

ESCUELA DE INGENIERÍA EN COMPUTACIÓN  
Proyecto Integrador**Solicitud de Proyecto Integrador**


Sr. Coordinador/Encargado  
de Proyecto Integrador  
Prof. Ing. Ventre, Luis O.  
S / D

Córdoba, 13 de Junio 2024.-

Me dirijo a Ud. a fin de solicitar la aprobación del tema de  
PROYECTO INTEGRADOR que propongo a continuación:

<b>Título del PI:</b> Explorador de antipatrón de vulnerabilidades de Wordpress
<b>Descripción:</b> <i>Ver Anexo.</i>
<b>Palabras Claves (Key Words, mín 3 máx 5):</b> Software engineering, Software Design, Software Safety . <b>(de acuerdo IEEE Taxonomy_v1.03)</b>
<b>Área temática:</b> Software engineering.
<b>Asignaturas (seleccionar entre 1 y 4 asignaturas que tengan relación directa con el desarrollo del PI):</b> Ingeniería de Software, Sistemas Operativos II, Redes de Computadoras, Criptografía y Seguridad en Redes, Sistemas de Gestión de Bases de Datos.-

**Director del Proyecto Integrador.**

Nombre: Miguel Solinas
Cargo: Profesor Titular DE
Dirección Personal o Laboral: Av. Vélez Sarsfield 1611
Teléfono celular: 351.2771379
E-mail: <a href="mailto:miguel.solinas@unc.edu.ar">miguel.solinas@unc.edu.ar</a>
Firma del Director: 

**Co-director del Proyecto Integrador**

Nombre: Javier Alejandro JORGE

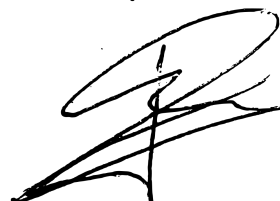
Cargo: Profesor Adjunto DS Regular

Dirección Personal o Laboral: Av. Vélez Sarsfield 1611

Teléfono celular: +54 9 3513 23-5866

E-mail: [javier.jorge@unc.edu.ar](mailto:javier.jorge@unc.edu.ar)

Firma del Co-Director:

**Datos del Alumno**

Nombre y Apellido: Gina Commisso

Matricula: 39446783

Asignaturas que adeuda:(requisito establecido en el Art. 4 del Reglamento PI):  
Sistemas Operativos 1, Sistemas Operativos 2, Arquitectura de Computadoras

Dirección: Av. Ciudad de Valparaíso 7000

Localidad: Córdoba Capital

Provincia: Córdoba

Teléfono celular: 3512-702702

E-mail: [ginacommisso@unc.edu.ar](mailto:ginacommisso@unc.edu.ar)

Firma:



**Objetivo:**

Diseñar una aplicación web para recuperación de información sobre antipatrones de vulnerabilidades vinculadas al Content Management System (CMS) Wordpress.

**Antecedentes de Trabajos Similares:**

- API Rest desarrollada en GO para autenticar usuarios y devolver información de sensores IoT , Trabajo Práctico VI de Sistemas Operativos 2, Ingeniería en Computación, FCEFYN, 2023.
- Actividades en el contexto de la Práctica Supervisada, relacionadas con vulnerabilidades de Wordpress. Informe de Práctica Supervisada, Ingeniería en Computación, FCEFYN, 2023.

**Duración y Fases de las tareas previstas:** Ver Diagrama de Gantt adjunto en Anexo

**Metodología:** Lugar previsto de Realización (indicar empresa, laboratorio o centro donde se desarrollará el trabajo): Prosecretaría de Informática de la UNC; Laboratorio de Redes y Ciberseguridad (LaRyC), FCEFYN, UNC.

**Requerimientos en instrumental y equipos:** Notebooks para trabajar; acceso a internet; servidores virtuales varios.

**Inversión estimada prevista por el alumno:** Ninguna.

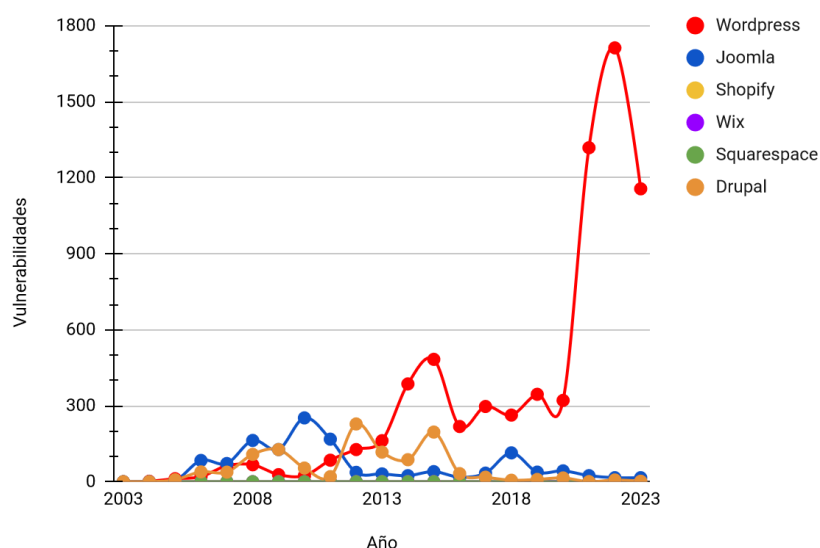
**Apoyo Económico Externo a la Facultad:** No.

**Referencias Bibliográficas:**

- [1] CVE Homepage, <https://www.cve.org/>. Last accessed 17 May 2024.
- [2] Microsoft Threat Modeling Tool, <https://acortar.link/tvesdb>. Last accessed 17 May 2024.
- [3] IONOS CMS en comparativa, <https://acortar.link/VkuQLQ>. Last accessed 4 Abr 2024
- [4] Brown, W. J.: AntiPatterns: Refactoring software, architectures, and projects in crisis. Wiley, New York (1998).
- [5] Corry, A.: Retrospectives Antipatterns; Addison-Wesley (2021).
- [6] CYBOK, <https://www.cybok.org/>. Last accessed 4 Abr 2024.
- [7] SWEBoK, <https://acortar.link/HnMV97>. Last accessed 4 Abr 2024.
- [8] Nafees, T: Addressing the Knowledge Transfer Problem in Secure Software Development Through Anti-Patterns. PhD Thesis. Abertay University. Dundee, Escocia, UK (2019).

## Anexo

Las vulnerabilidades cuantifican la superficie de ataque digital de un sistema. En particular el stack de componentes expuestos a una red pública o dentro de una red privada compleja incluyen en su versión mínima: firmware, sistema operativo, frameworks varios y múltiples aplicaciones con sus propias arquitecturas. Todos ellos tienen vulnerabilidades conocidas y registradas en bases de datos como CVE[1]. Dichas vulnerabilidades constituyen una información útil en la construcción de nuevas aplicaciones de software en la medida en que no se repitan. Sentido común que no tiene su correlato en la realidad y las vulnerabilidades conocidas se reiteran en nuevos sistemas todo el tiempo. ¿Cómo es posible que esto ocurra? Si bien los modelos de proceso de construcción de software tienen una madurez que aportan técnicas para minimizar el problema, por ejemplo la construcción de modelos de amenazas[2]; es frecuente que las herramientas utilizadas y urgencias de entregar funcionalidad conduzcan el desarrollo en la dirección de prácticas que repiten vulnerabilidades conocidas. Más la falta de formación formal en aspectos de ciberseguridad conducen a una escasa conciencia de las consecuencias de repetir fallas conocidas. Por otro lado, cuando se construye software utilizando Content Management Systems (CMS) los 10 principales frameworks representan más del 86 % del market share, con un 65,1 % de WordPress[3]. Cuando se utiliza este framework, existe una exposición a la historia de sus vulnerabilidades. En la Figura se muestra la evolución de la superficie de ataque de los principales CMS del mercado, con año en abscisas y vulnerabilidades descubiertas en ordenadas. Una curva llama la atención, la de WordPress. Sumado a su popularidad no es de extrañar que el escenario conduce a usuarios expuestos.



## **Antipatrón de vulnerabilidades**

Las vulnerabilidades conocidas se registran desde finales del siglo XX en CVE. Y sea una vulnerabilidad de día cero o conocida, se trata de un resultado indeseable que genera decididamente consecuencias negativas para los usuarios. Esto es un antipatrón [4][5]. Documentar vulnerabilidades como vulnerability antipattern (VAP) permite traducir conocimiento del dominio de la ciberseguridad[6] al de la ingeniería de software[7] y dependiendo de los autores, la documentación contiene desde una plantilla estándar hasta recomendaciones en términos de acciones para las etapas de requerimientos, diseño e implementación[8] que guían al desarrollador en la dirección de un software seguro.

## **Estado actual de las vulnerabilidades de Wordpress**

Las vulnerabilidades de WordPress que figuran en CVE (Common Vulnerabilities and Exposures) están parcheadas. Sin embargo, no siempre se desarrollan proyectos utilizando la última versión del CMS. Además, muchos sitios mantienen versiones anteriores activas para garantizar funcionalidades ya implementadas, lo cual no es aconsejable y pone en riesgo los activos de información.

Algunos plugins dependen de versiones específicas de otros plugins o del mismo WordPress. Si un plugin se actualiza y el otro no es compatible con la nueva versión, puede dejar de funcionar correctamente. Ej.: WooCommerce (para tiendas en línea) depende de WooCommerce Subscriptions (para productos basados en suscripciones)

## **Fuentes Utilizadas**

La fuente de información sobre vulnerabilidades es Common Vulnerabilities and Exposures (CVE), Common Weakness Enumeration (CWE), Common Attack Pattern Enumeration and Classification (CAPEC) de Mitre Corporation.

## **Transformaciones de los datos para elaborar la base de datos**

A partir de las vulnerabilidades registradas en CVE para el CMS y sus plugins, se recuperará información precisa de CWE y CAPEC, los (Security Pattern) SP, (Security Fault Patterns) SFP y recomendaciones de STRIDE para construir un Vulnerability AntiPattern (VAP).

Por último se documentarán los VAP utilizando una plantilla basada en otros trabajos de investigación, integrando información de las fuentes mencionadas.

## Aporte potencial del trabajo

Este proyecto facilitará la identificación de vulnerabilidades y debilidades, proporcionando un recurso valioso para desarrolladores al ofrecer recomendaciones basadas en antipatrones.

Esto se materializará con la construcción de un plugin para el CMS que haga efectiva la tarea de escanear las vulnerabilidades, matchearlas con los VAP y muestre las recomendaciones pertinentes para las etapas de requerimientos, diseño e implementación. Por otro lado la universidad tiene miles de sitios desarrollados con este CMS y sería un aporte importante contar con un método y una herramienta que facilite minimizar riesgos.

## Objetivos Parciales

1. Analizar documentación referida a antipatrón de vulnerabilidades.
2. Diseñar template de antipatrón.
3. Releva los antipatrones de vulnerabilidades más relevantes, vinculados a Wordpress.
4. Persistir en una base de datos los VAP relevados.
5. Diseñar una API Rest para permitir consulta de antipatrones.
6. Diseñar un front end para recuperar información de categorías de VAP en particular usando la API Rest.
7. Publicar el trabajo realizado.
8. Escribir informe de proyecto integrador.

## Diagrama de Gantt

T A R D E A	Quincena																
	Jun 1	Jun 2	Jul 1	Jul 2	Ago 1	Ago 2	Sep 1	Sep 2	Oct 1	Oct 2	Nov 1	Nov 2	Dic 1	Dic 2	Ene 1	Ene 2	Feb 1
1	X	X															
2		X	X														
3			X	X													
4				X	X	X											
5					X	X	X										
6							X	X	X	X	X						
7											X						
8											X	X	X	X	X	X	X

