Ministerio de Cultura y Educación **(Programa del año 2024)**
Universidad Nacional de San Luis
Facultad de Ciencias Físico Matemáticas y Naturales
Departamento: Informatica
Area: Area IV: Pr. y Met. de Des. del Soft.

I - Oferta Académica

Materia	Carrera	Plan	Año	Período
() OP.SEGURIDAD DE SISTEMAS DE SOFTWARE	ING. EN COMPUT.	28/12	2024	2° cuatrimestre
() OP.SEGURIDAD DE SISTEMAS DE	ING. INFORM.	026/12-08/15	2024	2° cuatrimestre
() OP.SEGURIDAD DE SISTEMAS DE SOFTWARE	LIC.CS.COMP.	32/12	2024	2° cuatrimestre

II - Equipo Docente

Docente	Función	Cargo	Dedicación
BERON, MARIO MARCELO	Prof. Responsable	P.Adj Exc	40 Hs

III - Características del Curso

Credito Horario Semanal				
Teórico/Práctico	**Teóricas**	**Prácticas de Aula**	**Práct. de lab/ camp/ Resid/ PIP, etc.**	**Total**
Hs	2 Hs	1 Hs	2 Hs	5 Hs

Tipificación	Periodo
B - Teoria con prácticas de aula y laboratorio	2º Cuatrimestre

Duración			
Desde	**Hasta**	**Cantidad de Semanas**	**Cantidad de Horas**
05/08/2024	15/11/2024	15	75

IV - Fundamentación

El amplio crecimiento que las tecnologías de la información y de la comunicación han experimentado en este último tiempo ha provocado una utilización masiva de los sistemas de software en todos los ámbitos de la sociedad. En la actualidad la gran mayoría de las actividades se realizan a través de la utilización de medios informáticos lo cual permite realizar tareas que tiempo atrás eran impensadas. Los usuarios realizan compras por Internet, inscriben a sus hijos en las instituciones educativas, solicitan turnos a médicos, pagan sus cuentas desde sus hogares, entre otras tantas alternativas. Muchas labores sociales han cambiado rotundamente debido a los avances en el desarrollo del hardware y del software. Es posible observar que existen trabajos en donde las tareas se realizan desde el hogar, comercios donde no son necesarios funcionarios que atiendan a los clientes, etc. Las organizaciones realizan tareas similares pero además construyen sistemas de software que hacen posible que los usuarios realicen sus labores a distancia. Como es posible percibir, el software ocupa un lugar central en la sociedad moderna. Los sistemas se hacen cada vez más complejos y grandes lo cual implica la participación de diferentes equipos cada uno de ellos compuesto por muchas personas. Asociado con los beneficios que la tecnología ha producido a la sociedad se encuentran riesgos, algunos de los cuales son muy serios, que hacen los usuarios queden expuestos a acciones no deseadas. Claramente, la tecnología y en particular el software llevan adelante tareas que son críticas para todos los miembros de la sociedad. Basta con pensar los problemas que pueden causar una transacción bancaria errónea,

un pago mal realizado o una ficha médica mal procesada, entre otras tantas posibilidades. Dichos problemas pueden ser causados de manera no intencional, por fallas en el proceso de construcción de software. No obstante, el desperfecto también se puede producir de manera intencional. Sea intencional o no, se puede decir que cuando se produce un desperfecto o fuga de información se está en presencia de una violación a la seguridad. En el caso de que ocurra una falla de software la misma se puede deber a errores en la construcción, en el mantenimiento o en la migración o evolución del software. Claramente es deseable que el software sea confiable, seguro, fácil de mantener y modificar. No obstante, esas metas son muy difíciles de alcanzar. De hecho existen líneas de investigación, tanto en el contexto académico como en el empresarial, que tienen como principal objetivo desarrollar métodos y estrategias para evitar los inconvenientes antes mencionados. Los accesos no deseados también están relacionados con la falta de calidad del software como así también a fallas en el proceso de construcción del mismo. Por las razones antes mencionadas, la seguridad de los sistemas de software han adquirido una importancia central en la actualidad y a posibilitado el surgimiento de una amplia variedad de investigaciones y desarrollos tecnológicos en el contexto de la seguridad de software. Lo antes mencionado hace que la seguridad informática sea un aspecto clave que los profesionales de informática necesitan dominar y aplicar a los sistemas de software. En esta materia optativa se brindan los conceptos y técnicas esenciales para evitar, detectar y subsanar fallas en la seguridad de los sistemas de software de forma tal de nutrir a los estudiantes con las bases para la producción de software seguro.

V - Objetivos / Resultados de Aprendizaje

Al finalizar el curso los alumnos deberán:
1. Conocer las principales fallas en la codificación de los sistemas de software.
2. Conocer estrategias de ataque y defensa.
3. Operar herramientas para implementar estrategias de protección de software.

Durante el dictado de la asignatura se abordan los siguientes ejes transversales:
1. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información/informática.
2. Utilización de técnicas y herramientas de aplicación en de ingeniería en sistemas de información/informática.
3. Generación de desarrollos tecnológicos y/o innovaciones tecnológicas.
4. Fundamentos para el aprendizaje continuo.
5. Fundamentos para el desempeño en equipos de trabajo.
6. Fundamentos para una comunicación efectiva.
7. Fundamentos para una actuación profesional ética y responsable.

VI - Contenidos

Por cada unidad se deja disponible el material correspondiente a los contenidos de la unidad, las diapositivas de clase, el apunte teórico y su correspondiente trabajo práctico en el repositorio digital. Para una mejor organización, tanto de los estudiantes como del equipo docente, se presenta un cronograma con la descripción de las días de teoría y práctica, fechas de parciales y fechas de entrega de prácticos de laboratorio.

Unidad 1: Desarrollo de Software Seguro
Seguridad en los Sistemas de Software. Procesos de Desarrollo de Software Seguro. Estándares de Codificación Segura. Codificación Segura en C. Codificación Segura en Java. Patrones de Errores en la Codificación.

Unidad 2: Protección de Software
Concepción. Ataque y Defensa. Análisis de Programas. Ofuscación de Código. Aplicaciones de la Ofuscación de Código. Tipos de Ofuscación. Software a Prueba de Manipulaciones. Aplicaciones. Marca de Agua. Similaridad de Software. Análisis Forense de Software. Marca de Nacimiento. Técnicas de Protección por Hardware.

Unidad 3: Análisis de Programas
Análisis Estático. Análisis de Control de Flujo. Análisis de Flujo de Datos. Análisis de Dependencia de Datos. Análisis de Alias. Slicing. Interpretación Abstracta. Análisis Dinámico. Depuración. Profiling. Tracing. Emulación. Reconstitución de los Fuentes. Desensamble. Decompilación. Análisis Pragmático. Métricas de Estilo. Métricas de Complejidad de Software. Visualización de Software.

Unidad 4: Métodos de Ataque y Defensa
Estrategias de Ataque. Objetivo de Cracking. Motivación del Adversario. Por qué el atacante logra el objetivo? Cuál es la metodología del atacante? Qué herramientas usa el atacante? Qué técnicas usa el adversario? Estrategias de Defensa. Motivación. Primitivas de Protección: Cubrir. Duplicar. Dividir. Mezclar. Reordenar. Map. Indirecto. Mímica. Aviso.

Unidad 5: Aplicaciones
Extracción de la Información. Lexer. Parser. Árbol de Parse. Recorridos sobre el Árbol de Parse. Extracción de la Información con Gramáticas de Atributos, Listener y Visitors. Herramientas de Generación Automática de Lexers y Parsers. Operación de herramientas de generación automática de lexers, parsers y árbol de parse.

VII - Plan de Trabajos Prácticos

Metodología de la Enseñanza

Los trabajos prácticos correspondientes a las unidades del programa consisten en problemas que deben ser resueltos por los estudiantes guiados por los docentes. En cada trabajo práctico los estudiantes deben resolver ejercicios que están relacionados con la teoría que se ha dictado en ese momento. Los ejercicios están organizados por orden de complejidad comenzando por ejercicios más simples, luego los de complejidad media y por último los más complejos. En cada práctico, con el fin de realizar una evaluación formativa, se resuelven ejercicios (correspondientes a cada nivel de complejidad) en el pizarrón y/o en la computadora de forma tal de fomentar la comunicación, hacer explícita la relación de conceptos, y detectar los inconvenientes que los estudiantes puedan tener.
En caso de detectar un problema, el docente resuelve ejercicios del mismo tipo y propone otros nuevos con la finalidad de que el estudiante pueda asimilar los conceptos y sus relaciones. Además, con este tipo de actividad, se busca que el estudiante empiece a expresarse de manera escrita utilizando un vocabulario acorde a los contenidos vistos, a socializar y reflexionar sobre sus respuestas y a detectar los conceptos no comprendidos.
Las clases están pensadas para que el alumno participe activamente respondiendo preguntas que el docente realiza las cuales pueden ser de teóricas o prácticas y resolviendo ejercicios en el pizarrón y/o en la computadora. A medida que se van desarrollando las clases, el docente va realizando análisis del desempeño de los estudiantes de forma tal de poder establecer el nivel de grupo y proponer actividades que resuelvan las dificultades que se puedan presentar.
Todos los ejercicios pueden ser validados utilizando herramientas (IDEs, compiladores e intérpretes) que permiten la ejecución de los ejercicios (dado que en la materia se enseñan estrategias de chequeo de la seguridad de los sistemas como así también se implementan estrategias de protección y se construyen herramientas que integran los conceptos). Con el fin de promover una relación más estrecha con el mercado laboral y también con el de investigación, en la materia se utilizan herramientas de uso profesional (los cuales también son muy utilizados en el contexto académico y el empresarial).
Durante todo el desarrollo de la materia los docentes hacen énfasis en conceptos de seguridad de software remarcado como se concretan en los sistemas e informando que el proceso de capacitación es continuo dado que los ataques y vulnerabilidades de la seguridad evolucionan y cada vez son más sofisticados. Se refuerza esta idea basándose en que las tecnologías y estrategias de ataque y defensa evolucionan constantemente lo que conduce a que aparezcan nuevos enfoque de la seguridad y por consiguiente se deban estudiar para evitar efectos no deseados por las organizaciones/usuarios de los sistemas de software.
También se promueve el trabajo en equipo, el desarrollo de soluciones innovadoras de los prácticos y el uso de vocabulario técnico, preciso y claro cuando los estudiantes explican los prácticos en clase.
La materia consta de un proyecto final integrador en el cual el estudiante integra los conocimientos adquiridos en los prácticos de aula y laboratorios. En este proyecto también se le solicita al estudiante que use técnicas de desarrollo de sistemas y de resolución creativa de problemas para promover la innovación. Una vez finalizado la tarea los estudiantes realizan una presentación la cual es evaluada por los docentes. En la evaluación se tiene en cuenta el vocabulario utilizado, la forma de comunicación, la calidad y pertinencia de la solución y las tareas realizadas por cada miembro del equipo.

Unidad 1: Desarrollo de Software Seguro
Analizar las metodologías de desarrollo de software seguro utilizadas por las grandes empresas de desarrollo de software y describir las actividades que se incorporan para considerar la característica de seguridad de software.

Objetivo: En este práctico se pretende que el estudiante:

- Aprenda las prácticas realizadas por las empresas para desarrollar software seguro.

Metodología:

- Se construyen grupos de dos o tres alumnos.
- Se distribuyen documentos donde se describen metodologías de desarrollo de software seguro.
- Se pide que realicen una presentación en donde se describan las procesos utilizados para el desarrollo de software seguro. La presentación se expone en clase.
- Luego de realizadas todas las presentaciones se lleva a cabo una discusión respecto de las ventajas y desventajas de cada metodología estudiada.
- La discusiones son seguidas por el docente el cual interviene cuando lo considere necesario.

El práctico incluye ejercicios donde se pretende que el estudiante:

- Analice las prácticas que se llevan a cabo para el desarrollo de software seguro.
- Proponga ejemplos de casos de características que tiene que tener el software seguro como así también de violaciones a la seguridad.

Unidad 2: Protección de Software

Implementación de estrategias de protección de software: Ofuscación, Marca de Agua, Marca de Nacimiento, Diversidad, entre otras. Diferentes variantes de cada una de las estrategias mencionadas con anterioridad.

Objetivo: En este práctico se pretende que el estudiante:

- Aprenda e implemente y aplique diferentes estrategias para proteger software.

Metodología:

- Se explican diferentes estrategias para proteger software
- Se construyen grupos de dos o tres alumnos.
- Se solicita la implementación de una estrategia y variantes de ella a cada grupo.
- Se pide que se realice una presentación en donde se explique las ventajas y desventajas de cada estrategia como así también las dificultades que surgieron durante su implementación.
- Luego de realizadas todas las presentaciones se lleva a cabo una discusión respecto de las ventajas y desventajas de cada estrategia de protección estudiada.
- La discusiones son seguidas por el docente el cual interviene cuando lo considere necesario.

El práctico incluye ejercicios donde se pretende que el estudiante:

- Conozca conceptualmente las estrategias de protección de software.
- Implemente prototipos de estrategias de protección.
- Analice las ventajas y desventajas de cada las estrategias de protección de software.

Unidad 3: Análisis de Programas

Operación de herramientas de Análisis de Programa. Análisis Estático. Análisis Dinámico. Componentes de necesarios para construir una herramienta de análisis de programa: Lexer, Parser, Árbol de Parse, Recorridos del Árbol de Parse para Extraer Información. Implementación de Técnicas Básicas de Análisis de Programas.

Objetivo: En este práctico se pretende que el estudiante:

- Aprenda a usar herramientas de análisis de programas.
- Entienda el funcionamiento de las diferentes componentes que forman parte de una estrategia de análisis estático y dinámico.
- Conozca los pasos necesarios para construir una herramienta de análisis de programas.
- Construya estrategias básicas que permitan realizar análisis estático de los sistemas de software.
- Construya estrategias básicas que permitan realizar análisis dinámico de los sistemas de software.

Metodología:

- Se utilizan diferentes herramientas para realizar análisis estático y dinámico.
- Se explica el proceso para realizar análisis estático y dinámico.
- Se construyen lexers y parsers para diferentes lenguajes de programación.
- Se construye el Árbol de Parse y se aplican recorridos para extraer información.
- Se construyen estrategias simple de ambos tipos de análisis (estático y dinámico)
- Se explican diferentes estrategias para proteger software
- Se construyen grupos de dos o tres alumnos.
- Se solicita la implementación de una estrategia y variantes de ella a cada grupo.
- Se pide que se realice una presentación en donde se explique las ventajas y desventajas de cada estrategia como así también las dificultades que surgieron durante su implementación.
- Luego de realizadas todas las presentaciones se lleva a cabo una discusión respecto de las ventajas y desventajas de cada estrategia de protección estudiada.
- La discusiones son seguidas por el docente el cual interviene cuando lo considere necesario.

El práctico incluye ejercicios donde se pretende que el estudiante:

- Conozca conceptualmente las estrategias de análisis estático y de análisis dinámico .
- Implemente prototipos de estrategias de cada tipo de análisis.
- Compare las distintas formas de llevar adelante un análisis estático.
- Compare las distintas formas de llevar adelante un análisis dinámico.
- Analice las ventajas y desventajas del análisis dinámico y del análisis estático.

Unidad 4: Métodos de Ataque y Defensa
Elaboración de estrategias de defensa ante situaciones de ataque.

Objetivos: En este práctico se pretende que el estudiante:
Aplique e implemente la estrategia de protección de software adecuada para un tipo de ataque específico.

Metodología:

- Se recapitulan las estrategias de ataque y defensa vistas en teoría.
- Se construyen grupos de dos o tres alumnos.
- Se analiza que tipo de información (estática o dinámica) se necesita para llevar adelante la implementación de las estrategias.
- Usando la información estática y dinámica se implementan estrategias de protección de software.

El práctico incluye ejercicios donde se pretende que el estudiante:

- Comprenda las diferentes formas en que se puede atacar un software.
- Implemente estrategias de protección de software.
- Analice las ventajas y desventajas de cada una de las estrategias implementadas.

Unidad 5: Aplicaciones
Construcción de aplicaciones que: Detecten problemas sencillos en el código fuente de un sistema de software; Protejan los activos del software.

Objetivo:
Integrar los conceptos de seguridad de los sistemas de software visto en la materia.

Metodología:
- Se construyen grupos de dos o tres alumnos.
- Se toma un sistema de complejidad media cuyo código sea de libre acceso.
- Se analiza el código del sistema.
- Se seleccionan primitivas de protección de software.
- Se construyen una herramienta con las primitivas de protección seleccionadas. La herramienta tiene que ser simple de usar de forma tal de que el ingeniero pueda usarla sin mayores inconvenientes.
- Se compara el sistema sin proteger y el sistema protegido y se sacan conclusiones respecto de la efectividad de la estrategia.

En esta unidad se realiza un laboratorio que consiste en la construcción de una herramienta básica de protección de software. En este laboratorio se utilizan implementaciones de primitivas realizadas en las unidades anteriores.

A continuación se explica como los ejes transversales empleados en la materia:
- Concepción, diseño y desarrollo de proyectos de de ingeniería en sistemas de información/informática: En los laboratorios se tienen que desarrollar proyectos de ingeniería en lo cuales se exigirá que los métodos, técnicas y herramientas vistos en la materia se apliquen correctamente.

- Utilización de técnicas y herramientas de aplicación en de ingeniería en sistemas de información/informática: Durante todo el desarrollo de la materia se fomentará el uso correcto del proceso de desarrollo de software seguro y la utilización de herramientas de análisis de código y programación empleadas en el mercado.

- Generación de desarrollos tecnológicos y/o innovaciones tecnológicas: Se promoverá, sobre todo en el desarrollo de los laboratorios, la construcción de soluciones innovadoras, en lo que respecta a construcción de software seguro, prestando especial atención aquellas en las que se requiere intervención del ingeniero.

- Fundamentos para el aprendizaje continuo: Las actividades tanto teóricas como prácticas se iniciarán con un repaso de contenidos previos pertinentes, con la participación de los estudiantes mediante consultas. Se realizará una corrección informada de las actividades solicitadas y de las evaluaciones. En todo el desarrollo de la materia se hará énfasis en la evolución de la tecnología y la necesidad de aprehender estrategias de estudio e investigación que faciliten el aprendizaje de las tecnologías orientadas desarrollar software seguro. Por otra parte se destacará la importancia de estar actualizado debido al incremento en la sofisticación de los ataques cibernéticos a los cuales el software está expuesto lo que demanda una vigilancia epistemológica constante.

- Fundamentos para el desempeño en equipos de trabajo: En todas las prácticas se organizarán grupos de estudiantes para fomentar el trabajo conjunto, tanto dentro como fuera del aula, verificando que cada integrante sea capaz de explicar parte de la tarea realizada.

- Fundamentos para una comunicación efectiva: En todas las actividades que implican la participación activa de los estudiantes, tanto escritas como orales, se prestará especial atención al empleo de terminología y notaciones propias de los desarrollos informáticos, así como a la claridad con que se expresen los conceptos de seguridad y programación utilizados. Además, se verificará que las intervenciones de los estudiantes sean pertinentes. En todos los casos, el equipo docente realizará las correcciones y/o sugerencias necesarias para una correcta comunicación, según el contexto.

- Fundamentos para una actuación profesional ética y responsable: Se compartirá un cronograma de evaluaciones y otras actividades. Para todas ellas, se establecerán plazos y formas de entrega. Se exigirán requisitos de asistencia a clases para regularizar y/o promocionar la materia. Se requerirá la presentación de los certificados correspondientes a quienes soliciten algún tipo de flexibilidad excepcional con causa que lo justifique. Se fomentará el reconocimiento de autoría cuando se usen algoritmos/estrategias de seguridad realizados por programadores que no forman parte del equipo de trabajo.

VIII - Regimen de Aprobación

Condiciones de Regularización:
i) Aprobar los prácticos de laboratorio. Se otorgarán dos recuperaciones por cada práctico de laboratorio a todos los alumnos.
ii) Aprobar un práctico teórico-práctico con una nota mayor o igual a 6 (seis). Se otorgarán dos recuperaciones para cada evaluación parcial a todos los alumnos.
Condiciones de Aprobación:
i) Por promoción:
a) El alumno debe:
a.1) Contar con la condición de regularización i).
a.2) Aprobar un práctico teórico-práctico con una nota mayor o igual a 7 (siete). Se otorgarán dos recuperaciones para cada práctico a todos los alumnos.
b) Aprobar una evaluación final integradora con una nota mayor o igual 7. Se otorgarán dos recuperaciones de esta evaluación a todos los alumnos.
c) Tener un porcentaje de asistencia del 80% a clases.
ii) Por examen final.
Alumnos Libres: No se aceptan alumnos con esta condición.

IX - Bibliografía Básica

[1] Application Security Verification Standard 3.0.1. OWASP. Open Web Application Security Project. 2016.
[2] Alfred V. Aho,Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman Compilers: Principles, Techniques, and Tools. Pearson
[3] Education. 2006.
[4] Long, Fred; Mohindra, Dhruv; Seacord,Robert C.; Sutherland, Dean F; Svoboda, David. Java Coding Guidelines. Addison-Wesley. ISBN- 3: 978-0-321-93315-7. 2014.
[5] Seacord, Robert C. Secure Coding in C and C++. Second Edition. Addison-Wesley. ISBN-13: 978-0-321-82213-0. 2013.
[6] Eilam, E. Reversing. Secrets of Reverse Engineering. Wiley. ISBN-10: 0764574817, ISBN-13: 978-0764574818. 2005.
[7] Pressman, R; Maxim, B. Software Engineering: A Practitioner’s Approach. McGraw-Hill. 8va Edición. ISBN-10:0078022126. ISBN-13: 978-0078022128. 2014.
[8] Chess, Brian; West,Jacob. Secure Programming with Static Analysis. ISBN: 0-321-42477-8. Addison-Wesley. 2007.
[9] Tomassetti, Federico. How to create pragmatic, lightweightlanguages. Leanpub. 2018.
[10] Howard, Michael; Lipner, Steve. THE SECURITY DEVELOPMENT LIFECYCLE. SDL: A Process for Developing Demonstrably More Secure Software. Microsoft Press. ISBN: 978-07356-2214-2. 2006.

X - Bibliografia Complementaria

XI - Resumen de Objetivos

1. Conocer las principales fallas en la codificación de los sistemas de software.
2. Conocer estrategias de ataque y defensa.
3. Operar herramientas para implementar estrategias de protección de software.

XII - Resumen del Programa

Unidad 1: Desarrollo de Software Seguro
Unidad 2: Protección de Software
Unidad 3: Análisis de Programas
Unidad 4: Métodos de Ataque y Defensa
Unidad 5: Aplicaciones

XIII - Imprevistos

Los imprevistos serán resueltos por la cátedra en la medida que aparezcan.
Correo de Contacto:
Mario Berón -- mberon@email.unsl.edu.ar

XIV - Otros

Las vías de comunicación con los estudiantes son las siguientes:

*Correo electrónico del responsable de la materia:mberon@email.unsl.edu.ar
*Oficina: 3 Bloque II - Primer Piso
*Teléfono: +54 (266) 4520300 - Interno: 2103