei. Integer overflow/underflow exploitation tutorial. Disponible en: https://www.exploit-db.com/docs/english/28477-linux-integer-overflow-and-underflow.pdf
xii. H. Shacham. The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In S. De Capitani di Vimercati and P. Syverson, editors, Proceedings of CCS 2007, pages 552–61. ACM Press, Oct. 2007.
xiii. E. Buchanan, R. Roemer, H. Shacham, and S. Savage. When good instructions go bad: Generalizing return-oriented programming to RISC. In P. Syverson and S. Jha, editors, Proceedings of CCS 2008, pages 27–38. ACM Press, Oct. 2008.
xiv. R. Hund, T. Holz, and F. Freiling. Return-oriented rootkits: Bypassing kernel code integrity protection mechanisms. In F. Monrose, editor, Proceedings of USENIX Security 2009, pages 383–98. USENIX, Aug. 2009.
xv. R. Roemer. Finding the bad in good code: Automated return-oriented programming exploit discovery. Master's thesis, UC San Diego, Mar. 2009. Online: https://cseweb.ucsd.edu/~rroemer/doc/thesis.pdf.
xvi. Stephen Checkoway, Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi, Hovav Shacham, Marcel Winandy: Return-oriented programming without returns. CCS 2010: 559-572

- **Módulo 5**. Software Assurance Maturity Model. *Firewalls* para aplicaciones Web (e.g. ModSecurity). OWASP Application Security Verification Standard. Programación Web segura en Java. Programación Web segura en PHP. Cross Site Scripting (XSS). XPATH Injection. CSV Injection. Otros ataques específicos de aplicaciones Web indicados por OWASP.

  Bibliografía sugerida

i. Michael Howard, David LeBlanc and John Viega. *24 Deadly Sins of Software Security*. McGraw-Hill 2010, 978-0-07-162676-7. Disponible en:

https://vdoc.pub/download/24-deadly-sins-of-software-security-programming-flaws-and-how-to-fix-them-3ges8rqic82g

ii. OWASP. Software Assurance Maturity Model: A guide to building security into software development – VERSION 1.5. Disponible en. https://owasp.org/www-pdf-archive/SAMM_Core_V1-5_FINAL.pdf

iii. Maximillan Dermann; Mirko Dziadzka; Boris Hemkemeier; Alexander Meisel; Matthias Rohr; Thomas Schreiber. OWASP Best Practices: Use of Web Application Firewalls ver. 1.0.5. OWASP 2008. OWASP. Disponible en:

https://wiki.owasp.org/index.php/Category:OWASP_Best_Practices:_Use_of_Web_Application_Firewalls/Version_1.0.5

iv. Paul E. Black, Elizabeth Fong, Vadim Okun and Romain Gaucher. Software Assurance Tools: Web Application Security Scanner Functional Specification Version 1.0. SAMATE NIST 2008. Disponible en:

https://www.nist.gov/system/files/documents/2021/03/23/webapp_scanner_spec_sp500-269.pdf

v. Jason Pubal. Web Application Firewalls. SANS Institute 2015. SANS Institute InfoSec Reading Room. Disponible en:

https://sansorg.egnyte.com/dl/70ml4jyHfO

vi. OWASP. Application Security Verification Standard 4.0.3. 2021. Disponible en: https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf

vii. Oracle. Secure Coding Guidelines for Java SE. 2022. Disponible en: https://www.oracle.com/java/technologies/javase/seccodeguide.html

viii. Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland, David Svoboda. *The CERT Oracle Secure Coding Standard for Java*. Addison-Wesley Professional 2011. ISBN: 978-0-321-80395-5.

ix. Chris Snyder, Thomas Myer and Michael Southwell. *Pro PHP Security*. Apress 2010, 978-1-4302-3318-3.

x. Chris Shiflett. *Essential PHP Security*. O'Reilly 2005, ISBN: 0-596-00656-X.

xi. Paragon Initiative Enterprises Staff. The 2018 Guide to Building Secure PHP Software. Paragon Initiative Enterprises 2018. Disponible en: https://paragonie.com/blog/2017/12/2018-guide-building-secure-php-software

xii. KirstenS. Cross Site Scripting (XSS). OWASP. Disponible en: https://owasp.org/www-community/attacks/xss/

xiii. Timo Goosen and Albinowax. CSV Injection. OWASP. Disponible en. https://owasp.org/www-community/attacks/CSV_Injection

xiv. XPATH Injection. OWASP. Disponible en:

https://owasp.org/www-community/attacks/XPATH_Injection

- **Módulo 6**. Profundizar los temas vistos en el módulo cubriendo vulnerabilidades no vistas en clase de manera tal de poder reconocerlas leyendo código, eventualmente con la ayuda de herramientas.

Bibliografía sugerida

i. Michael Howard, David LeBlanc and John Viega. 24 Deadly Sins of Software Security. McGraw-Hill 2010, 978-0-07-162676-7. Disponible en:
ii. https://vdoc.pub/download/24-deadly-sins-of-software-security-programming-flaws-and-how-to-fix-them-3ges8rqic82g
iii. Oracle. Secure Coding Guidelines for Java SE. 2022. Disponible en: https://www.oracle.com/java/technologies/javase/seccodeguide.html
iv. Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland, David Svoboda. The CERT Oracle Secure Coding Standard for Java. Addison-Wesley Professional 2011, ISBN: 978-0-321-80395-5.
v. Chris Snyder, Thomas Myer and Michael Southwell. Pro PHP Security. Apress 2010, 978-1-4302-3318-3.
vi. Chris Shiflett. Essential PHP Security. O'Reilly 2005, ISBN: 0-596-00656-X.

8. Taller de Didáctica del Desarrollo Seguro

¿Cuáles son las preguntas que un docente tiene que realizarse al momento de armar una asignatura? Diagnóstico. Infraestructura de la institución. Características del alumnado. Definición del formato curricular en función de los objetivos (materia, taller, ateneo). Profundidad de contenidos. Planificación. Evaluación.

Contenidos teóricos y prácticos. Necesidad de trabajar estos últimos sobre la computadora. Configuración del laboratorio.