Universidad de Oviedo

MÁSTER UNIVERSITARIO EN INGENIERÍA WEB

Orientación [Seguridad Web]

Curso 2024/2025

ARAGOG

Seguridad Automatizada.

AUTOR: AGUSTÍN WIDMAN AGUAYO

TUTOR: JOSE MANUEL REDONDO LÓPEZ

ÍNDICE

Resumen

El presente trabajo describe el diseño y desarrollo de *Aragog*, un sistema automatizado para el despliegue de servidores web seguros sobre una máquina virtual preconfigurada. Su objetivo es proporcionar entornos web aislados, reproducibles y alineados con estándares internacionales de seguridad como CIS Benchmark y OWASP, facilitando su uso en contextos educativos o institucionales. Aragog combina tecnologías como Vagrant, Docker, Ubuntu Pro y herramientas de protección activa (Fail2Ban, ModSecurity) para ofrecer una solución modular, escalable y de bajo mantenimiento. La automatización garantiza homogeneidad, reduce errores humanos y permite desplegar, en minutos, sistemas listos para producción o docencia.

Palabras clave: seguridad web, automatización, contenedores, CIS Benchmark, OWASP, virtualización, Docker, Vagrant, infraestructura educativa segura.

1. INTRODUCCIÓN

ARAGOG

Seguridad Automatizada.

1.1 Motivación

En entornos académicos, institucionales o de investigación, el despliegue de servidores web suele realizarse a partir de máquinas virtuales configuradas por defecto, sin contemplar medidas de seguridad avanzadas. Esta práctica representa un riesgo significativo, ya que los sistemas operativos y servicios web, en su estado inicial, presentan vulnerabilidades comunes como configuraciones inseguras, servicios innecesarios activos o ausencia de controles de acceso refinados que pueden ser fácilmente explotados.

Aunque existen guías de endurecimiento como los benchmarks del Center for Internet Security[1] y herramientas especializadas como ModSecurity[2] o Fail2Ban[3], su correcta integración y automatización sigue siendo una tarea compleja y propensa a errores si se realiza manualmente. Esta falta de soluciones reproducibles, seguras y fáciles de gestionar, especialmente adaptadas a entornos educativos multiusuario, pone de manifiesto la necesidad de herramientas que garanticen la seguridad desde el diseño sin comprometer la facilidad de uso o la escalabilidad.

1.2 Finalidad del proyecto

El objetivo principal de este proyecto es desarrollar un sistema automatizado capaz de generar entornos web seguros a través de una máquina virtual preconfigurada y endurecida por medio de ubuntu pro CIS Benchmark[4], que permita desplegar servidores Apache[5] o Nginx[6] aislados mediante contenedores. El sistema, denominado *Aragog*, busca garantizar que cada servidor generado cumpla desde su inicio con buenas prácticas de seguridad, configuraciones estandarizadas y controles de protección activos, sin necesidad de intervención manual.

A diferencia de soluciones genéricas que requieren configuración manual o conocimientos avanzados, *Aragog* integra herramientas ampliamente utilizadas de automatización para generar una arquitectura modular, que permite su adaptación a diferentes escenarios. Su valor diferencial reside en su capacidad para automatizar todo el proceso, desde el endurecimiento del sistema operativo hasta la entrega de un entorno web funcional y seguro para cada usuario, con especial atención a entornos educativos, donde la coexistencia de múltiples usuarios y la trazabilidad son críticas.

2. ÁMBITOS DE APLICACIÓN

2.1 Posibles Ámbitos de Aplicación

El sistema Aragog presenta un enfoque versátil que permite su aplicación en diversos escenarios, destacando entre ellos:

- Entornos educativos: Ideal para laboratorios de formación en ciberseguridad, administración de sistemas o desarrollo web, donde se requiere un entorno aislado, seguro y replicable para cada estudiante.
- Proyectos de investigación: Facilita el despliegue rápido de servidores web en entornos controlados, garantizando estándares de seguridad exigidos por políticas de cumplimiento.
- Pequeñas organizaciones o startups: Ofrece una solución robusta para desplegar servidores web con buenas prácticas de seguridad sin necesidad de conocimientos avanzados en configuración manual.
- Auditorías y pruebas de cumplimiento: Gracias a sus capacidades de autoevaluación y generación de informes normativos, el sistema puede ser utilizado como herramienta de validación frente a marcos de referencia como CIS[1] o OWASP[7].

Su modularidad, junto a la automatización de procesos clave, posicionan a Aragog como una solución adaptable, escalable y fácilmente integrable en infraestructuras reales que requieran seguridad por diseño desde las primeras etapas del despliegue.

3. PLANIFICACIÓN Y GESTIÓN DEL TFM

3.1Planificación del Proyecto

3.1.1 Identificación de Interesados

En esta sección se muestra la lista de interesados en el proyecto. Además del Autor y del director de proyecto se ha tenido en cuenta otro estudiante en calidad de pruebas y validación del sistema en distintos entornos de despliegue, cumpliendo un rol activo en la detección de errores y evaluación de usabilidad.

Nombre	Cargo	Departamento
Agustín Widman Aguayo	Autor del Proyecto	Universidad de Oviedo
José Manuel Redondo López	Tutor del Proyecto	Universidad de Oviedo
Administrador de sistemas	Administrador de ARAGOG	Universidad de Oviedo
Estudiantes Usuarios Finales	Usuarios ARAGOG	Universidad de Oviedo

Tabla 3.1 Lista de Interesados

3.1.2 OBS Y PBS

En la siguiente tabla se muestran los recursos del OBS (Organization Breakdown Structure) que forma parte del proyecto.

Nombre del recurso	Tipo	Iniciales	Capacidad Máxima	Tasa Estándar	Tasa Horas Extra	Costo/ Uso	Acumular	Calendario Base
Agustín Widman Aguayo	Trabajo	A	1	25,00 €/hora	0,00 €/hora	0	Prorrateo	TFM
José Manuel Redondo López	Trabajo	J	0,2	30,00 €/hora	0,00 €/hora	0	Prorrateo	TFM

Tabla 3.2 OBS

En el siguiente diagrama se presenta el PBS (Product Breakdown Structure), que recoge todos los productos entregables del proyecto.

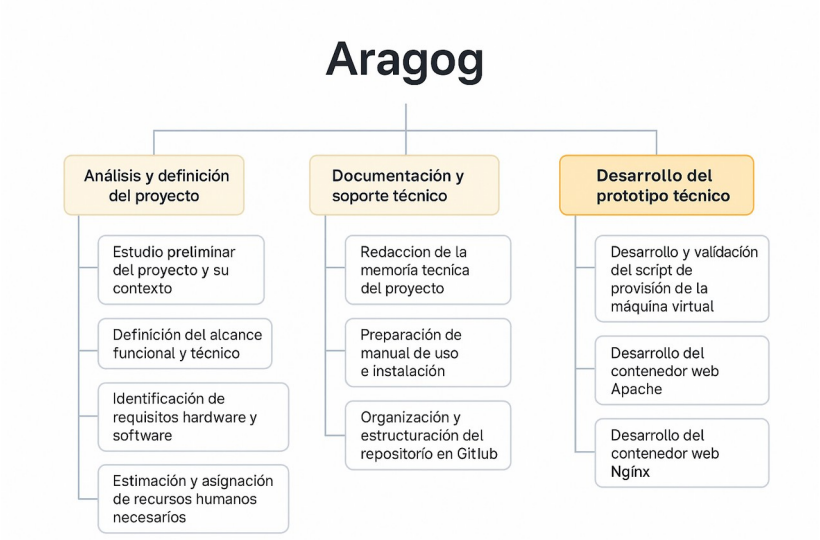


FIGURA 3.1: PBS del proyecto

3.1.3 Planificación inicial. WBS

En este apartado se desglosa el WBS (Work Breakdown Structure) inicial. Para cada tarea se muestra su ID, su duración, fechas de comienzo y fin y tareas predecesoras.

EDT	Nombre de Tarea	Duración (días)	Comienzo	Fin
1	Reuniones mensuales	4	15/03/2025	26/05/2025
2	Documentación e Investigación	20	17/03/2025	11/04/2025
3	Definición y diseño del prototipo	15	10/04/2025	30/04/2025
4	Desarrollo del prototipo	28	01/05/2025	31/05/2025
5	Desarrollo de los contenidos del Proyecto	18	01/06/2025	24/06/2025
6	Desarrollo de la Memoria del Proyecto	22	19/06/2025	15/07/2025

Tabla 3.3 Tabla de planificación de tareas resumida del proyecto

A continuación, se puede observar la Tabla de planificación de tareas completa del proyecto:

EDT	Tarea	Duración	Comienzo	Fin	Predecesoras
1	Reuniones Mensuales	4 días	15/03/2025	26/05/2025	-
1.1	Reuniones Mensuales 1	2 horas	15/03/2025	15/03/2025	-
1.2	Reuniones Mensuales 2	2 horas	15/04/2025	15/04/2025	-
1.3	Reuniones Mensuales 3	2 horas	12/05/2025	12/05/2025	-
1.4	Reuniones Mensuales 4	2 horas	26/05/2025	26/05/2025	-
2	Documentación e Investigación	20 días	17/03/2025	09/04/2025	-
2.1	Revisión de requisitos del TFM	2 días	17/03/2025	18/03/2025	-
2.2	Investigación sobre hardening en Linux y Ubuntu Pro	3 días	19/03/2025	21/03/2025	2.1
2.3	Investigación sobre Vagrant y VirtualBox	2 días	24/03/2025	25/03/2025	2.1
2.4	Análisis de herramientas de seguridad (Fail2Ban, PortSentry)	3 días	26/03/2025	28/03/2025	2.1
2.5	Investigación sobre Docker y contenedores seguros	4 días	31/03/2025	03/04/2025	2.1
2.6	Estudio de Apache y Nginx seguros con ModSecurity	3 días	04/04/2025	06/04/2025	2.1
2.7	Análisis de OWASP CRS y su aplicación práctica	2 días	07/04/2025	08/04/2025	2.6
2.8	Investigación sobre automatización de procesos de seguridad en sistemas	1 día	09/04/2025	09/04/2025	2.1
3	Definición y diseño del prototipo	15 días	10/04/2025	30/04/2025	2.1
3.1	Definición de los objetivos funcionales y de seguridad del prototipo	2 días	10/04/2025	11/04/2025	2.1
3.2	Diseño de la arquitectura general del sistema Aragog	3 días	14/04/2025	16/04/2025	3.1
3.3	Definición de los componentes principales (VM base, contenedores, scripts)	3 días	17/04/2025	19/04/2025	3.2
3.4	Especificación de flujos de uso para administradores y alumnos	2 días	21/04/2025	22/04/2025	3.2
3.5	Diseño del entorno multiusuario y política de aislamiento	2 días	23/04/2025	24/04/2025	3.3
3.6	Diseño de la estructura de scripts automatizados	2 días	25/04/2025	26/04/2025	3.3
3.7	Validación del diseño con el consultor técnico o tutor	1 día	29/04/2025	29/04/2025	3.1 a 3.6
4	Desarrollo del prototipo	28 días	01/05/2025	31/05/2025	3.7
4.1	Desarrollo del script de provisión automatizada (provision.sh)	5 días	01/05/2025	07/05/2025	3.7
4.2	Configuración segura de la máquina virtual base (Ubuntu Pro + hardening)	4 días	06/05/2025	09/05/2025	3.7, 4.1
4.3	Integración y configuración de herramientas de seguridad (Fail2Ban, PortSentry)	3 días	10/05/2025	12/05/2025	4.2
4.4	Creación del script de despliegue para alumnos (crear_alumno.sh)	4 días	13/05/2025	16/05/2025	4.1
4.5	Desarrollo del contenedor Docker seguro para Apache	3 días	13/05/2025	15/05/2025	4.1
4.6	Desarrollo del contenedor Docker seguro para Nginx	3 días	17/05/2025	19/05/2025	4.5
4.7	Implementación del script de reparación de permisos (reparar_permisos.sh)	2 días	20/05/2025	21/05/2025	4.4
4.8	Automatización del mapeo de volúmenes y puertos para contenedores	3 días	22/05/2025	24/05/2025	4.6, 4.7
4.9	Pruebas de funcionamiento y corrección de errores del sistema Aragog	4 días	27/05/2025	31/05/2025	4.1-4.8
5	Desarrollo de los contenidos del Proyecto	18 días	01/06/2025	24/06/2025	4.9
5.1	Elaboración de descripciones técnicas de cada componente del sistema	3 días	01/06/2025	03/06/2025	4.9
5.2	Desarrollo de diagramas de arquitectura y flujo de trabajo	4 días	04/06/2025	07/06/2025	5.1
5.3	Redacción de instrucciones para el uso del sistema por parte de alumnos	2 días	08/06/2025	09/06/2025	5.1
5.4	Preparación de documentación para administradores y	3 días	10/06/2025	12/06/2025	5.2

	responsables técnicos				
5.5	Generación y organización del contenido gráfico (capturas, esquemas, etc.)	3 días	13/06/2025	15/06/2025	5.2, 5.4
5.6	Desarrollo de la tabla de riesgos y planificación de respuesta	2 días	16/06/2025	17/06/2025	4.9
5.7	Elaboración del WBS y planificación detallada de tareas del proyecto	1 día	18/06/2025	18/06/2025	5.1-5.6
6	Desarrollo de la Memoria del Proyecto	22 días	19/06/2025	15/07/2025	5.7
6.1	Introducción y Fijación de objetivos	2 días	19/06/2025	20/06/2025	5.7
6.2	Planificación y Gestión del TFM	3 días	21/06/2025	23/06/2025	5.7
6.3	Estado Actual de los Conocimientos Científico-Técnicos	4 días	24/06/2025	27/06/2025	6.1
6.4	Descripción del Sistema	5 días	28/06/2025	02/07/2025	6.2, 6.3
6.5	Metodología del trabajo	3 días	03/07/2025	05/07/2025	6.4
6.6	Resultados Obtenidos	3 días	06/07/2025	08/07/2025	6.5
6.7	Conclusiones y trabajo futuro	1 día	09/07/2025	09/07/2025	6.6
6.8	Realización de la memoria del proyecto	1 día	10/07/2025	10/07/2025	6.1-6.7

Tabla 3.4 Tabla de planificación de tareas del proyecto

3.1.4 Riesgos del proyecto

En esta sección se encuentran los apartados relativos a la gestión de riesgos.(Anexos Riesgos)

3.1.4.1 Plan de gestión de riesgos

La documentación correspondiente al plan de gestión de riesgos se encuentra en el anexo 10.1 de este documento.

3.1.4.2 Identificación de riesgos

A continuación, se presenta una tabla con los principales riesgos identificados en el desarrollo del proyecto, incluyendo los siguientes elementos:

- **ID**: Código único asignado a cada riesgo para su referencia y seguimiento.
- **Nombre**: Breve descripción que define la naturaleza del riesgo.
- **Responsable**: Personas o roles vinculados directa o indirectamente con la aparición, gestión o mitigación del riesgo.
- **Probabilidad**: Estimación de la posibilidad de que el riesgo se materialice. Esta valoración se apoya en la escala definida en la *Matriz de Probabilidad e Impacto*, contemplada en el Plan de Gestión de Riesgos.
- **Impacto**: Nivel de afectación que tendría el riesgo sobre el proyecto, considerando variables como presupuesto, cronograma, alcance o calidad. La evaluación también sigue la escala establecida en la *Matriz de Probabilidad e Impacto*.

- **Priorización**: Valor numérico entre 0 y 0.5 que indica la criticidad del riesgo, y por tanto, la necesidad de seguimiento continuo.
- **Respuesta**: Estrategia o acciones previstas a adoptar en caso de que el riesgo se concrete.

3.1.4.3 Registro de riesgos

ID	Nombre	Responsable
1	Baja disponibilidad del autor durante el desarrollo	Autor del Proyecto
2	Problemas técnicos con VirtualBox o Vagrant incompatibles con el sistema anfitrión	Autor del Proyecto
3	Fallos en la activación o caducidad del token de Ubuntu Pro	Administrador de ARAGOG
4	Errores de configuración en Docker o conflictos entre contenedores	Administrador de ARAGOG
5	Cambios de última hora en los criterios del tutor o en los requisitos del TFM	Tutor del Proyecto
6	Fallos de seguridad no detectados en los contenedores generados	Autor del Proyecto
7	Retrasos en la redacción de la memoria y estructuración de contenidos	Autor del Proyecto
8	Incapacidad para realizar pruebas reales con estudiantes	Usuarios Aragog
9	Inestabilidad del entorno de pruebas o del equipo anfitrión	Autor del Proyecto
10	Pérdida o corrupción del repositorio de GitHub	Administrador de ARAGOG
11	Desacuerdo del Administrador de ARAGOG con aspectos clave del diseño o implementación	Autor del Proyecto
12	Dificultades para documentar correctamente los scripts y procesos técnicos	Autor del Proyecto
13	Problemas de visibilidad o comunicación del proyecto entre departamentos interesados	Tutor del Proyecto
14	Posible obsolescencia de versiones de dependencias o herramientas antes de la presentación	Administrador de ARAGOG
15	Problemas en la validación de resultados ante entornos distintos al original	Autor del Proyecto
16	Fallos en la automatización que impidan una replicación exacta del sistema	Administrador de ARAGOG

Tabla 3.5 Registro de Riesgos

ID	**Probabilidad**	**Presup.**	**Planific.**	**Alcance**	**Calidad**	**Impacto Total**	**Priorización**
1	Baja	Alto	Medio	Critico	Medio	0,27	
2	Medio	Medio	Critico	Alto	Medio	0,5	
3	Alto	Bajo	Critico	Alto	Alto	0,7	
4	Medio	Bajo	Critico	Alto	Medio	0,5	
5	Medio	Bajo	Alto	Medio	Medio	0,3	
6	Medio	Bajo	Medio	Alto	Critico	0,4	
7	Medio	Bajo	Critico	Medio	Alto	0,25	
8	Medio	Bajo	Medio	Alto	Alto	0,3	
9	Alto	Medio	Alto	Bajo	Medio	0,3	
10	Medio	Medio	Alto	Alto	Critico	0,65	
11	Medio	Bajo	Critico	Medio	Alto	0,5	
12	Medio	Bajo	Medio	Medio	Alto	0,45	
13	Medio	Bajo	Medio	Alto	Medio	0,45	
14	Medio	Medio	Alto	Medio	Critico	0,65	
15	Medio	Bajo	Medio	Alto	Alto	0,55	
16	Medio	Bajo	Alto	Critico	Alto	0,65	

Tabla 3.6 Probabilidad e Impacto

ID	Impacto	Priorización	Respuesta
1	0,27		Planificar hitos realistas, documentar avances regularmente y habilitar espacios de consulta asíncrona, por ejemplo, vía repositorio o notas compartidas
2	0,5		Evaluar herramientas alternativas compatibles como Docker nativo, establecer un entorno mínimo validado por adelantado y mantener documentación de configuración para soporte rápido.
3	0,7		Automatizar la renovación y validación del token en los scripts de despliegue, y mantener un registro actualizado de su estado y fechas de expiración.
4	0,5		Definir configuraciones estándar, utilizar archivos de composición bien documentados y realizar pruebas aisladas de cada contenedor antes de integrarlos.
5	0,3		Establecer revisiones periódicas con el tutor y mantener una trazabilidad clara de decisiones y cambios acordados.
6	0,4		Integrar herramientas de análisis de vulnerabilidades en los scripts y realizar pruebas de seguridad automatizadas antes de cada entrega
7	0,25		Dividir la redacción en entregas parciales con fechas internas y usar herramientas de control de versiones para avanzar progresivamente
8	0,3		Coordinar pruebas piloto con usuarios clave previamente identificados y simular escenarios comunes en un entorno controlado con retroalimentación
9	0,3		Preparar entornos alternativos, realizar respaldos frecuentes y mantener la infraestructura bajo control con herramientas de monitoreo básicas
10	0,65		Establecer backups automáticos del repositorio, aplicar ramas protegidas y activar autenticación multifactor para mantener la integridad del proyecto.
11	0,5		Acordar desde el inicio una agenda de revisión con entregas parciales, y mantener comunicación continua por canales asíncronos
12	0,45		Adoptar una plantilla de documentación desde el principio, escribir mientras se desarrolla y revisar con checklist al finalizar cada módulo
13	0,45		Compartir avances en puntos de control definidos, involucrar desde temprano a los interesados clave y documentar decisiones de forma accesible.
14	0,65		Congelar versiones estables en los scripts, documentar dependencias clave y verificar compatibilidad antes de cada entrega
15	0,55		Definir un entorno base reproducible, incluir validaciones automáticas en diferentes sistemas y documentar requisitos mínimos de ejecución
16	0,65		Realizar pruebas de replicación desde cero en distintos equipos, mantener los scripts bajo control de versiones y registrar los pasos críticos del despliegue

Tabla 3.7 Impacto y Respuesta

3.1.5 Presupuesto Inicial

En esta sección se encuentran los apartados relativos al presupuesto del proyecto.

3.1.5.1 Presupuesto de costes

En esta sección se encuentra el presupuesto de costes. Las horas de reuniones se suman a la partida de planificación inicial. Las tarifas para cada recurso se muestran en la siguiente tabla.

Las tarifas para cada recurso se muestran en la siguiente tabla.

Perfil	Precio Coste	Precio Venta	Horas Invertidas
Agustín Widman Aguayo	25,00 €/hora	45,00 €/hora	280
José Manuel Redondo López	30,00 €/hora	50,00 €/hora	14

Tabla 3.8 Tarifa de Recursos

Para calcular el precio de venta se han estimado los siguientes costes. Los gastos e ingresos del personal se calculan sobre las horas que dedican a este proyecto.

Concepto	Ingresos	Gastos
Personal	13.300,00 €	7.420,00 €
Equipos y Licencias	0,00 €	30,00 €
Servicios	0,00 €	100,00 €
Formación	0,00 €	150,00 €
Costes financieros (1,00% de los ingresos)	0,00 €	168,00 €
Otros costes (asesoría instalación)	3.500,00 €	500,00 €
Subtotal	16.800,00 €	8.368,00 €
Beneficios	8.432,00 €	

Tabla 3.9 Costes

3.1.5.2 Presupuesto de cliente

ARAGOG – SEGURIDAD AUTOMATIZADA				
Partida	**Item**	**Partida**	**Importe**	**Total**
1		Reuniones mensuales	497,20 €	497,20 €
2		Documentación e Investigación	2.685,98 €	2.685,98 €
3		Definición y diseño del prototipo		2.324,49 €
	1	Definición de requerimientos y funcionalidades	1.323,71 €	
	2	Diseño de arquitectura de prototipo	1.000,78 €	
4		Desarrollo del prototipo		4.080,37 €
	1	Codificación y pruebas del prototipo	3.300,31 €	
	2	Optimización y mejoras funcionales	780,06 €	
5		Desarrollo de los contenidos del Proyecto	2.387,38 €	2.387,38 €
6		Desarrollo de la Memoria del Proyecto	2.926,58 €	2.926,58 €
	1	Redacción y estructuración de la memoria final	2.926,58 €	
7		Gestión de riesgos e integración de seguridad	1.100,00 €	1.100,00 €
8		Revisión normativa y aseguramiento de calidad	948,00 €	948,00 €
9		Presentación, entrega final y soporte de cierre	700,00 €	700,00 €
		TOTAL CLIENTE		**18.650,00 €**

Tabla 3.10 Presupuesto del cliente

ARAGOG – SEGURIDAD AUTOMATIZADA		
Codigo	**Partida**	**Total**
1	Reuniones mensuales	497,20 €
2	Documentación e Investigación	2.685,98 €
3	Definición y diseño del prototipo	2.324,49 €
4	Desarrollo del prototipo	4.080,37 €
5	Desarrollo de contenidos del proyecto	2.387,38 €
6	Desarrollo de la memoria del proyecto	2.926,58 €
7	Gestión de riesgos, revisión normativa y cierre	3.747,00 €
	TOTAL CLIENTE	**18.650,00 €**

Tabla 3.11 Presupuesto del cliente Resumido

3.2 Ejecución del proyecto

3.2.1 Plan seguimiento de planificación

En este apartado se detalla el seguimiento de las seis áreas principales que estructuran el desarrollo del proyecto (Reuniones mensuales, Documentación e Investigación, Definición y diseño del prototipo, Desarrollo del prototipo, Desarrollo de los contenidos del proyecto y Desarrollo de la memoria del proyecto).

El trabajo comenzó con la ejecución completa de la fase de Documentación e Investigación, la cual fue fundamental para adquirir una base sólida sobre la automatización de entornos seguros, las herramientas de hardening en sistemas Linux, el uso de contenedores con Apache/Nginx[5][6] y la integración de mecanismos de seguridad como ModSecurity[2], Fail2Ban[3] o PortSentry[8]. Esta etapa también permitió analizar soluciones existentes, identificar buenas prácticas y justificar las decisiones técnicas adoptadas en el diseño del sistema Aragog.

Una vez consolidado el conocimiento necesario, se procedió a la definición y diseño del prototipo, especificando la arquitectura general, la estructura de los scripts de automatización, los flujos de trabajo y el modelo multiusuario. Sobre esta base se llevó a cabo la fase de desarrollo del prototipo, implementando una máquina virtual con hardening automatizado y contenedores seguros configurados para su despliegue individualizado por alumno.

Completado el desarrollo funcional, se avanzó en la generación de contenidos técnicos y materiales de apoyo, incluyendo diagramas, capturas, documentación técnica para usuarios y administradores, así como la definición del WBS y la tabla de riesgos del proyecto.

Finalmente, se abordó el proceso de redacción de la memoria, consolidando la información técnica y metodológica, y preparando el documento final que reflejará tanto el desarrollo del sistema como las conclusiones, resultados y lecciones aprendidas durante la ejecución del proyecto.

3.2.2 Bitácora de incidencias del proyecto

A continuación, se comentará un resumen de las incidencias que han ido ocurriendo durante el desarrollo del proyecto. Aunque la mayoría de los riesgos identificados fueron mitigados correctamente, durante el proceso se presentaron algunas situaciones relacionadas con la compatibilidad de herramientas, la estabilidad del entorno de pruebas y ajustes necesarios en los procesos de automatización. También se tomaron medidas anticipadas para minimizar el impacto de

posibles fallos en el despliegue o uso del sistema por parte de los alumnos. Estas incidencias fueron gestionadas conforme surgieron, y se detallan a continuación junto con las acciones adoptadas para su resolución.

3.3.3 Riesgos

ID	Nombre del Riesgo	Seguimiento
1	Baja disponibilidad del autor durante el desarrollo	Se establecieron objetivos semanales flexibles y se documentó todo el desarrollo en scripts y el repositorio compartido, lo cual permitió continuar el avance incluso en momentos de baja disponibilidad.
2	Problemas técnicos con VirtualBox o Vagrant incompatibles con el sistema anfitrión	No se detectaron incompatibilidades durante el desarrollo. El entorno base se mantuvo estable y funcional en todo momento gracias a la validación previa.
3	Fallos en la activación o caducidad del token de Ubuntu Pro	Se dejó documentado en el manual de uso una advertencia para el administrador de Aragog, indicando la necesidad de renovar el token periódicamente, ya que su caducidad impediría el uso de funcionalidades como el USG.
4	Errores de configuración en Docker o conflictos entre contenedores	No se presentaron conflictos durante el despliegue. Los contenedores fueron configurados con parámetros estandarizados y validados de forma individual antes de su integración, lo que garantizó un funcionamiento correcto.
5	Cambios de última hora en los criterios del tutor o en los requisitos del TFM	No se produjeron cambios inesperados. La coordinación con el tutor fue fluida y se mantuvieron reuniones regulares que permitieron alinear el desarrollo con los requisitos desde el inicio.
6	Fallos de seguridad no detectados en los contenedores generados	Se llevaron a cabo diversas pruebas de seguridad, incluyendo intentos de ataque controlado desde el host hacia la máquina virtual y los contenedores. Además, se utilizó el informe de auditoría para validar el cumplimiento del perfil de seguridad CIS aplicado.
7	Retrasos en la redacción de la memoria y estructuración de contenidos	La documentación se fue realizando de forma continua a lo largo del desarrollo. El avance progresivo y el uso del control de versiones facilitaron una redacción ordenada y sin retrasos significativos.
8	Incapacidad para realizar pruebas reales con estudiantes	Se realizaron pruebas controladas que simularon el uso por parte de estudiantes, verificando la funcionalidad completa del sistema. Estas pruebas permitieron asegurar la estabilidad y el correcto aislamiento de los entornos desplegados.
9	Inestabilidad del entorno de pruebas o del equipo anfitrión	Se realizaron pruebas en distintas distribuciones de Linux para verificar la compatibilidad y estabilidad del sistema. En todos los casos, la solución funcionó correctamente sin presentar fallos.
10	Pérdida o corrupción del repositorio de GitHub	Se incluyó en el manual de uso una recomendación explícita para realizar respaldos locales tras el clonado del repositorio, como medida preventiva ante posibles pérdidas o corrupciones.
11	Desacuerdo del Administrador de ARAGOG con aspectos clave del diseño o implementación	La agenda de revisiones se cumplió según lo previsto, y se utilizó el repositorio de GitHub como canal principal para compartir avances. Esto permitió que el consultor técnico pudiera realizar seguimientos y aportar comentarios en cualquier momento.
12	Dificultades para documentar correctamente los scripts y procesos técnicos	La documentación técnica se mantuvo actualizada durante todo el desarrollo. Los scripts fueron comentados progresivamente conforme se avanzaba, lo que facilitó su comprensión y

		mantenimiento posterior.
13	Problemas de visibilidad o comunicación del proyecto entre departamentos interesados	Se notificaron los avances relevantes tanto al tutor como al consultor técnico por correo electrónico, y se mantuvo actualizado el repositorio en GitHub para facilitar el acceso a la información en todo momento.
14	Posible obsolescencia de versiones de dependencias o herramientas antes de la presentación	Se utilizaron versiones estables y se documentaron todas las dependencias relevantes. No se detectaron problemas de compatibilidad, y todo el sistema funcionó correctamente durante las pruebas finales.
15	Problemas en la validación de resultados ante entornos distintos al original	Se documentó que Aragog está diseñado para funcionar sobre sistemas Linux. Para entornos Windows, se recomienda el uso de una máquina virtual con Linux.
16	Fallos en la automatización que impidan una replicación exacta del sistema	Se realizaron pruebas de replicación desde cero en distintos entornos y equipos, confirmando que el sistema se despliega correctamente. Los scripts se mantuvieron versionados en GitHub y se documentaron todos los pasos relevantes.

Tabla 3.12 Seguimiento de los Riesgos del Proyecto

3.3 Cierre del proyecto.

3.3.1 Planificación Final

Durante la ejecución del proyecto, se produjeron desviaciones en los plazos inicialmente planificados. Las tareas relacionadas con las pruebas de viabilidad se completaron con anticipación, mientras que otras actividades sufrieron retrasos debido a incidencias técnicas no previstas.

Ajustes:

EDT	Nombre de Tarea	Duración (días)	Comienzo	Fin
1	Reuniones mensuales	4	15/03/2025	28/05/2025
2	Documentación e Investigación	20	17/03/2025	11/04/2025
3	Definición y diseño del prototipo	15	14/04/2025	30/04/2025
4	Desarrollo del prototipo	28	01/05/2025	03/06/2025
5	Desarrollo de los contenidos del Proyecto	18	04/06/2025	25/06/2025
6	Desarrollo de la Memoria del Proyecto	22	26/06/2025	22/07/2025

Tabla 3.13 Tabla de planificación de tareas resumida del proyecto FINAL

EDT	Tarea	Duración	Comienzo	Fin	Predecesoras
1	Reuniones Mensuales	4 días	15/03/2025	26/05/2025	-
1.1	Reuniones Mensuales 1	2 horas	15/03/2025	15/03/2025	-
1.2	Reuniones Mensuales 2	2 horas	15/04/2025	15/04/2025	-
1.3	Reuniones Mensuales 3	2 horas	12/05/2025	12/05/2025	-
1.4	Reuniones Mensuales 4	2 horas	26/05/2025	26/05/2025	-
2	Documentación e Investigación	20 días	17/03/2025	11/04/2025	-
2.1	Revisión de requisitos del TFM	2 días	17/03/2025	18/03/2025	-
2.2	Investigación sobre hardening en Linux y Ubuntu Pro	3 días	19/03/2025	21/03/2025	2.1
2.3	Investigación sobre Vagrant y VirtualBox	2 días	24/03/2025	25/03/2025	2.1
2.4	Análisis de herramientas de seguridad (Fail2Ban, PortSentry)	3 días	26/03/2025	28/03/2025	2.1
2.5	Investigación sobre Docker y contenedores seguros	4 días	31/03/2025	03/04/2025	2.1
2.6	Estudio de Apache y Nginx seguros con ModSecurity	3 días	04/04/2025	08/04/2025	2.1
2.7	Análisis de OWASP CRS y su aplicación práctica	2 días	09/04/2025	10/04/2025	2.6
2.8	Investigación sobre automatización de procesos de seguridad en sistemas	1 días	11/04/2025	11/04/2025	2.1
3	Definición y diseño del prototipo	15 días	14/04/2025	30/04/2025	2.1
3.1	Definición de los objetivos funcionales y de seguridad del prototipo	2 días	14/04/2025	15/04/2025	2.1
3.2	Diseño de la arquitectura general del sistema Aragog	3 días	16/04/2025	18/04/2025	3.1
3.3	Definición de los componentes principales (VM base, contenedores, scripts)	3 días	21/04/2025	23/04/2025	3.2
3.4	Especificación de flujos de uso para administradores y alumnos	2 días	24/04/2025	25/04/2025	3.2
3.5	Diseño del entorno multiusuario y política de aislamiento	2 días	26/04/2025	27/04/2025	3.3
3.6	Diseño de la estructura de scripts automatizados	2 días	28/04/2025	29/04/2025	3.3
3.7	Validación del diseño con el consultor técnico o tutor	1 días	30/04/2025	30/04/2025	3.1 a 3.6
4	Desarrollo del prototipo	28 días	01/05/2025	03/06/2025	3.7
4.1	Desarrollo del script de provisión automatizada (provision.sh)	5 días	01/05/2025	07/05/2025	3.7
4.2	Configuración segura de la máquina virtual base (Ubuntu Pro + hardening)	4 días	06/05/2025	09/05/2025	3.7, 4.1
4.3	Integración y configuración de herramientas de seguridad (Fail2Ban, PortSentry)	3 días	10/05/2025	12/05/2025	4.2
4.4	Creación del script de despliegue para alumnos (crear_alumno.sh)	4 días	13/05/2025	16/05/2025	4.1
4.5	Desarrollo del contenedor Docker seguro para Apache	3 días	13/05/2025	15/05/2025	4.1
4.6	Desarrollo del contenedor Docker seguro para Nginx	3 días	17/05/2025	19/05/2025	4.5
4.7	Implementación del script de reparación de permisos (reparar_permisos.sh)	2 días	20/05/2025	21/05/2025	4.4
4.8	Automatización del mapeo de volúmenes y puertos para contenedores	3 días	22/05/2025	24/05/2025	4.6, 4.7
4.9	Pruebas de funcionamiento y corrección de errores del sistema Aragog	6 días	25/05/2025	30/05/2025	4.1-4.8
5	Desarrollo de los contenidos del Proyecto	18 días	04/06/2025	25/06/2025	4.9
5.1	Elaboración de descripciones técnicas de cada	3 días	04/06/2025	06/06/2025	4.9

	componente del sistema				
5.2	Desarrollo de diagramas de arquitectura y flujo de trabajo	3 días	09/06/2025	11/06/2025	5.1
5.3	Redacción de instrucciones para el uso del sistema por parte de alumnos	2 días	12/06/2025	13/06/2025	5.1
5.4	Preparación de documentación para administradores y responsables técnicos	3 días	16/06/2025	18/06/2025	5.2
5.5	Generación y organización del contenido gráfico (capturas, esquemas, etc.)	3 días	19/06/2025	23/06/2025	5.2, 5.4
5.6	Desarrollo de la tabla de riesgos y planificación de respuesta	2 días	24/06/2025	25/06/2025	4.9
5.7	Elaboración del WBS y planificación detallada de tareas del proyecto	2 días	24/06/2025	25/06/2025	5.1-5.6
6	Desarrollo de la Memoria del Proyecto	22 días	26/06/2025	22/07/2025	5.7
6.1	Introducción y Fijación de objetivos	3 días	26/06/2025	30/06/2025	5.7
6.2	Planificación y Gestión del TFM	2 días	01/07/2025	02/07/2025	5.7
6.3	Estado Actual de los Conocimientos Científico-Técnicos	4 días	03/07/2025	08/07/2025	6.1
6.4	Descripción del Sistema	4 días	09/07/2025	14/07/2025	6.2, 6.3
6.5	Metodología del trabajo	3 días	15/07/2025	17/07/2025	6.4
6.6	Resultados Obtenidos	3 días	18/07/2025	22/07/2025	6.5
6.7	Conclusiones y trabajo futuro	2 días	21/07/2025	22/07/2025	6.6
6.8	Realización de la memoria del proyecto	1 días	22/07/2025	22/07/2025	6.1-6.7

Tabla 3.14 Planificación Final Actualizada

3.3.2 Informe final de riesgos.

Los riesgos más probables fueron mitigados de forma efectiva a lo largo del desarrollo, mientras que aquellos con mayor impacto potencial sobre el alcance o la calidad del proyecto no llegaron a materializarse. Las medidas anticipadas, como la planificación con márgenes de seguridad y la automatización de tareas clave, permitieron mantener el control sobre los posibles desvíos. Algunas incidencias menores surgidas durante la implementación principalmente relacionadas con compatibilidad o ajustes técnicos fueron resueltas con agilidad, sin afectar significativamente la planificación general. En conjunto, el impacto de los riesgos sobre el cronograma y los recursos fue mínimo.

3.3.3 Presupuesto final de costes.

A pesar de que durante el desarrollo del proyecto ARAGOG – Seguridad Automatizada se produjeron ciertos desfases en el calendario previsto, los costes finales se mantuvieron alineados con el presupuesto inicial. Esta estabilidad se debe a una compensación natural que ocurrió a lo largo de las distintas etapas del proyecto.

En concreto, si bien algunas tareas experimentaron ampliaciones de plazo, otras pudieron optimizarse y ejecutarse en menos tiempo de lo previsto, gracias a una mejor planificación y al aprovechamiento eficiente de los recursos técnicos y humanos. Además, no fue necesario recurrir a contrataciones o licencias adicionales no contempladas inicialmente, lo que permitió conservar el equilibrio presupuestario.

Concepto	Ingresos	Gastos
Personal	13.300,00 €	7.420,00 €
Equipos y Licencias	0,00 €	30,00 €
Servicios	0,00 €	100,00 €
Formación	0,00 €	150,00 €
Costes financieros (1,00% de los ingresos)	0,00 €	168,00 €
Otros costes (asesoría instalación)	3.500,00 €	500,00 €
Subtotal	16.800,00 €	8.368,00 €
Beneficios	8.432,00 €	

Tabla 3.15 Costes final.

3.5.4 Informe de lecciones aprendidas

A pesar de los desafíos inherentes a cualquier proyecto, la combinación de planificación rigurosa, adaptabilidad y trabajo colaborativo permitió lograr los objetivos con éxito. Estas lecciones refuerzan la importancia de integrar metodologías ágiles con control estricto de plazos y recursos.

4. ESTADO ACTUAL DE LOS CONOCIMIENTOS CIENTÍFICO-TÉCNICOS

4.1 Seguridad en servidores web

La seguridad en servidores web es un pilar esencial en cualquier infraestructura digital, ya que estos sistemas están permanentemente expuestos a conexiones externas y constituyen un objetivo habitual de ataques. Para protegerlos de manera eficaz, es necesario aplicar un enfoque en múltiples capas que combine medidas preventivas, proactivas y reactivas.

4.1.1 Medidas preventivas

Las medidas preventivas se centran en reducir la superficie de ataque desde el inicio. Entre ellas, destaca el endurecimiento del sistema operativo, que implica desactivar servicios innecesarios, cerrar puertos no utilizados y aplicar políticas de seguridad basadas en estándares como los benchmarks del Center for Internet Security (CIS)[1]. Herramientas como Ubuntu Security Guide USG.[9] permiten automatizar este proceso y generar informes de cumplimiento.

En la configuración del servidor web Apache[5] o Nginx[6], se deben desactivar métodos HTTP innecesarios como TRACE y OPTIONS, evitar el listado de directorios y eliminar información sensible de las cabeceras de respuesta por ejemplo, la versión del servidor. Además, es fundamental limitar el acceso a recursos críticos y aplicar un control estricto de permisos a nivel de sistema de archivos.

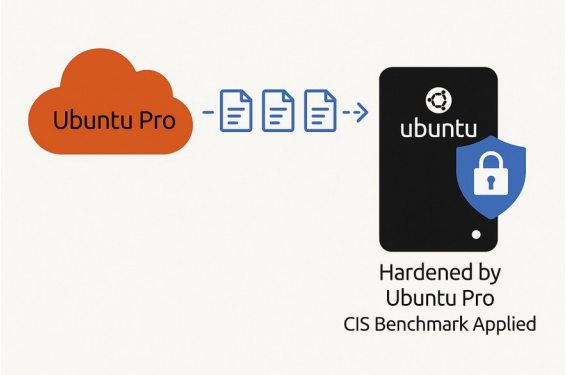


Imagen 4.1. Ubuntu. Pro hardening

4.1.2 Medidas proactivas

Las medidas proactivas buscan anticiparse a los ataques mediante tecnologías que inspeccionan y controlan el tráfico. En este contexto, el uso de un Web Application Firewall (WAF) como ModSecurity[2], junto con el conjunto de reglas de OWASP Core Rule Set (CRS) [7], permite bloquear automáticamente ataques conocidos, como inyecciones SQL o scripts maliciosos (XSS).

Complementariamente, la configuración de cabeceras HTTP seguras refuerza la protección del navegador frente a ataques del lado cliente. Algunas de las más relevantes son:

- Strict-Transport-Security
- Content-Security-Policy
- X-Frame-Options
- X-XSS-Protection

Estas cabeceras ayudan a prevenir técnicas de clickjacking, inyección de contenido y otras manipulaciones comunes.

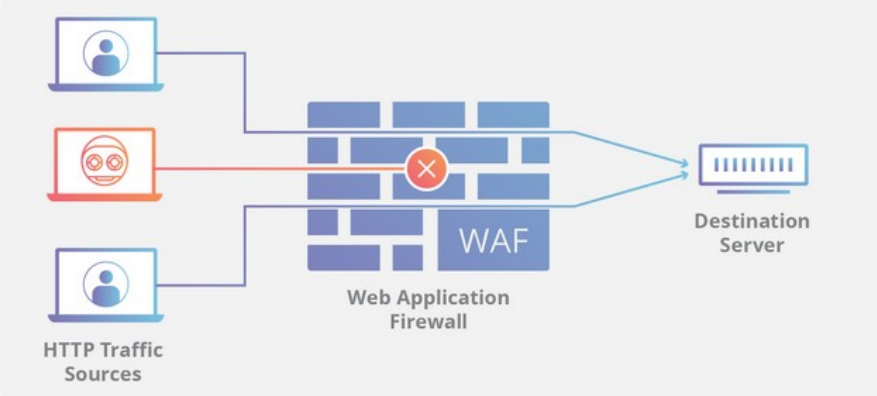


Imagen 4.2. Web Application Firewall - Fuente: Cloudflare [10].

4.1.3 Medidas reactivas

Las medidas reactivas permiten responder a actividades sospechosas o ataques en curso. Entre las herramientas más útiles están:

- Fail2Ban, que bloquea automáticamente direcciones IP tras múltiples intentos fallidos de autenticación.[3]
- PortSentry, que detecta escaneos de puertos y puede activar respuestas automáticas.[8]

La monitorización activa del entorno es crucial para detectar amenazas en tiempo real y mantener una postura de defensa adaptable.

La protección de un servidor web no depende de una única herramienta, sino de la integración de múltiples mecanismos que actúan en distintas fases: prevención, detección y reacción. Esta estrategia en capas forma la base sobre la que se construye el proyecto Aragog, cuyo objetivo es ofrecer entornos web seguros por defecto, automatizados y alineados con buenas prácticas internacionales.

4.2 Automatización de despliegues

La automatización en el despliegue de infraestructuras y servicios ha cobrado una relevancia creciente en el ámbito de la administración de sistemas y el desarrollo de software. Tradicionalmente, la configuración manual de servidores implicaba un alto grado de intervención humana, lo que aumentaba la probabilidad de errores, inconsistencias entre entornos y una menor escalabilidad. Como respuesta a estas limitaciones, han surgido herramientas y enfoques que permiten automatizar tanto la provisión de recursos como la configuración de los servicios sobre ellos.

Entre las herramientas más utilizadas para automatizar la creación de entornos virtuales destaca Vagrant[11], que facilita la gestión de máquinas virtuales reproducibles mediante archivos de configuración declarativos (Vagrantfile). Vagrant permite definir desde el sistema operativo base hasta los recursos asignados (memoria, CPU, red), así como vincular scripts de provisión para ejecutar configuraciones automáticamente al levantar la máquina. Este enfoque garantiza que cualquier instancia generada se comporte de forma idéntica, sin importar en qué sistema anfitrión se ejecute.

Otro componente fundamental en la automatización moderna es el uso de scripts de configuración o provisión. A través de lenguajes como Bash, Python o mediante gestores de configuración, se pueden instalar servicios, ajustar parámetros del sistema, aplicar políticas de seguridad o establecer entornos personalizados. Esta capacidad resulta especialmente valiosa cuando se requiere desplegar entornos homogéneos para múltiples usuarios, como sucede en entornos académicos o de investigación.

En el contexto de servidores web, la automatización no solo permite desplegar el servicio HTTP como tal, sino también aplicar configuraciones de seguridad avanzadas, generar entornos aislados, establecer usuarios específicos, y garantizar que cada instancia cumpla con los mismos requisitos técnicos y normativos. La combinación de esta automatización con tecnologías de contenedorización, como Docker[12], potencia aún más la reproducibilidad y el aislamiento, al encapsular servicios completos en imágenes fácilmente reutilizables.

En este contexto, el proyecto Aragog se alinea con estas prácticas, estructurándose en tres etapas claramente diferenciadas: una fase inicial de provisión de una máquina virtual endurecida, y dos fases posteriores orientadas al despliegue automatizado de servidores web. Esta organización modular favorece la mantenibilidad, la escalabilidad y futuras actualizaciones del sistema automatizado.

4.3 Virtualización

La virtualización es una tecnología clave en la arquitectura moderna de sistemas informáticos. Permite ejecutar múltiples entornos aislados y funcionales sobre un único sistema físico, gracias a la abstracción del hardware mediante un hipervisor. Esta capacidad ha transformado tanto los entornos de producción como los de desarrollo, al facilitar la replicación de sistemas completos, la mejora en la eficiencia del uso de recursos y una administración más flexible y segura .[13]

Cada máquina virtual emula completamente un sistema operativo sobre una capa de hipervisor como VirtualBox, ejecutándose de forma independiente y con su propio sistema de archivos, procesos y configuración. Esta separación garantiza un elevado grado de aislamiento entre entornos, lo cual es especialmente útil en contextos donde se requiere evitar interferencias o mantener integridad frente a errores o ataques.[15]

Por otro lado, la contenedorización, liderada por herramientas como Docker, representa una forma de virtualización a nivel de sistema operativo, más ligera y centrada en el aislamiento de aplicaciones. Aunque los contenedores comparten el kernel del host, ofrecen entornos encapsulados y portables que pueden desplegarse de forma instantánea con configuraciones definidas previamente. Su eficiencia en el uso de recursos y rapidez de ejecución los convierte en una solución ideal para servicios específicos como servidores web o microservicios.

Desde una perspectiva de seguridad, la virtualización aporta varias ventajas: permite crear entornos desechables y revertibles, realizar pruebas sin comprometer el sistema principal, aplicar configuraciones específicas sin afectar a otros servicios, y reforzar el aislamiento en caso de que una instancia se vea comprometida. Asimismo, facilita el cumplimiento de políticas de seguridad basadas en imágenes estándar, reduciendo la variabilidad entre entornos.

En el ámbito educativo y de investigación, la virtualización permite crear laboratorios prácticos seguros, donde cada estudiante o grupo puede operar con su propio sistema virtualizado sin riesgo de interferencia mutua.[15] Esto habilita escenarios realistas para el aprendizaje de administración de sistemas, seguridad, redes o desarrollo web.

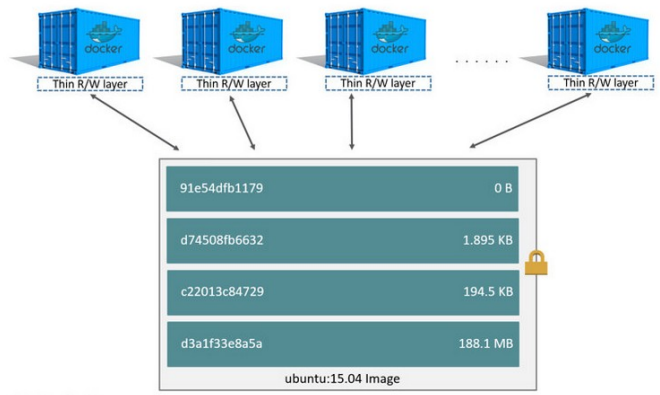


Imagen 4.3. Multiusuario Fuente: IESGN Granadilla [14]

El proyecto Aragog se apoya en estas capacidades para construir su arquitectura: utiliza máquinas virtuales como entorno base controlado y seguro, y contenedores para desplegar aplicaciones web de forma aislada y reproducible. Esta combinación proporciona robustez, flexibilidad, facilidad de mantenimiento y sobre todo Seguridad. alineándose con los principios actuales de diseño de infraestructuras TI seguras y escalables.

4.4 Gestión multiusuario en entornos educativos

Los entornos educativos, especialmente aquellos orientados a disciplinas como la ingenieria WEB, la ciberseguridad o la administración de sistemas, requieren soluciones que permitan la convivencia de múltiples usuarios en un mismo sistema sin que ello comprometa la seguridad, la estabilidad ni la privacidad de los datos. Esta necesidad se acentúa cuando los recursos son limitados y deben ser compartidos entre varios alumnos o grupos de trabajo, como sucede frecuentemente en laboratorios universitarios o plataformas de prácticas virtualizadas [15].

Gestionar múltiples usuarios en un mismo entorno conlleva varios desafíos técnicos: el aislamiento de los entornos personales de cada alumno, el control de permisos de acceso, la asignación de recursos de forma equitativa y el mantenimiento de la integridad del sistema frente a acciones accidentales o maliciosas. Además, el entorno debe ser reproducible, recuperable y fácil de administrar, incluso cuando los usuarios realizan cambios profundos en su espacio asignado.

Una práctica común para enfrentar esta situación es la creación de entornos personales preconfigurados, bien sea mediante directorios de usuario aislados en un sistema multiusuario tradicional o mediante el uso de contenedores o máquinas virtuales dedicadas. Estos enfoques permiten asignar a cada alumno un espacio seguro donde trabajar con autonomía, sin interferir en el

entorno de otros ni en el sistema global. Complementariamente, deben definirse políticas de permisos estrictas, establecer límites mediante umask, y aplicar scripts de verificación o restauración periódica de los entornos.

También es fundamental restringir ciertos accesos (por ejemplo, al sistema base o a servicios críticos) y garantizar que los usuarios no puedan elevar privilegios indebidamente. Herramientas como sudo, el control granular de grupos de usuarios y configuraciones del sistema base.

En el contexto académico, estas medidas deben equilibrar seguridad con usabilidad: los alumnos deben contar con libertad suficiente para realizar sus prácticas, pero en un entorno que minimice el riesgo de comprometer la infraestructura común. Asimismo, la gestión debe facilitar la incorporación, supervisión y eliminación de usuarios de forma ágil, especialmente en cursos con alta rotación.

El proyecto Aragog se ajusta a estas necesidades al proporcionar una arquitectura modular preparada para gestionar múltiples usuarios de forma controlada, garantizando el aislamiento, la trazabilidad y la integridad del entorno. Este enfoque facilita su integración en escenarios educativos reales, donde la seguridad multiusuario es una prioridad operativa.

4.5 Justificación de la integración tecnológica

Las tecnologías analizadas virtualización, automatización de despliegues, contenedores Docker[12], Fail2Ban[3], ModSecurity y el hardening con CIS/USG[9] no son novedosas en sí mismas, pero su valor estratégico emerge al integrarse con un propósito común: garantizar entornos seguros desde su origen, minimizando errores humanos y asegurando cumplimiento normativo.

El proyecto Aragog no busca innovar por sustitución, sino por combinación eficiente. Cada herramienta cumple un rol específico dentro de una arquitectura modular y automatizada que cubre todas las capas del sistema: desde el sistema operativo hasta la capa de aplicación web. Esta integración coherente maximiza la seguridad, la trazabilidad y la escalabilidad.

En entornos educativos, este enfoque cobra especial relevancia: permite a los alumnos interactuar con tecnologías reales dentro de espacios controlados, reproducibles y seguros, donde la configuración por defecto ya incorpora buenas prácticas de seguridad.

4.6 Resumen de herramientas de seguridad utilizadas en Aragog

Herramienta	Función principal	Tipo de medida
Ubuntu Security Guide (USG)	Aplicación de hardening basado en CIS Level 1	Preventiva
Docker	Aislamiento y despliegue modular de servicios	Preventiva
ModSecurity + OWASP CRS	Inspección profunda y bloqueo de tráfico malicioso	Proactiva
Cabeceras HTTP seguras	Políticas de navegador seguras (CSP, HSTS, etc.)	Proactiva
Fail2Ban	Bloqueo automático ante intentos fallidos repetidos	Reactiva
PortSentry	Detección de escaneos de puertos	Reactiva

Tabla 4.1. Herramientas Aragog

5. DESCRIPCIÓN DEL SISTEMA.

5.1 Propuesta general del sistema

El sistema propuesto Aragog, consiste en el diseño y desarrollo de una infraestructura virtualizada capaz de desplegar entornos web seguros de forma automatizada, reproducible y alineada con estándares internacionales de seguridad.

Esta propuesta está orientada principalmente a entornos educativos y técnicos, como los laboratorios de prácticas de la Universidad de Oviedo, donde se requiere una plataforma segura, práctica y fácilmente administrable para alojar servicios web en contextos multiusuario.

5.1.1 Máquina virtual base (Ubuntu Pro + CIS Level 1)

Esta máquina sirve como base segura para los demás componentes. Está preconfigurada con las siguientes medidas de seguridad:

- Sistema operativo: Ubuntu Pro[4], con soporte ampliado y parches automáticos de seguridad.
- Perfil de hardening: aplicación de las directrices del CIS Benchmark Level 1[1] para servidores Linux.
- Instalación y configuración de Fail2Ban[3] para prevenir ataques de fuerza bruta.
- Uso de PortSentry[8] para detectar y bloquear escaneos de puertos.
- Desactivación de servicios y puertos innecesarios.
- Configuración de UFW (Uncomplicated Firewall)[16] con políticas restrictivas por defecto.
- Autenticación mediante clave pública/privada (SSH)[17], con desactivación de acceso por contraseña.
- Refuerzo de logs y auditoría mediante auditd.

5.1.2 Contenedor web Apache seguro

Este contenedor se despliega sobre la máquina base y hereda todas sus medidas de seguridad, además de incluir:

- Configuración reforzada de Apache (mod_headers, mod_ssl, mod_rewrite).[5]

- Activación de políticas de seguridad HTTP (Content Security Policy, X-Frame-Options, etc.).
- Uso de HTTPS con certificados generados automáticamente (por ejemplo, con Let's Encrypt).[18]
- Integración de ModSecurity[2] como Web Application Firewall (WAF) con reglas OWASP CRS.[7]
- Desactivación de módulos innecesarios y listado de directorios.
- Aislamiento mediante contenedor con control de recursos (CPU, memoria).

5.1.3 Contenedor web Nginx seguro

También desplegable sobre la máquina base y con las medidas de esta heredadas, el contenedor de Nginx incluye:

- Configuración segura de Nginx con refuerzo de headers HTTP.
- Redirección forzada a HTTPS.
- Soporte para HTTP/2.
- Integración con ModSecurity (en modo proxy inverso).
- Configuración avanzada de seguridad para sitios estáticos o como proxy de aplicaciones backend.
- Monitorización básica mediante nginx-status y control de acceso a esta información.[6]

5.2 Arquitectura del sistema

La arquitectura del sistema *Aragog* está diseñada para ofrecer una solución modular, segura y automatizada, compuesta por una máquina virtual base y un conjunto de scripts y contenedores que permiten escalar funcionalidades de manera controlada. Esta estructura permite mantener una separación lógica entre los distintos componentes del sistema, facilitando su gestión, auditoría y mantenimiento. (Ver imagen 5.1)

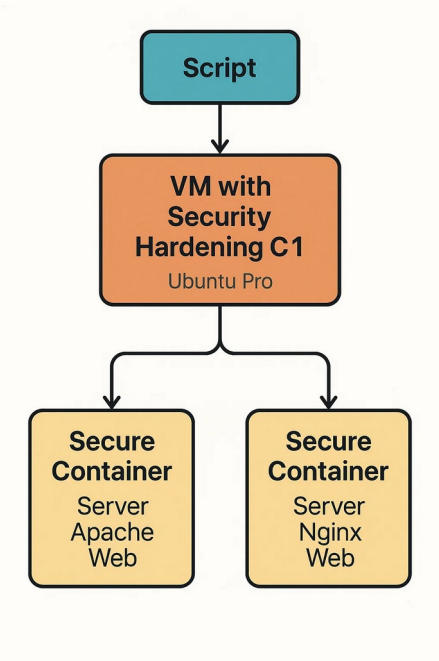


Imagen 5.1. Diagrama de los tres modulos principales.

5.3 Componentes principales

5.3.1 Maquina Virtual base

La infraestructura de *Aragog* se construye sobre una máquina virtual creada mediante Vagrant y ejecutada con VirtualBox, utilizando como sistema operativo base Ubuntu 22.04 LTS con suscripción activa a Ubuntu Pro. En la fase de provisión inicial, se aplica un proceso de endurecimiento integral utilizando el perfil de seguridad CIS Level 1 Server Benchmark, proporcionado por la herramienta Ubuntu Security Guide (USG)[9].

Este proceso configura el sistema conforme a más de 240 reglas de seguridad propuestas por la comunidad internacional de ciberseguridad y recogidas en el estándar SCAP (Security Content Automation Protocol)[19]. Además, se instalan mecanismos de protección activa como Fail2Ban, encargado de mitigar ataques de fuerza bruta sobre servicios como SSH, y PortSentry[8], que responde ante escaneos de puertos sospechosos. Ambos servicios están configurados mediante ficheros locales de reglas (jail.local, portsentry.conf), lo que permite su actualización o ajuste sin necesidad de reinstalación, facilitando una gestión segura, modular y escalable. Esta estructura responde a un principio ampliamente aceptado en desarrollo de software: la separación entre lógica y configuración, que mejora la mantenibilidad y permite adaptar el comportamiento del sistema de forma flexible y controlada.

Una característica destacable de esta arquitectura es la generación automática de un informe de auditoría. Se ejecuta una auditoría completa mediante usg audit y se genera un informe en formato HTML.*(ver Imagen 5.2)* Este informe contiene un desglose detallado del cumplimiento del sistema frente al perfil CIS, incluyendo identificadores SCAP, estado de cada regla, severidad, y puntuación total.

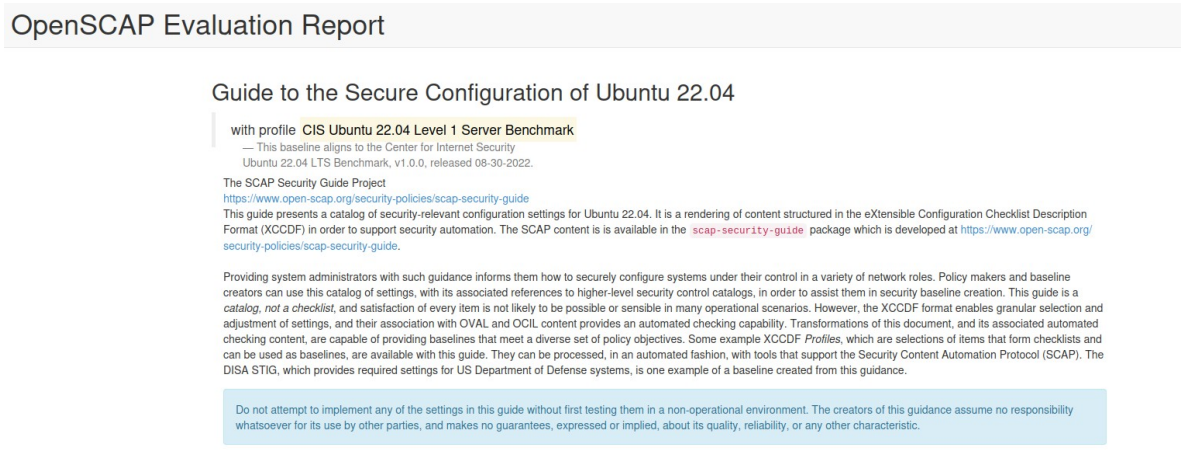
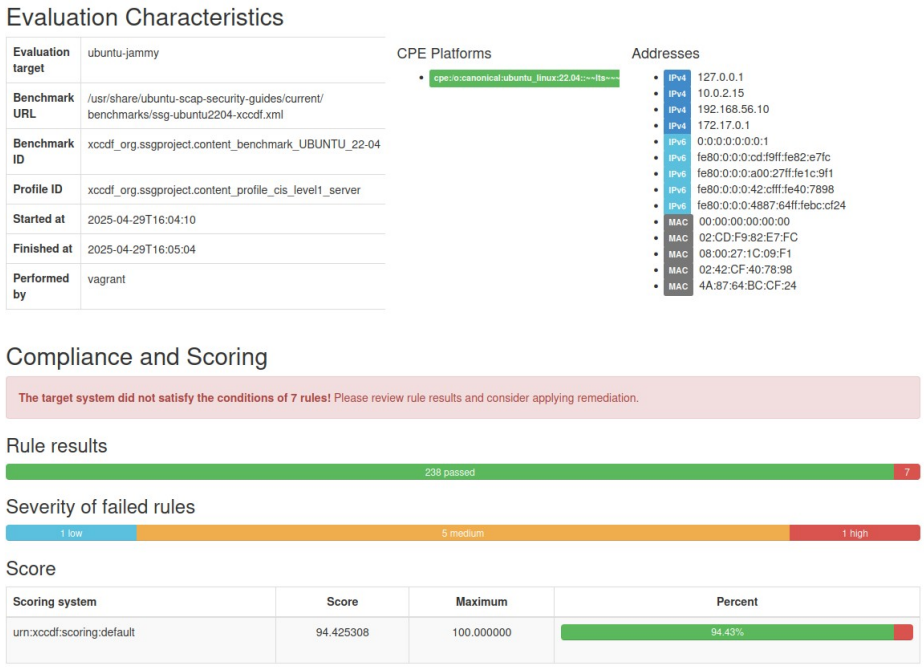


Imagen 5.2. Vista principal del informe HTML generado por USG.

En pruebas realizadas, el sistema alcanzó un 94.43% de cumplimiento según los estándares definidos para Ubuntu 22.04. *(ver Imagen 5.3)* Las pocas reglas no satisfechas corresponden a configuraciones que, por motivos de seguridad práctica, no deben automatizarse, como políticas de contraseñas, intervalos de bloqueo o validación de accesos individuales, ya que requieren intervención del administrador o del usuario final. Esta decisión consciente permite evitar vulnerabilidades asociadas a credenciales genéricas o configuraciones forzadas sin contexto, *(ver Imágenes 5.4 y 5.5)*



Evaluation Characteristics

Evaluation target	ubuntu-jammy
Benchmark URL	/usr/share/ubuntu-scap-security-guides/current/benchmarks/ssg-ubuntu2204-xccdf.xml
Benchmark ID	xccdf_org.ssgproject.content_benchmark_UBUNTU_22-04
Profile ID	xccdf_org.ssgproject.content_profile_cis_level1_server
Started at	2025-04-29T16:04:10
Finished at	2025-04-29T16:05:04
Performed by	vagrant

CPE Platforms

- cpe:/o:canonical:ubuntu_linux:22.04::~~lts~~~

Addresses

- IPv4 127.0.0.1
- IPv4 10.0.2.15
- IPv4 192.168.56.10
- IPv4 172.17.0.1
- IPv6 0:0:0:0:0:0:0:1
- IPv6 fe80:0:0:0:cd:f9ff:fe82:e7fc
- IPv6 fe80:0:0:0:a00:27ff:fe1c:9f1
- IPv6 fe80:0:0:0:42:cfff:fe40:7898
- IPv6 fe80:0:0:0:4887:64ff:febc:cf24
- MAC 00:00:00:00:00:00
- MAC 02:CD:F9:82:E7:FC
- MAC 08:00:27:1C:09:F1
- MAC 02:42:CF:40:78:98
- MAC 4A:87:64:BC:CF:24

Compliance and Scoring

The target system did not satisfy the conditions of 7 rules! Please review rule results and consider applying remediation.

Rule results

238 passed | 7

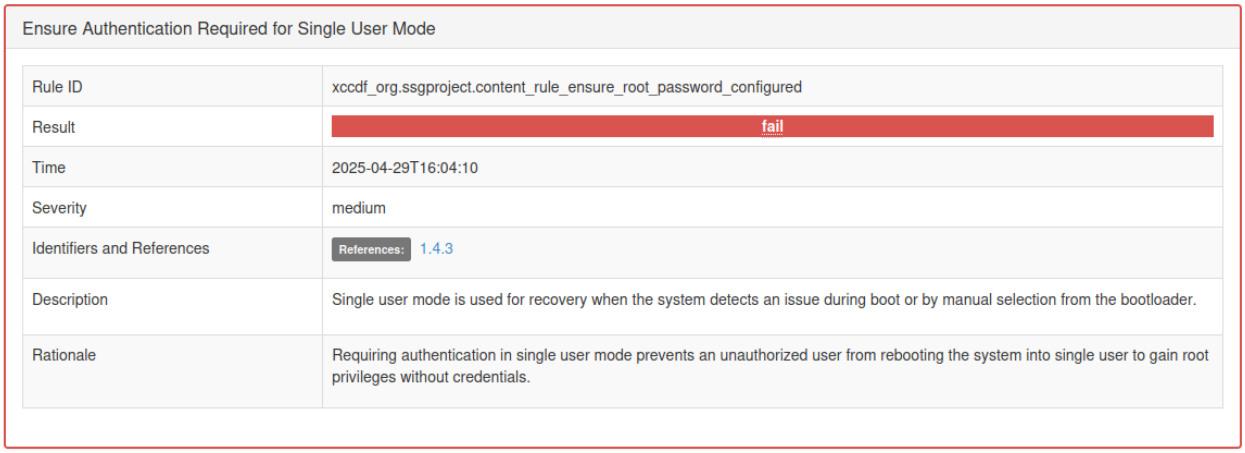
Severity of failed rules

1 low | 5 medium | 1 high

Score

Scoring system	Score	Maximum	Percent
urn:xccdf:scoring:default	94.425308	100.000000	94.43%

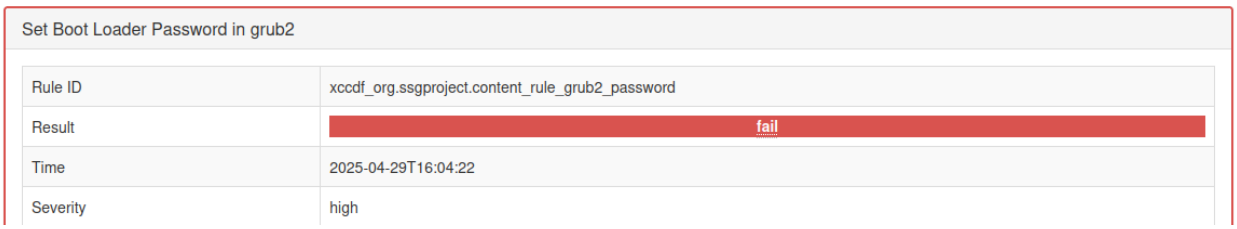
Imagen 5.3 Resultado general de cumplimiento de seguridad (94.43%).



Ensure Authentication Required for Single User Mode

Rule ID	xccdf_org.ssgproject.content_rule_ensure_root_password_configured
Result	fail
Time	2025-04-29T16:04:10
Severity	medium
Identifiers and References	References: 1.4.3
Description	Single user mode is used for recovery when the system detects an issue during boot or by manual selection from the bootloader.
Rationale	Requiring authentication in single user mode prevents an unauthorized user from rebooting the system into single user to gain root privileges without credentials.

Imagen 5.4. Ejemplo de regla fallida relacionada con configuración de contraseñas.



Set Boot Loader Password in grub2

Rule ID	xccdf_org.ssgproject.content_rule_grub2_password
Result	fail
Time	2025-04-29T16:04:22
Severity	high

Imagen 5.5. Otra regla fallida de carácter manual.

Toda la instalación y configuración del sistema base se encuentra contenida en un único script (provision.sh), que es ejecutado automáticamente por Vagrant en el primer arranque de la máquina virtual. Este script garantiza que el sistema quede configurado de forma idéntica y conforme a las políticas de seguridad definidas, evitando errores manuales y asegurando un entorno reproducible. Incluye la activación de Ubuntu Pro, la instalación de paquetes necesarios, la configuración de usuarios, y la integración de servicios de seguridad y scripts administrativos.

5.3.2 Servidores WEB

Uno de los componentes clave del sistema *Aragog* es la base segurida bajo hardening estricto y su subsistema de servidores web desplegables mediante contenedores Docker. Este componente está diseñado para proporcionar entornos seguros, aislados y consistentes para el alojamiento de sitios web individuales por usuario. Cada servidor web es una instancia encapsulada que incluye no solo el servicio de publicación HTTP, sino también una serie de mecanismos de protección integrados que garantizan su robustez desde el primer despliegue.

El sistema dispone de dos alternativas principales, ambas disponibles como contenedores configurables.

5.3.2.1 Servidor web Apache en Docker

La imagen de Apache está basada en Ubuntu 22.04 e incluye el servidor Apache 2.4 junto con el módulo ModSecurity[2] activado. Este módulo actúa como un firewall de aplicaciones web (WAF) embebido, capaz de analizar el tráfico HTTP y aplicar reglas de seguridad en tiempo real.

La imagen también incorpora:

- El conjunto de reglas OWASP Core Rule Set (CRS)[7], ampliamente aceptado como estándar para la detección y mitigación de amenazas comunes como inyecciones, XSS, fugas de información o manipulación de cabeceras.
- Una configuración de cabeceras HTTP seguras (mediante security.conf) para proteger contra ataques del lado cliente, controlar el almacenamiento en caché y restringir métodos peligrosos.
- Eliminación del listado de directorios, ocultación de información sensible del servidor y desactivación de métodos inseguros como TRACE.
- Configuración del motor de ModSecurity en modo activo (On), no solo de detección.

Todo esto está contenido en una única imagen Docker reutilizable y fácilmente replicable para múltiples usuarios.

5.3.2.2 Servidor web Nginx en Docker

La alternativa basada en Nginx se apoya en la imagen oficial de OWASP ModSecurity v3 para Nginx, y está diseñada para ofrecer un rendimiento elevado sin comprometer la seguridad.[21]

Esta imagen contiene:

- El servidor Nginx con ModSecurity v3 activado en modo inline.
- El mismo conjunto de reglas OWASP CRS, adaptado al motor ModSecurity v3.
- Cabeceras HTTP estrictas aplicadas directamente desde los bloques de configuración del servidor virtual (como Content-Security-Policy, Strict-Transport-Security, etc.).
- Eliminación de la configuración por defecto de Nginx, reemplazada por una plantilla segura que evita fugas de información y errores de configuración comunes.[6]

El contenedor se entrega completamente preconfigurado, con la capacidad de aceptar un volumen externo para montar el contenido web del usuario sin exponer la estructura interna del contenedor.

5.3.3Ventajas del diseño contenerizado

El uso de contenedores para alojar los servidores web dentro de *Aragog* responde a una serie de principios arquitectónicos y operativos clave:

- Aislamiento lógico y de ejecución: cada contenedor se ejecuta de forma independiente, evitando que un fallo afecte a otros entornos o al sistema anfitrión.
- Separación de responsabilidades: la lógica del servidor web está encapsulada dentro del contenedor, mientras que el contenido web pertenece exclusivamente al volumen del usuario.
- Escalabilidad horizontal: es posible desplegar múltiples servidores web simultáneamente, cada uno con configuración y puerto propios.
- Modularidad y mantenibilidad: todas las reglas de seguridad y configuraciones están separadas del núcleo del sistema, lo que permite actualizaciones o ajustes sin afectar al funcionamiento general.

Este diseño convierte a los servidores web de *Aragog* en componentes reutilizables, seguros y consistentes, preparados para integrarse sin fricción en contextos educativos o entornos de pruebas controladas.

5.4 Controles y Medidas de seguridad.

La seguridad es el eje central del sistema *Aragog*, y se ha abordado aplicando principios reconocidos de defensa en profundidad, segmentación funcional y prevención activa. A continuación, se describen las tecnologías y enfoques utilizados para proteger el sistema en sus distintas capas, explicando su naturaleza, propósito y campo de aplicación.

5.4.1 Hardening del sistema operativo (Ubuntu Pro + CIS Benchmark)

El término *hardening* hace referencia al proceso de asegurar un sistema eliminando configuraciones innecesarias, cerrando vectores de ataque y aplicando políticas restrictivas. En este caso, el sistema operativo base —Ubuntu 22.04— ha sido fortalecido mediante la herramienta Ubuntu Security Guide (USG), una utilidad oficial integrada con las suscripciones de Ubuntu Pro.

USG permite aplicar perfiles de seguridad predefinidos desarrollados por organizaciones como el Center for Internet Security (CIS)[4], una entidad sin ánimo de lucro reconocida por establecer estándares de configuración segura para sistemas y software. En *Aragog*, se ha aplicado el perfil CIS Ubuntu 22.04 Level 1 Server Benchmark, orientado a servidores estándar que requieren protección sin comprometer su operatividad. Este perfil incluye más de 240 reglas que cubren:

- Seguridad de cuentas y contraseñas.
- Configuración de servicios del sistema.
- Auditoría de accesos.
- Control de permisos de archivos.
- Seguridad en la red.

CIS también define otros niveles de seguridad, como el Level 1 Workstation, más orientado a entornos de escritorio, y el Level 2 Server,[9] que introduce reglas más estrictas para sistemas críticos, aunque a menudo con un mayor impacto en la usabilidad. La posibilidad de elegir entre estos perfiles permite adaptar la política de seguridad según el propósito del sistema.

5.4.2 Fail2Ban: defensa contra ataques de fuerza bruta

Fail2Ban es una herramienta de prevención de intrusiones que protege servicios expuestos (como SSH, HTTP, FTP, etc.) frente a ataques de fuerza bruta. Su funcionamiento se basa en el monitoreo de los archivos de registro del sistema (logs): al detectar un número elevado de intentos

fallidos de autenticación desde una misma dirección IP, bloquea automáticamente dicha IP durante un periodo de tiempo configurable, mediante reglas del cortafuegos local.[3]

Es una solución ampliamente utilizada por su bajo consumo de recursos, su flexibilidad en la configuración de "jails" (entornos de vigilancia por servicio) y su capacidad de responder en tiempo real. Se trata de una defensa reactiva pero automatizada que reduce la ventana de oportunidad para atacantes persistentes.

En el siguiente fragmento del archivo jail.local de Fail2Ban,*(ver Imagen 5.6)* donde se define la política para el servicio SSH: tras 3 intentos fallidos de autenticación en un intervalo de 10 minutos, la IP ofensora será bloqueada durante 1 hora. El sistema también permite notificar por correo al administrador y genera los registros correspondientes en /var/log/auth.log

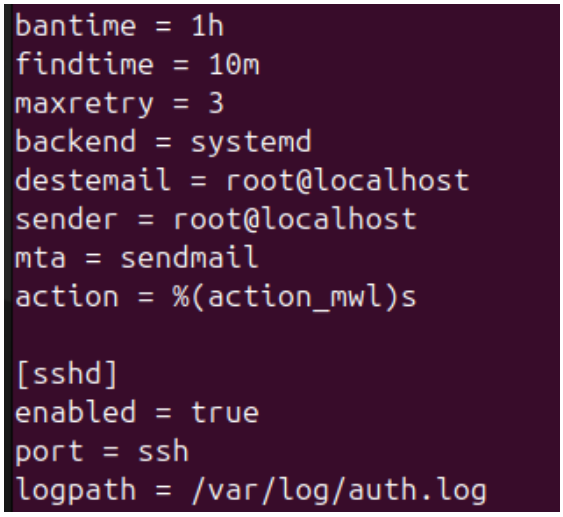


Imagen 5.6 Reglas Fail2ban

5.4.3 PortSentry: detección de escaneos de red

PortSentry es una herramienta diseñada para detectar y bloquear escaneos de puertos, una técnica común en la fase de reconocimiento previa a un ataque.[8] Los atacantes suelen utilizar herramientas como Nmap para descubrir qué servicios están activos en un host determinado. PortSentry detecta patrones de escaneo sospechosos (tanto en TCP como en UDP) y, si se produce una coincidencia, puede bloquear inmediatamente la dirección IP atacante modificando las reglas del firewall (iptables) o los archivos de control de acceso (hosts.deny).

Esto permite responder de forma preventiva incluso antes de que se intente explotar una vulnerabilidad, y convierte al sistema en un objetivo menos atractivo para el atacante, ya que no permite el reconocimiento pasivo de sus servicios.

En el siguiente fragmento del archivo portsentry.conf,*(ver Imagen 5.7)* donde se configuran los modos de monitoreo TCP y UDP, las rutas de bloqueo por firewall (iptables), y los rangos de puertos a vigilar tanto en modo simple como en modo sigiloso (*stealth*). La configuración excluye puertos de uso habitual como 22 (SSH), 80/443 (HTTP/HTTPS) para evitar falsos positivos. Además, se desactiva la resolución DNS para prevenir errores de análisis. Este archivo permite adaptar fácilmente el comportamiento de PortSentry sin modificar el binario del servicio, manteniendo una arquitectura flexible y segura.

```
# ============= Configuración Básica =============
TCP_MODE="atcp"
UDP_MODE="audp"

# ============= Configuración de Bloqueo =============
BLOCK_UDP="1"
BLOCK_TCP="1"
KILL_ROUTE="/sbin/iptables -I INPUT -s $TARGET$ -j DROP"
KILL_HOSTS_DENY="ALL: $TARGET$"

# ============= Archivos de Configuración =============
IGNORE_FILE="/etc/portsentry/portsentry.ignore"
HISTORY_FILE="/var/lib/portsentry/portsentry.history"
BLOCKED_FILE="/etc/portsentry/portsentry.blocked"

# ============= Puertos =============
# Básicos (modo simple)
TCP_PORTS="1-1024"
UDP_PORTS="1-1024"

# Avanzados (modo stealth)
ADVANCED_PORTS_TCP="1024"
ADVANCED_PORTS_UDP="1024"
ADVANCED_EXCLUDE_TCP="22,80,443,8080-8095"
ADVANCED_EXCLUDE_UDP="53"

# ============= Opciones Adicionales =============
SCAN_TRIGGER="0"
USE_RESOLV="0"  # Desactiva resolución DNS para evitar el error "Could not resolve"
```

Imagen 5.7. Reglas portsentry

5.4.4 ModSecurity y OWASP CRS: protección en la capa de aplicación

Uno de los pilares fundamentales de la seguridad en la capa de aplicación es el cumplimiento de las recomendaciones establecidas por la iniciativa OWASP (*Open Worldwide Application Security Project*)[7], una comunidad internacional enfocada en mejorar la seguridad del software. Esta organización elabora y mantiene múltiples recursos, siendo el más conocido el OWASP Top 10, un listado que agrupa las principales categorías de riesgos de seguridad en aplicaciones web, basado en datos empíricos, revisiones de expertos y encuestas globales.

El OWASP Top 10 de 2021 representa una evolución respecto a ediciones anteriores, no solo en nombre sino en enfoque: en lugar de centrarse en los síntomas técnicos, se reestructura para

identificar las causas raíz de los problemas de seguridad. En esta edición, se incorporan tres nuevas categorías, y se consolidan o renombran otras para reflejar con mayor precisión el panorama actual de vulnerabilidades en la industria del desarrollo web.(ver Imagen 5.8)

Imagen 5.8. Cambios introducidos en el OWASP Top 10 entre las ediciones 2017 y 2021.

A continuación, se destacan brevemente las diez categorías del OWASP Top 10 (2021), todas ellas abordadas por el sistema *Aragog* a través de reglas de detección integradas en el OWASP CRS dentro de ModSecurity[22]:

- A01:2021 - Pérdida de Control de Acceso: encabezando la lista, esta categoría agrupa fallas que permiten a usuarios no autorizados acceder a recursos restringidos. Representa el 3,81% de incidencia con más de 318.000 ocurrencias de CWEs asociadas.
- A02:2021 - Fallas Criptográficas: aborda errores en el uso, implementación o gestión de criptografía, incluyendo almacenamiento inseguro, transmisión sin cifrado y uso de algoritmos obsoletos. Anteriormente era tratada como "Exposición de Datos Sensibles".
- A03:2021 - Inyección: incluye ataques como SQL, NoSQL, OS Command Injection y otros. Agrupa también el Cross-Site Scripting (XSS), con una incidencia de hasta el 19% en algunas pruebas, lo que demuestra su persistencia como vector de ataque.
- A04:2021 - Diseño Inseguro: nueva categoría que reconoce que muchos fallos no derivan de errores en la implementación, sino de ausencia de controles desde el diseño arquitectónico. Esta categoría refuerza la importancia del enfoque de seguridad desde las primeras fases del desarrollo (*security by design*).
- A05:2021 - Configuración de Seguridad Incorrecta: destaca por su alta prevalencia (90% de las aplicaciones analizadas) e incluye configuraciones por defecto peligrosas, errores en los

permisos de archivos y servicios innecesarios activos. También absorbe a la antigua categoría XXE.

- A06:2021 - Componentes Vulnerables y Desactualizados: enfatiza el riesgo que representa el uso de librerías o dependencias con vulnerabilidades conocidas, especialmente cuando no se dispone de una política de actualización o revisión constante.
- A07:2021 - Fallas de Identificación y Autenticación: refleja debilidades en la verificación de identidad del usuario, uso de credenciales débiles o mal gestionadas, y exposición a ataques como credential stuffing o bypass de autenticación.
- A08:2021 - Fallas en la Integridad del Software y los Datos: nueva categoría centrada en las suposiciones inseguras dentro de pipelines de integración continua (CI/CD), validación de actualizaciones o integridad de los datos. Absorbe la anterior “Deserialización Insegura”.
- A09:2021 - Fallas en el Registro y Monitoreo: resalta la importancia de contar con mecanismos adecuados para detectar, alertar y responder ante incidentes. Sin una correcta visibilidad, muchas intrusiones pasan inadvertidas o se identifican demasiado tarde.
- A10:2021 - Falsificación de Solicitudes del Lado del Servidor (SSRF): aunque su incidencia estadística es baja, se incluye debido a su potencial impacto y a la prioridad que la comunidad de seguridad le otorga.

La integración del OWASP Core Rule Set (CRS)[7] dentro de *Aragog*, operando en conjunto con ModSecurity, permite mitigar activamente estas amenazas, proporcionando una cobertura actualizada y adaptativa frente a los principales riesgos de seguridad definidos por esta iniciativa.

5.4.5 Cabeceras HTTP seguras y control de políticas en el cliente

Además de las protecciones aplicadas en el servidor, *Aragog* incorpora mecanismos de defensa en la capa de presentación mediante el uso de cabeceras HTTP de seguridad. Estas cabeceras permiten definir políticas explícitas que influyen directamente en el comportamiento del navegador del usuario, reduciendo así la exposición a ataques del lado cliente, incluso cuando otras defensas hayan sido superadas.

La configuración por defecto de los servidores web desplegados en *Aragog* tanto Apache como Nginx incluye un conjunto robusto de cabeceras, alineadas con las recomendaciones del OWASP Secure Headers Project. Estas son[7]:

- Content-Security-Policy (CSP): restringe los orígenes permitidos para la carga de scripts, hojas de estilo, imágenes y otros recursos, bloqueando ataques como el Cross-Site Scripting (XSS).
- Strict-Transport-Security (HSTS): fuerza al navegador a usar exclusivamente conexiones HTTPS, protegiendo contra ataques de downgrade o interceptación.
- X-Frame-Options: impide que el sitio sea cargado dentro de un iframe en otro dominio, evitando ataques de clickjacking.
- X-XSS-Protection: activa mecanismos de detección de scripts maliciosos en navegadores compatibles.
- Referrer-Policy: limita el nivel de información de navegación (referer) enviada a sitios externos, protegiendo la privacidad del usuario.

Estas cabeceras actúan como barreras proactivas en el navegador, y son fundamentales en la defensa en profundidad, especialmente en entornos donde los usuarios finales acceden a contenidos dinámicos o cargan sus propios archivos.

Fragmento del archivo security.conf en el servidor Apache,(Ver imagen 5.9) donde se configuran cabeceras HTTP de seguridad (X-Frame-Options, X-XSS-Protection, Referrer-Policy, etc.) y políticas de control de caché. Estas directivas refuerzan la seguridad del navegador del usuario, mitigando ataques como clickjacking, sniffing de tipos MIME y recolección no autorizada de datos de navegación.

```
# Configuraciones de Seguridad para Apache

# Ocultar versión del servidor
ServerTokens Prod
ServerSignature Off

# Evitar listado de directorios
<Directory />
    Options -Indexes
    AllowOverride None
</Directory>

# Cabeceras de seguridad
Header always set X-Frame-Options "SAMEORIGIN"
Header always set X-Content-Type-Options "nosniff"
Header always set X-XSS-Protection "1; mode=block"
Header always set Referrer-Policy "no-referrer"

# Politicas de cache para privacidad
Header set Cache-Control "no-store, no-cache, must-revalidate, proxy-revalidate"
Header set Pragma "no-cache"
Header set Expires 0

# Evitar TRACE HTTP Method
TraceEnable Off
```

Imagen 5.9. Security.conf del servidor Apache.

5.4.6 Segmentación, aislamiento y control de acceso

La segmentación de usuarios, el aislamiento de recursos y la aplicación de controles de acceso granulares son pilares fundamentales en el diseño de entornos seguros[15]. *Aragog* adopta una arquitectura multiusuario sustentada en tecnologías ampliamente aceptadas para garantizar estos principios:

- Permisos y roles del sistema operativo (Linux)

  El sistema de usuarios en Linux se basa en un modelo de control de acceso discrecional (DAC), que permite asignar a cada archivo o directorio un propietario, un grupo y una política de permisos de lectura, escritura y ejecución. Este esquema garantiza que cada usuario solo pueda interactuar con los recursos explícitamente permitidos, evitando la interferencia entre entornos.

- Acceso remoto mediante SSH:

  Para la administración y el uso por parte de los alumnos, se emplea el protocolo Secure Shell (SSH), que permite el acceso remoto cifrado al sistema. SSH no solo protege la confidencialidad de las credenciales y las acciones del usuario, sino que, combinado con las restricciones del sistema de archivos, asegura que el alumno solo tenga visibilidad y control sobre su propio espacio de trabajo[17].

- Contenerización con Docker:

  Docker proporciona un entorno de aislamiento a nivel de sistema operativo, encapsulando procesos y servicios dentro de contenedores ligeros que comparten el kernel pero operan de forma independiente. Este mecanismo permite desplegar múltiples servidores web en paralelo —uno por usuario— con un alto grado de separación funcional. Cada contenedor dispone de su propia red virtual, espacio de archivos aislado y configuración de servidor personalizada, lo que impide que los usuarios o procesos interactúen entre entornos[20].

Este conjunto de tecnologías permite establecer un modelo de seguridad basado en el principio de privilegios mínimos, la contención de daños ante fallos o intrusiones, y la escalabilidad segura del sistema. Así, *Aragog* garantiza que cada alumno trabaje en un entorno controlado, sin comprometer la integridad del sistema o de otros usuarios.

5.5 Flujo general de funcionamiento

El sistema *Aragog* ha sido diseñado para facilitar el despliegue automatizado de entornos web seguros, estandarizados y multiusuario. A continuación, se describe el flujo operativo general desde la provisión inicial de la infraestructura hasta el uso final por parte de los alumnos.

5.5.1 Provisión inicial de la máquina virtual

La puesta en marcha del sistema comienza con la ejecución de Vagrant, herramienta que permite orquestar la creación de entornos reproducibles. En este caso, se utiliza para desplegar una máquina virtual basada en Ubuntu 22.04 LTS, con soporte extendido de seguridad proporcionado por Ubuntu Pro.(Ver imagen 5.10)

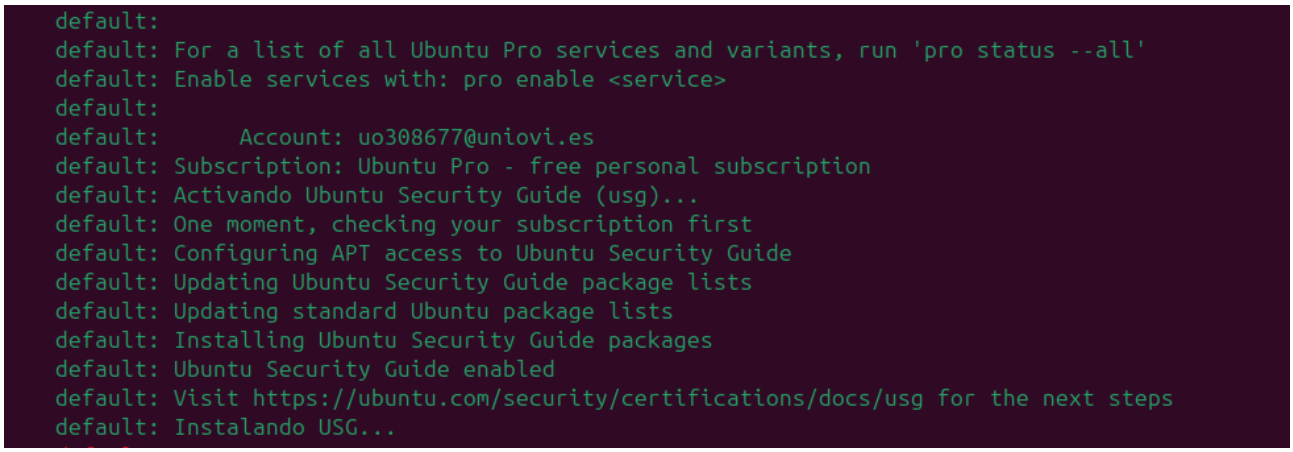


Imagen 5.10. Activando ubuntu security guide

Durante el primer arranque, Vagrant ejecuta automáticamente el script de provisión provision.sh, el cual automatiza la configuración del sistema base y garantiza que este cumpla con las políticas de seguridad predefinidas. Entre las tareas realizadas destacan:

- Actualización completa del sistema operativo y sus paquetes, asegurando que se inicie desde un estado actualizado y consistente.
- Activación del servicio Ubuntu Security Guide (USG) y aplicación del perfil de hardening CIS Level 1 Server Benchmark, alineado con los estándares del Center for Internet Security.
- Instalación de servicios de defensa activa, como Fail2Ban (protección frente a ataques de fuerza bruta) y PortSentry (detección y bloqueo de escaneos de puertos).
- Creación de un usuario administrativo (adminserver) con privilegios restringidos y perteneciente a los grupos sudo y docker.

- Restricción de accesos remotos para usuarios privilegiados, como root y el propio adminserver, mediante la configuración del servicio SSH.
- Estructuración del entorno multiusuario, incluyendo la generación del directorio base /home/WEBS_ALUMNOS, que alojará los espacios de trabajo individuales de los alumnos.

Al finalizar la provisión, el sistema genera automáticamente un informe de auditoría en formato HTML (usg_audit_report.html), que documenta el nivel de cumplimiento alcanzado frente al perfil de seguridad CIS aplicado. Este informe constituye una prueba verificable del estado de endurecimiento del sistema y permite validar que la configuración cumple con los estándares definidos desde el inicio.

5.5.2. Creación de entornos web para alumnos

Una vez provisionada la máquina virtual y asegurado el entorno base, el administrador puede generar entornos web personalizados para cada alumno mediante la ejecución del script crear_alumno.sh. Este script encapsula un proceso automatizado y controlado, que garantiza la coherencia, el aislamiento y la seguridad de cada despliegue individual. El flujo de trabajo que implementa incluye los siguientes pasos:

- Recopilación de datos del alumno, como el nombre completo y el identificador institucional (UO), que se utilizarán para generar un nombre de usuario único en el sistema.
- Selección del tipo de servidor web a desplegar: el administrador puede optar entre Apache o Nginx, según las necesidades del entorno educativo o del perfil del alumno. (Ver imagen 5.11)
- Creación de un usuario UNIX no privilegiado, con un directorio personal aislado bajo /home/WEBS_ALUMNOS. Este usuario se encuentra restringido a su espacio de trabajo y no posee capacidades administrativas sobre el sistema anfitrión.
- Generación automática de credenciales seguras, incluyendo una contraseña aleatoria codificada en base64, y asignación de los permisos estándar (propietario, grupo y umask 022) al entorno del usuario.(Ver imagen 5.12)
- Configuración del entorno operativo del alumno, incluyendo la copia del script reparar_permisos.sh, el cual permite al estudiante restaurar los permisos recomendados en su espacio de trabajo en cualquier momento.
- Validación o construcción de la imagen Docker correspondiente, seleccionando la imagen apache_seguro o nginx_seguro dependiendo de la opción elegida. Si la imagen no existe en el sistema, el script se encarga de compilarla automáticamente a partir del Dockerfile y sus configuraciones asociadas.
- Despliegue de un contenedor Docker aislado, que incluye:

- Mapeo del volumen del alumno: el directorio /home/WEBS_ALUMNOS/<usuario> se monta dentro del contenedor como raíz del servidor web (/var/www/html en Apache o /usr/share/nginx/html en Nginx).
- Asignación de un puerto TCP libre del host, expuesto externamente y vinculado al puerto 80 del contenedor, permitiendo el acceso HTTP al sitio web del alumno desde el exterior.

Este procedimiento garantiza que cada entorno web sea autónomo, reproducible y aislado, cumpliendo con las mejores prácticas en diseño seguro de sistemas multiusuario. Asimismo, el despliegue modular y contenerizado permite la gestión eficiente de múltiples instancias en paralelo, sin comprometer la seguridad ni la estabilidad del sistema anfitrión.

```
vagrant@ubuntu-jammy:~$ sudo bash /vagrant/crear_alumno.sh
Nombre completo del alumno: AGUSTIN WIDMAN AGUAYO
UO del alumno (ejemplo UO308677): UO308677
Seleccione tipo de servidor:
1) Apache
2) Nginx
Opción (1 o 2): 1
Creando carpeta /home/WEBS_ALUMNOS/UO308677A...
Creando usuario SSH UO308677A...
Configurando umask 022 para UO308677A...
Copiando script de reparación de permisos...
La imagen apache_seguro no existe. Construyéndola ahora...
```

Imagen 5.11 Ejecución del script como Administrador.

```
Successfully built fa77d62950a8
Successfully tagged apache_seguro:latest
Desplegando contenedor Docker para UO308677A en puerto 8080...
5ba4898d40be14d827941a87e5b9296e9a6fff80521b1ccd9657b7410bc91297
------------------------------------------------
Alumno creado correctamente.
Nombre completo: AGUSTIN WIDMAN AGUAYO
Usuario SSH: UO308677A
Contraseña SSH: rUfuuwU+wb6VHQIu
Servidor Web: apache
Carpeta asignada: /home/WEBS_ALUMNOS/UO308677A
Puerto asignado: 8080
------------------------------------------------
```

Imagen 5.12. Entrega de Datos para el Alumno

5.5.3. Acceso y gestión por parte del alumno

Una vez desplegado el entorno web personalizado, el alumno puede acceder de forma segura al sistema a través del protocolo SSH, utilizando las credenciales generadas específicamente para su usuario. Este acceso se restringe únicamente a su directorio personal dentro de /home/WEBS_ALUMNOS, garantizando un entorno operativo completamente aislado.

Desde su sesión, el alumno puede:

- Gestionar de forma autónoma los archivos de su sitio web, mediante la creación, edición o carga de documentos dentro de su directorio asignado.
- Ejecutar el script reparar_permisos.sh, incluido automáticamente en su carpeta de trabajo, el cual restablece los permisos recomendados del sistema: 755 para directorios y 644 para archivos. Esto asegura que los contenidos sean accesibles por el servidor web sin comprometer la privacidad del usuario. (Ver imagen 5.13)
- Visualizar su sitio en tiempo real, ya que el contenido es servido automáticamente a través del contenedor Docker previamente desplegado. Dependiendo del servidor elegido, el directorio personal del alumno es montado como volumen en la ruta raíz del servidor (/var/www/html en Apache o /usr/share/nginx/html en Nginx), permitiendo una integración transparente entre edición y visualización.(Ver imagen 5.14)

Este modelo de operación permite a cada alumno desplegar y mantener su sitio web de forma aislada, controlada y sin intervención adicional del administrador, preservando la integridad del sistema anfitrión y de los entornos de otros usuarios. La contención lógica, junto con el modelo de acceso restringido, asegura una experiencia educativa realista y segura.

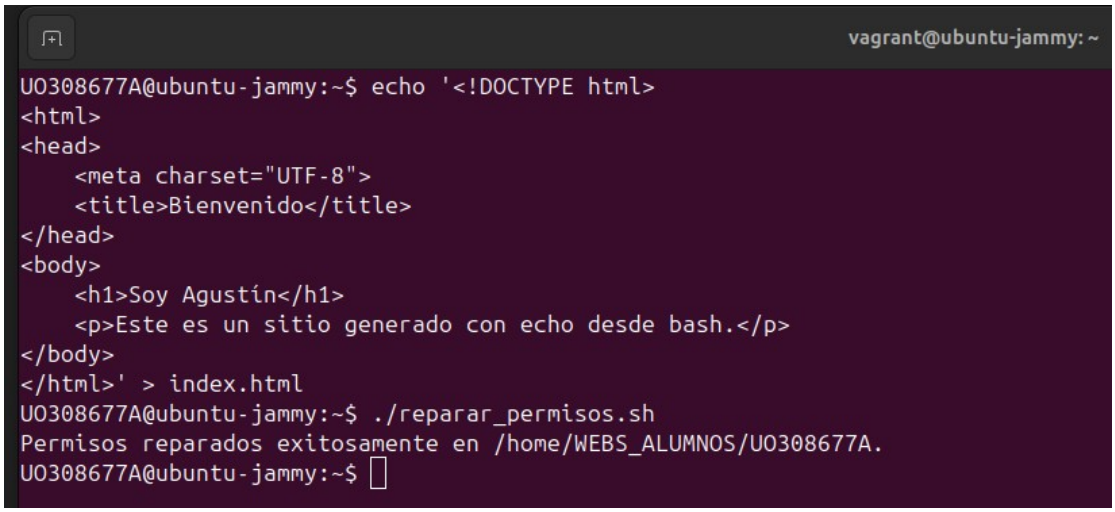


Imagen 5.13. Generando HTML y ejecución del script Permisos.

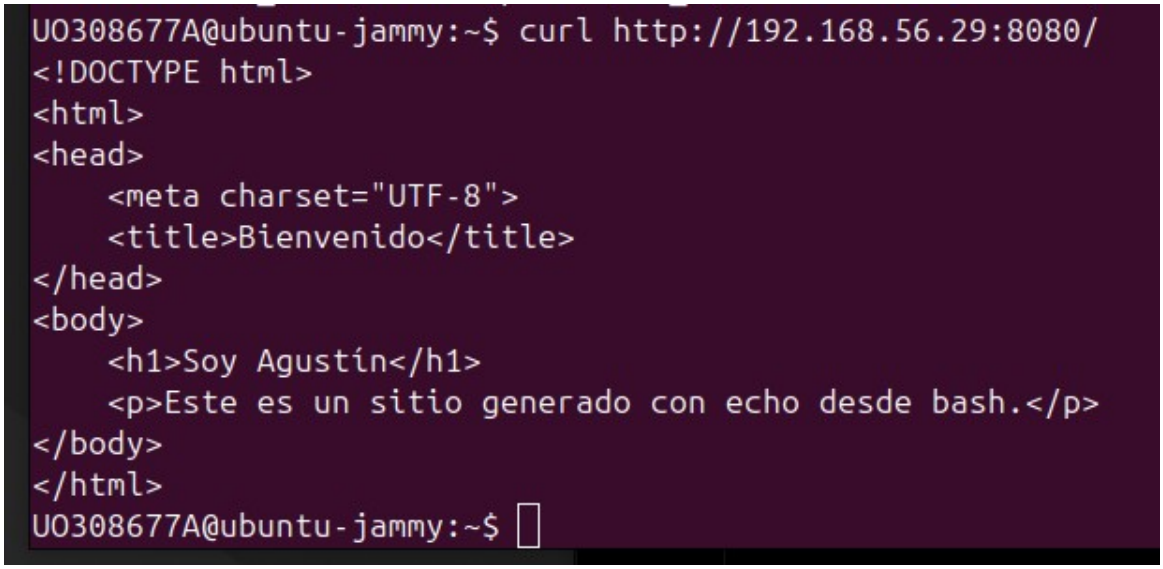


Imagen 5.14. Revisión del despliegue con CURL en puerto 8080

Diagrama general del flujo de funcionamiento del sistema *Aragog*, desde la provisión inicial de la máquina virtual hasta el acceso y operación por parte del alumno. (Ver imagen 5.15)

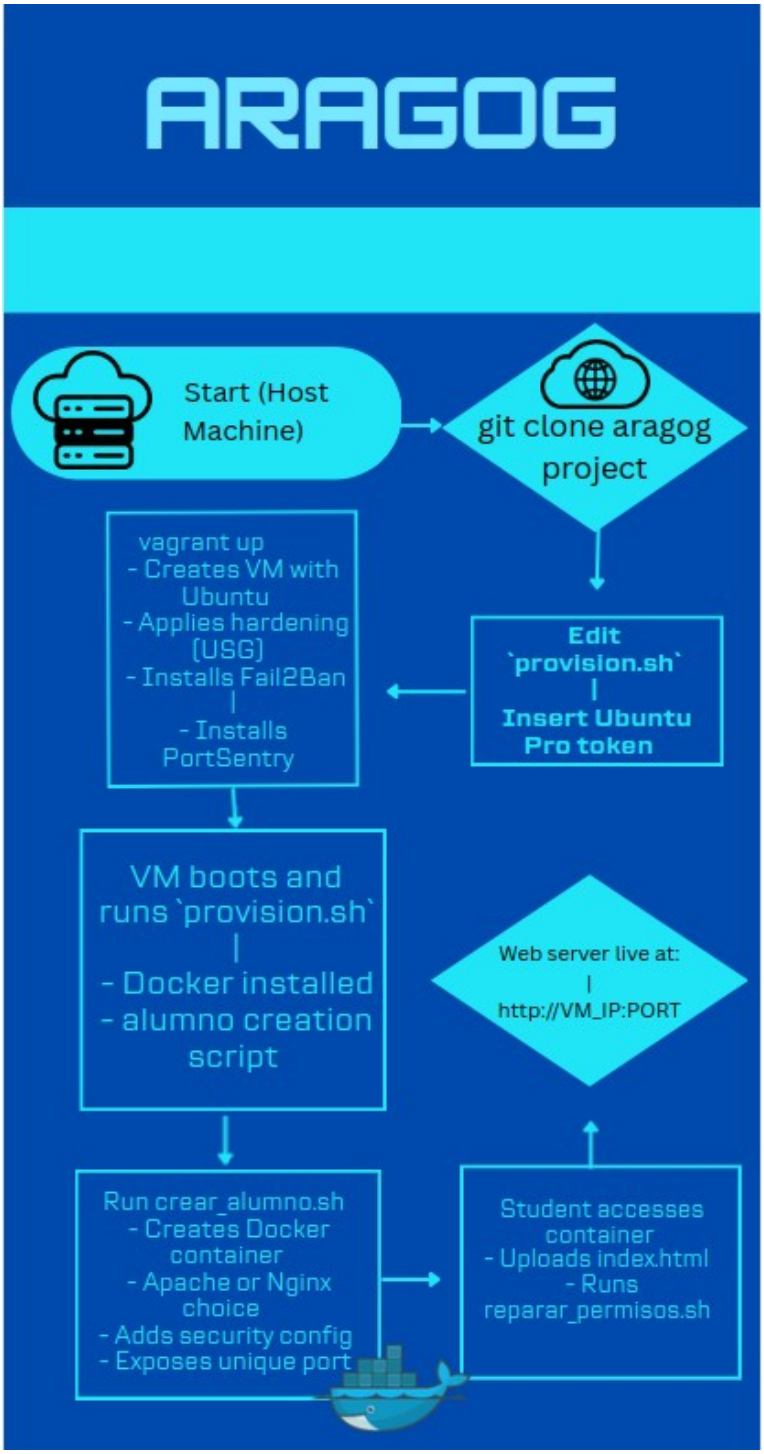


Imagen 5.15. Diagrama de Flujo Aragog

6. METODOLOGÍA DE TRABAJO

El presente capítulo describe la metodología empleada durante el desarrollo del sistema *Aragog*, detallando las fases, criterios técnicos y decisiones estratégicas que guiaron su implementación. La metodología ha sido orientada a garantizar la reproducibilidad del entorno, la seguridad desde el diseño y la escalabilidad de la solución en contextos educativos.

En coherencia con los objetivos del proyecto, se ha priorizado el uso de herramientas de automatización, la integración de estándares reconocidos de ciberseguridad, y el diseño modular del sistema. Asimismo, se ha documentado cada paso del proceso de construcción para que pueda ser replicado, validado o adaptado por otros equipos técnicos o investigadores interesados en la creación de infraestructuras educativas seguras.

6.1 Justificación de decisiones técnicas adoptadas

El desarrollo del sistema *Aragog* implicó la adopción de múltiples decisiones técnicas orientadas a cumplir los principios de seguridad, automatización, mantenibilidad y escalabilidad. Estas decisiones no fueron arbitrarias, sino que respondieron a criterios fundados tanto en las necesidades del proyecto como en las buenas prácticas recomendadas en entornos de infraestructura segura y virtualización educativa. A continuación, se detallan las principales herramientas seleccionadas y las razones técnicas que justifican su elección.

6.1.1 Ubuntu 22.04 LTS y Ubuntu Pro

Se seleccionó Ubuntu 22.04 LTS (Jammy Jellyfish) como sistema operativo base por su estabilidad, soporte a largo plazo (hasta 2032 mediante suscripción Pro) y amplia adopción en entornos profesionales, educativos y de investigación. Su ecosistema facilita la integración de herramientas de seguridad, scripting en Bash, soporte para contenedores y disponibilidad de documentación comunitaria.

La activación de Ubuntu Pro se justifica por el acceso adicional a actualizaciones de seguridad para paquetes del universo y la posibilidad de utilizar herramientas avanzadas como Ubuntu Security Guide (USG). USG permite aplicar perfiles de hardening validados por organismos internacionales como el Center for Internet Security (CIS), lo cual garantiza que el sistema se configure de acuerdo a estándares reconocidos y auditables. En este proyecto, se utilizó el perfil CIS Level 1 Server Benchmark, adecuado para entornos donde se prioriza la seguridad sin comprometer la funcionalidad.

6.1.2 Vagrant y VirtualBox

Para la creación automatizada y reproducible de la infraestructura, se optó por el uso conjunto de Vagrant y VirtualBox.

Vagrant permite definir entornos de desarrollo o pruebas mediante archivos de configuración (Vagrantfile), lo cual facilita la automatización del aprovisionamiento y elimina la variabilidad entre entornos. Esta herramienta resulta especialmente adecuada en contextos educativos y de investigación donde se requiere replicabilidad exacta del sistema.

VirtualBox, por su parte, se seleccionó como hipervisor por su compatibilidad multiplataforma (Linux, Windows y macOS), su facilidad de uso y su integración directa con Vagrant. Su arquitectura ligera lo hace adecuado para laboratorios virtualizados en equipos personales o aulas de informática sin requerimientos de virtualización avanzada.

6.1.3 Docker

Para el despliegue de los entornos web por alumno, se seleccionó Docker como tecnología de contenerización. Esta decisión se basa en varios factores clave:

- Docker permite aislar aplicaciones y sus dependencias en contenedores ligeros, sin la sobrecarga de una máquina virtual completa.
- Ofrece una gestión modular del entorno, lo que facilita la construcción, despliegue y destrucción de servidores individuales de forma independiente.
- Al contenerizar los servidores Apache y Nginx, se asegura que cada alumno opere en un entorno autocontenido, con su propia configuración de seguridad y red, sin interferir con otros usuarios ni con el sistema anfitrión.
- Docker también mejora la escalabilidad operativa del sistema, ya que permite ejecutar múltiples instancias de forma simultánea, asignando dinámicamente puertos y recursos.

6.1.4 Fail2Ban y PortSentry

Como mecanismos de defensa activa en el sistema anfitrión, se integraron Fail2Ban y PortSentry, ambos ampliamente utilizados en servidores públicos y entornos expuestos a tráfico de red.

- Fail2Ban fue seleccionado por su eficacia en mitigar ataques de fuerza bruta, especialmente en servicios como SSH. Su enfoque basado en monitoreo de logs y generación automática de reglas iptables lo convierte en una solución flexible, de bajo consumo y fácil integración.
- PortSentry complementa esta defensa anticipando intentos de reconocimiento de red mediante escaneos de puertos. Su capacidad de detección en modos activo y sigiloso lo hace ideal para prevenir ataques tempranos, y su configuración personalizable permite adaptarlo a distintas políticas de red.

Ambas herramientas fueron configuradas mediante archivos externos (jail.local, portsentry.conf), lo que permite actualizar sus políticas de respuesta sin modificar el código base del sistema.

6.1.5. Separación de lógica y configuración

Finalmente, se aplicó un principio fundamental del diseño seguro y mantenible: la separación entre lógica de ejecución y configuración. Todas las reglas, políticas y parámetros relevantes (incluyendo cabeceras HTTP, configuraciones de ModSecurity, CRS, reglas de puertos, etc.) se encuentran en archivos independientes, versionables y editables por el administrador.

Esta decisión:

- Mejora la modularidad del sistema.
- Facilita la actualización o endurecimiento posterior sin recompilar ni modificar scripts.
- Refuerza la auditoría y trazabilidad de los cambios realizados.

Esta arquitectura facilita la adaptabilidad del sistema *Aragog* a nuevos entornos, tecnologías o requisitos normativos, y al mismo tiempo permite su mantenimiento eficiente por equipos técnicos con distintos niveles de experiencia.(Ver imagen 6.1)

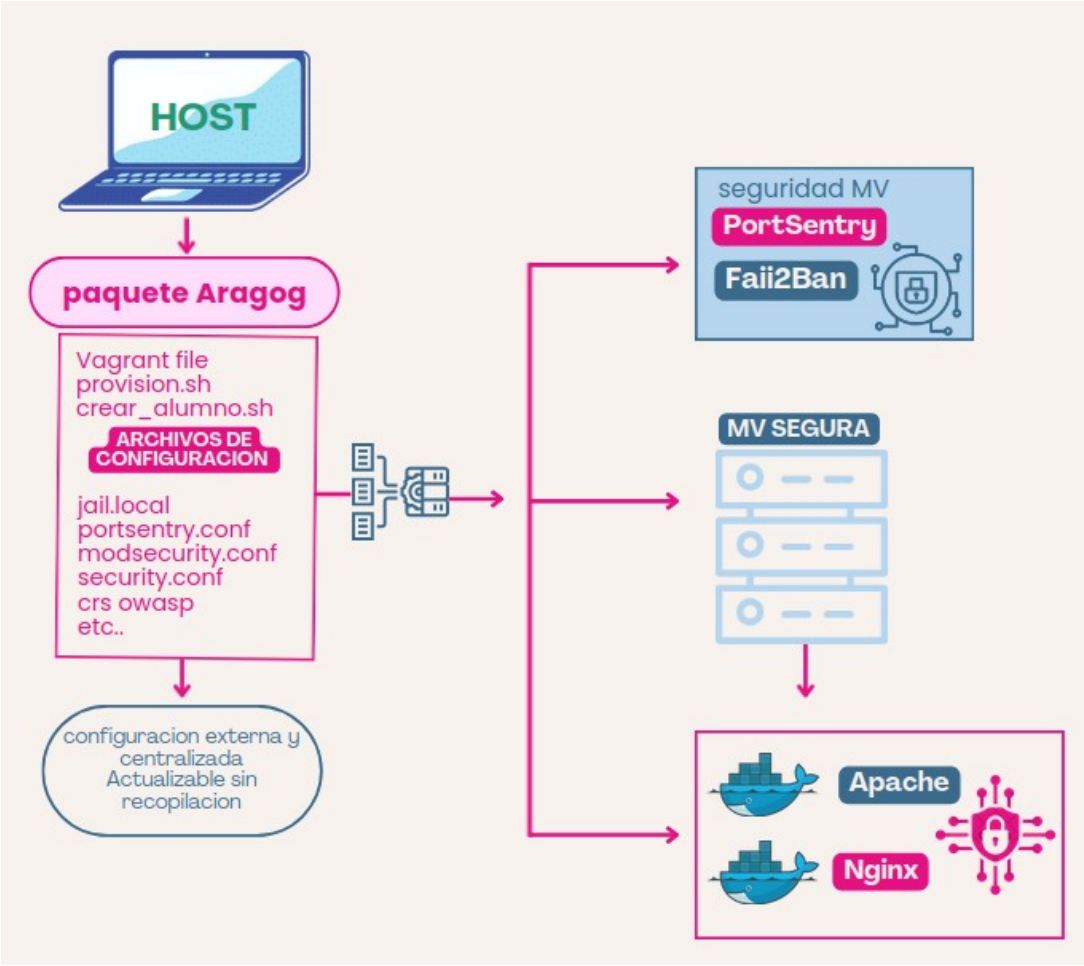


Imagen 6.1. Diagrama técnico de arquitectura Aragog

6.2 Repetibilidad del proceso y replicación por terceros

Uno de los pilares metodológicos del sistema *Aragog* es su capacidad de ser replicado de forma precisa por otros técnicos, docentes o investigadores, sin pérdida de funcionalidad ni necesidad de personalización manual. Esta característica responde a la necesidad de generar entornos reproducibles en contextos académicos y de formación, donde la consistencia, trazabilidad y estandarización del entorno son fundamentales.

Para ello, *Aragog* se distribuye como un repositorio público en GitHub (https://github.com/cuxonet/aragog), el cual contiene todos los archivos necesarios para su ejecución, incluyendo los scripts de provisión (provision.sh), el archivo de configuración de infraestructura (Vagrantfile), los ficheros de configuración de seguridad (Fail2Ban, PortSentry, ModSecurity, etc.), los Dockerfiles para los servidores web, y una estructura organizada y documentada.

Este diseño garantiza que cualquier usuario con un entorno Linux básico, junto con VirtualBox, Vagrant y Git, pueda clonar el repositorio y ejecutar el sistema con un único comando. **vagrant up.**

Este comando levanta una máquina virtual completamente provisionada, con todas las medidas de seguridad aplicadas y los servicios necesarios listos para operar. La provisión incluye actualizaciones, endurecimiento del sistema, instalación de herramientas de defensa activa, configuración del usuario administrador y despliegue de la estructura base para los alumnos.

Además, gracias a la separación entre lógica y configuración, todas las políticas de seguridad están definidas en ficheros externos (por ejemplo, jail.local, portsentry.conf, modsecurity.conf, security.conf, etc.), lo que permite modificar el comportamiento del sistema sin alterar el código base. Esta arquitectura modular y desacoplada facilita la adaptabilidad del sistema a otros entornos, como centros de formación, pruebas de concepto o laboratorios de ciberseguridad.

Asimismo, la documentación estructurada y los scripts comentados permiten a cualquier profesional replicar no solo el entorno técnico, sino también el flujo operativo completo: desde el despliegue de la máquina hasta la creación de contenedores web personalizados por alumno, pasando por la auditoría de seguridad en HTML.

En conjunto, estas decisiones metodológicas permiten que *Aragog* no sea únicamente una solución funcional, sino también una herramienta transferible, auditable y adecuada para investigación, docencia o extensión técnica, cumpliendo con los principios de reproducibilidad exigidos en proyectos académicos rigurosos.

6.3 Instalación del sistema Aragog

Con el fin de facilitar la replicación exacta del sistema *Aragog* en cualquier entorno técnico o académico, se ha desarrollado un flujo de instalación automatizado, reproducible y documentado. El procedimiento completo puede ejecutarse sobre cualquier equipo con sistema operativo Linux y requiere únicamente herramientas ampliamente soportadas: Git, VirtualBox y Vagrant.

A continuación, se describe el proceso completo para la puesta en marcha del sistema desde el repositorio oficial.

Requisitos previos

Para ejecutar el sistema correctamente, el equipo anfitrión debe contar con:

- Un sistema operativo Linux (Ubuntu 20.04 LTS o superior recomendado).
- Git para clonar el repositorio.
- VirtualBox como hipervisor.
- Vagrant para automatizar la creación de la máquina virtual.

1. **Clonación del repositorio**

El repositorio oficial de *Aragog* está alojado públicamente en GitHub. Desde una terminal:

git clone https://github.com/cuxonet/aragog.git

cd aragog

Esto descargará la totalidad del sistema, incluyendo scripts de aprovisionamiento, configuraciones de seguridad, definiciones de contenedores y documentación.

2. **Configuración del token de Ubuntu Pro**

Para aplicar el hardening automatizado mediante el perfil CIS, es necesario contar con una suscripción gratuita a Ubuntu Pro. Una vez obtenido el token, debe editarse el archivo provision.sh para reemplazar el valor correspondiente:

nano provision.sh

Ubicar la línea: sudo pro attach xxxxxxxxx

Y sustituir xxxxxxxxx por el token personal de acceso.

3. **Provisión automática de la máquina virtual**

Con el entorno configurado, se inicia el despliegue completo con:

vagrant up

Este comando ejecutará el proceso automatizado que:

- Crea una máquina virtual basada en Ubuntu 22.04 LTS.
- Aplica el endurecimiento del sistema según el benchmark CIS Level 1 Server.
- Instala herramientas de protección activa como Fail2Ban y PortSentry.
- Crea un usuario administrador con privilegios controlados (adminserver).
- Restringe accesos críticos (como SSH para root).
- Configura las rutas base para entornos de alumno.
- Genera un informe de auditoría (usg_audit_report.html) que detalla el cumplimiento de las políticas de seguridad aplicadas.

4. **Creación de entornos web personalizados**

Con la máquina ya en ejecución, el administrador puede desplegar entornos individuales para los alumnos utilizando el siguiente comando dentro de la máquina virtual:

vagrant ssh

sudo bash /vagrant/crear_alumno.sh

Este script interactivo solicitará la información necesaria del alumno (nombre, identificador UO) y el tipo de servidor web (Apache o Nginx). A partir de ello:

- Se crea un usuario UNIX con carpeta personal aislada.
- Se asigna un puerto dinámico al contenedor.
- Se lanza un contenedor Docker configurado con políticas de seguridad avanzadas (ModSecurity, OWASP CRS, cabeceras HTTP seguras).

Cada entorno es completamente independiente, asegurando aislamiento lógico, trazabilidad y control.

5. **Prueba funcional rápida (opcional)**

Para verificar el correcto funcionamiento de un entorno recién creado, se puede realizar una prueba básica desde el contenedor del alumno:

sudo su - UOXXXXXA # Reemplazar por el ID real del alumno

echo '<h1>Welcome to Aragog</h1>' > index.html

./reparar_permisos.sh

Luego, basta con acceder desde un navegador a la IP de la máquina y el puerto asignado para ver el contenido servido.

Este proceso garantiza que *Aragog* pueda ser desplegado de forma consistente, segura y reproducible, incluso en entornos distintos al del autor, cumpliendo con los principios de transparencia y transferibilidad que se esperan en entornos de investigación aplicada y formación técnica.

6.4 Modularidad, personalización y adaptabilidad del sistema

Uno de los principios rectores en el diseño del sistema *Aragog* ha sido la modularidad. Esta característica se traduce en una arquitectura basada en componentes claramente diferenciados y configuraciones desacopladas, lo que facilita tanto su mantenimiento como su evolución en el tiempo.

Cada aspecto crítico del sistema desde la infraestructura de red y la provisión de servicios hasta las políticas de seguridad aplicadas en los contenedores web ha sido definido mediante archivos externos, fácilmente modificables. Esta estrategia responde a buenas prácticas de ingeniería del software que promueven la separación entre lógica y configuración, permitiendo que la funcionalidad del sistema pueda adaptarse a nuevas necesidades sin alterar el código base.

Gracias a este diseño, cualquier usuario puede:

- Modificar los recursos asignados a la máquina virtual.
- Reemplazar o actualizar imágenes base.
- Integrar nuevas herramientas de seguridad.
- Ampliar el sistema con nuevos servicios Docker.
- Adaptar políticas de seguridad según normativas específicas o futuras versiones del benchmark CIS.

En los apartados siguientes se detallan algunos ejemplos concretos de archivos clave del sistema *Aragog* y cómo pueden ser personalizados para ajustarse a distintos contextos.

6.4.1 Personalización del archivo de configuración de infraestructura (Vagrantfile)

El archivo Vagrantfile es el punto de partida para definir la infraestructura virtual sobre la cual se despliega el sistema *Aragog*. A través de este archivo, se establece la configuración inicial de la máquina virtual que sirve como entorno anfitrión para los contenedores web y las herramientas de seguridad.

Actualmente, el Vagrantfile especifica:

- La imagen base del sistema operativo: Ubuntu 22.04 LTS (ubuntu/jammy64), una distribución moderna, estable y compatible con Ubuntu Pro.
- Un entorno de red privada mediante DHCP, que permite asignar direcciones IP locales a la VM sin necesidad de configuración manual.
- Asignación de recursos controlada: por defecto, se asigna 1 CPU y 1 GB de RAM, lo cual permite ejecutar el sistema en la mayoría de los equipos personales.
- La ejecución automática de un script de aprovisionamiento (provision.sh) al iniciar por primera vez la máquina virtual, lo que garantiza la aplicación inmediata de todas las medidas de seguridad definidas.

Posibilidades de adaptación y crecimiento

El diseño del archivo permite su modificación sin alterar el resto del sistema. Algunas de las extensiones o adaptaciones posibles incluyen:

- Aumentar los recursos asignados (por ejemplo, RAM o núcleos de CPU) en entornos donde se necesite mayor capacidad de procesamiento, como sesiones prácticas con múltiples usuarios simultáneos.
- Cambiar la imagen base para utilizar otras versiones de Ubuntu (como ubuntu/focal64 para 20.04) o incluso otras distribuciones compatibles con Vagrant, siempre que se ajusten los scripts de aprovisionamiento en consecuencia.
- Modificar el tipo de red a “pública” para permitir el acceso externo desde otras máquinas físicas o laboratorios.

- Definir entornos diferenciados mediante múltiples archivos de provisión (por ejemplo, provision_dev.sh, provision_prod.sh, etc.), lo cual permite mantener perfiles distintos para desarrollo, pruebas o despliegue en producción académica.

En conjunto, el archivo Vagrantfile actúa como un punto central de configuración de infraestructura, cuya adaptabilidad permite escalar el sistema, integrarlo con nuevas herramientas o ajustarlo a nuevas necesidades sin comprometer su diseño general. Esta estructura consolida la naturaleza modular, predecible y mantenible de *Aragog*.

6.4.2 Automatización del sistema base: `provision.sh`

El archivo provision.sh constituye el núcleo del proceso de configuración automatizada del sistema *Aragog*. Se trata de un script que se ejecuta automáticamente al iniciar por primera vez la máquina virtual mediante Vagrant, y cuya misión es transformar una instalación básica del sistema operativo en una plataforma segura, operativa y lista para desplegar servidores web en contenedores aislados.

Este script realiza tareas fundamentales como:

- La actualización integral del sistema operativo.
- La activación de una cuenta de Ubuntu Pro para aplicar configuraciones avanzadas de seguridad.
- La instalación y configuración de herramientas de protección activa como Fail2Ban y PortSentry.
- La habilitación del perfil de hardening CIS Level 1 Server Benchmark mediante Ubuntu Security Guide (USG).
- La creación de usuarios de administración con privilegios restringidos.
- La implementación de restricciones de acceso remoto (por ejemplo, la desactivación de acceso SSH para root).
- La integración de scripts auxiliares (como el generador de entornos web o el reparador de permisos).

La estructura modular del script, junto con el uso de archivos de configuración externos, permite que su comportamiento pueda modificarse fácilmente sin necesidad de alterar su lógica principal. De este modo, se favorece la mantenibilidad y la adaptabilidad del sistema a diferentes escenarios operativos.

Posibilidades de adaptación y crecimiento

El diseño del provision.sh está pensado para admitir cambios orientados a distintos contextos de uso o necesidades técnicas, entre las que destacan:

- Ampliación del perfil de hardening: el script actualmente aplica el perfil cis_level1_server, recomendado para entornos educativos y de uso general. Sin embargo, puede adaptarse para aplicar otros perfiles más estrictos, como cis_level2_server (para entornos más críticos) o cc_eal4+, enfocado en cumplimiento de certificaciones de alta exigencia. Este cambio requiere simplemente modificar una línea del script.
- Evolución de la cuenta de Ubuntu Pro: el sistema ha sido diseñado para operar bajo la versión gratuita de Ubuntu Pro, que permite la activación de políticas de seguridad básicas y el uso de USG. No obstante, si se desea acceder a funcionalidades avanzadas (como soporte ampliado, validación de cumplimiento normativo empresarial, o acceso a más perfiles de hardening), se puede sustituir el token gratuito por una suscripción profesional sin modificar el resto del sistema.
- Adición de nuevas herramientas de seguridad: en entornos donde la máquina virtual vaya a cumplir funciones adicionales (más allá de alojar contenedores web), el script puede modificarse para incluir nuevas herramientas como antivirus, IDS/IPS, escáneres de vulnerabilidades, servicios de monitorización o mecanismos de autenticación centralizada.
- Habilitación o eliminación de restricciones: en función del contexto, puede que algunas medidas —como la desactivación del acceso SSH para determinados usuarios— deban relajarse o endurecerse. Estas configuraciones se encuentran claramente definidas en el script, lo que permite modificar el comportamiento sin reescribir el flujo principal.
- Personalización por entorno: el script puede duplicarse y adaptarse para generar versiones diferenciadas (por ejemplo, provision_docencia.sh, provision_pentest.sh, provision_demo.sh), permitiendo mantener un único sistema base pero con múltiples variantes de despliegue.

En conjunto, el archivo provision.sh ejemplifica el enfoque modular y extensible de *Aragog*, ofreciendo un equilibrio entre automatización completa y capacidad de adaptación técnica. Esta versatilidad garantiza su aplicabilidad en múltiples entornos, tanto educativos como corporativos, sin necesidad de rediseñar el sistema desde cero.

6.4.3 Automatización de entornos web personalizados: crear_alumno.sh

El script crear_alumno.sh es el componente que permite al administrador del sistema desplegar entornos web seguros y aislados para cada alumno de forma totalmente automatizada. Su diseño modular no solo simplifica la gestión de múltiples usuarios, sino que permite mantener un control riguroso sobre la seguridad, la trazabilidad y la escalabilidad de los entornos creados.

Este script ejecuta un conjunto de operaciones encadenadas que incluyen:

- Solicitud de los datos básicos del alumno (nombre completo e identificador UO).
- Selección del tipo de servidor web a utilizar (Apache o Nginx).
- Creación de un usuario UNIX con credenciales seguras, sin acceso privilegiado.
- Generación de una carpeta personal bajo la estructura /home/WEBS_ALUMNOS/, con permisos restringidos y controlados.
- Mapeo de dicha carpeta como volumen compartido con el contenedor web del alumno.
- Verificación de la existencia de la imagen Docker correspondiente y, en caso necesario, su construcción desde los Dockerfile existentes.
- Despliegue de un contenedor web individual con políticas de seguridad reforzadas y asignación dinámica de un puerto no utilizado.

Capacidad de adaptación

El enfoque modular adoptado en este script permite realizar ajustes sin modificar su lógica principal. Entre las posibilidades de personalización destacan:

- Cambio o ampliación de los tipos de servidores disponibles: es posible incorporar nuevos entornos (por ejemplo, servidores Node.js, Python/Flask o PHP-FPM) simplemente añadiendo nuevos bloques condicionales y directorios con sus respectivos Dockerfiles.
- Ajuste de políticas de creación de usuarios: el administrador puede modificar los parámetros de nombre de usuario, formato de contraseñas o shell por defecto para adaptarse a políticas institucionales o corporativas.
- Redefinición de la estructura de directorios: si se desea utilizar otra jerarquía de almacenamiento (por ejemplo, rutas específicas por curso o asignatura), el script puede modificarse sin afectar la gestión de contenedores.
- Integración con sistemas externos de gestión de identidad: si se desea escalar el sistema a un entorno más amplio, el script puede adaptarse para integrarse con directorios LDAP o bases de datos de usuarios, permitiendo la sincronización automática de cuentas.
- Automatización de recursos adicionales: es posible añadir nuevas operaciones durante la creación del entorno, como el despliegue de bases de datos, plantillas web base o certificados TLS autofirmados, todo sin alterar el núcleo del sistema.

Este enfoque modular, centrado en la ejecución controlada de acciones mediante un único punto de entrada (crear_alumno.sh), permite a *Aragog* escalar horizontalmente, adaptarse a nuevos escenarios docentes o técnicos, y mantener la coherencia y seguridad de los entornos generados.

6.4.4 Arquitectura modular de los Dockerfiles

El sistema *Aragog* implementa el despliegue de servidores web a través de contenedores Docker construidos desde imágenes definidas en archivos Dockerfile, ubicados en los directorios apache/ y nginx/. Estos archivos establecen los entornos de ejecución seguros para los alumnos, encapsulando configuraciones del servidor, dependencias, módulos de seguridad y puntos de montaje para los volúmenes compartidos.

Los Dockerfiles no solo cumplen la función de instanciar entornos de servicio funcionales, sino que representan una capa fundamental dentro del diseño modular del sistema: cada contenedor es autoconfigurado, versionable, portable y fácilmente extensible, lo que permite mantener la integridad del sistema global sin comprometer la independencia de sus componentes.

Arquitectura de construcción

- Apache:
    - Utiliza una imagen base de Ubuntu 22.04.
    - Instala Apache2 junto con el módulo de protección mod_security2.
    - Habilita configuraciones de seguridad mediante el archivo security.conf.
    - Carga reglas de ModSecurity y habilita su modo de bloqueo activo.
    - Expone el puerto 80 y configura el servidor para servir contenido desde un volumen externo (/var/www/html).
- Nginx:
    - Se construye sobre la imagen oficial owasp/modsecurity:nginx, optimizada para la integración de ModSecurity v3.
    - Incorpora de forma explícita el conjunto de reglas OWASP Core Rule Set (CRS), incluyendo su archivo de configuración y mapeo de Unicode.
    - Reemplaza la configuración por defecto de Nginx con una definición estricta que incluye políticas de cabecera seguras.
    - Expone de igual forma el puerto 80, sirviendo el contenido desde /usr/share/nginx/html.

Potencial de personalización

El carácter modular y desacoplado de estos Dockerfiles permite su modificación con mínimos riesgos colaterales:

- Integración de nuevos stacks tecnológicos (PHP-FPM, Node.js, Python/Flask, entre otros).

- Incorporación de herramientas de monitoreo o métricas (como Prometheus o supervisores de procesos).
- Inclusión de políticas de acceso más estrictas mediante capas de autenticación externa (como JWT, OAuth, LDAP).
- Definición de contenedores diferenciados por nivel de prácticas o perfil de usuario (por ejemplo, servidores con vulnerabilidades intencionadas para entornos de pentesting controlado).

La arquitectura basada en Docker no solo aporta ligereza y portabilidad, sino que garantiza la consistencia entre entornos, algo esencial en contextos educativos y de formación técnica.

6.4.5 Políticas de seguridad embebidas en contenedores

La seguridad en entornos de ejecución web no debe limitarse únicamente a la protección del sistema operativo o a las medidas perimetrales del host. En *Aragog*, los servidores Apache y Nginx desplegados en contenedores Docker están reforzados mediante múltiples capas de configuración defensiva, incorporadas directamente en sus respectivos entornos. Esta estrategia garantiza que cada instancia sirva contenido desde una posición de seguridad activa, alineada con los estándares actuales de ciberseguridad para aplicaciones web.

Todas las medidas implementadas se apoyan en archivos de configuración externos, lo que permite su revisión, actualización o sustitución sin requerir la reconstrucción completa de las imágenes. Esta separación entre lógica y configuración, además de facilitar el mantenimiento, permite adaptar el nivel de protección a distintos perfiles de uso (pruebas, docencia, demostraciones o entornos simulados).

Mecanismos de protección implementados

ModSecurity es el componente central del cortafuegos de aplicaciones web (WAF) integrado en ambos servidores:

- En Apache, se activa mediante el módulo mod_security2, con su política configurada explícitamente en modo activo (SecRuleEngine On). La configuración es gestionada a través de un archivo externo (modsecurity.conf) incluido en el contenedor y adaptable a necesidades específicas.
- En Nginx, se utiliza una imagen base especializada (owasp/modsecurity:nginx), que integra ModSecurity v3 nativamente y permite una configuración granular, compatible con versiones actuales del OWASP CRS.

OWASP Core Rule Set (CRS) se encuentra plenamente implementado en el contenedor Nginx. Este conjunto de reglas, mantenido por la comunidad OWASP, proporciona protección frente a las amenazas más comunes recogidas en el OWASP Top 10, incluyendo:

- Inyección de código (SQL, XSS, OS Command Injection, etc.).
- Acceso a recursos no autorizados.
- Manipulación de cabeceras HTTP.
- Evasión de políticas de sesión o autenticación.

Estas reglas son actualizables de forma independiente, permitiendo incorporar nuevas versiones del CRS sin modificar la estructura del contenedor.

Cabeceras de seguridad del lado del servidor

Ambos servidores integran una serie de cabeceras HTTP estrictas configuradas por defecto, alineadas con las mejores prácticas en seguridad del lado cliente. Entre ellas:

- Content-Security-Policy (CSP): restringe la carga de scripts, estilos, fuentes y otros recursos a orígenes confiables.
- Strict-Transport-Security (HSTS): obliga al navegador a utilizar HTTPS en futuras conexiones.
- X-Frame-Options: impide la incrustación del sitio en otros dominios, mitigando ataques de clickjacking.
- X-Content-Type-Options: desactiva la inferencia automática de tipos MIME.
- Referrer-Policy: limita el envío de cabeceras de referencia, protegiendo la privacidad del usuario.

Estas cabeceras son gestionadas mediante ficheros de configuración externos (security.conf), uno por servidor, y su contenido puede modificarse o ampliarse sin intervención sobre el código de los contenedores.

Escalabilidad de la configuración defensiva

La arquitectura propuesta en *Aragog* permite ajustar progresivamente el nivel de protección sin afectar la disponibilidad del sistema. Gracias a su diseño desacoplado y modular:

- Las reglas de ModSecurity pueden habilitarse, deshabilitarse o personalizarse por entorno o grupo de usuarios.
- El CRS puede actualizarse o sustituirse por perfiles alternativos (por ejemplo, para prácticas de testing controlado).
- Las cabeceras pueden adaptarse para cumplir regulaciones específicas como RGPD, ENS, LOPDGDD o normas corporativas.[23]
- Pueden incorporarse capas adicionales de protección (autenticación, firma de contenidos, integración con proxies inversos) manteniendo el mismo diseño base.

Este enfoque, en el que la seguridad se considera un componente activo del entorno de ejecución y no una capa añadida, consolida la orientación de *Aragog* hacia entornos reales, seguros y replicables, acordes con los retos de la formación técnica moderna.

7. RESULTADOS OBTENIDOS: Validación Técnica Aragog

7.1 Casos de uso

Este apartado presenta los resultados de las pruebas técnicas realizadas sobre el sistema *Aragog*, incluyendo datos empíricos y su interpretación directa, con el objetivo de evaluar el cumplimiento de los objetivos planteados. Cada resultado se acompaña de evidencia generada por el propio sistema (capturas, logs o reportes) y un análisis técnico objetivo.

7.2 Auditoría del sistema base: cumplimiento del perfil CIS

Tras completar el proceso de aprovisionamiento automatizado de la máquina virtual de *Aragog*, se ejecuta de forma automática una auditoría de seguridad utilizando la herramienta Ubuntu Security Guide (USG). Esta evaluación mide el grado de cumplimiento del sistema frente al perfil CIS Ubuntu 22.04 Level 1 Server Benchmark, un estándar ampliamente reconocido en la industria para sistemas operativos seguros en entornos productivos y educativos.

El informe generado, en formato HTML (ver Imagen 1), proporciona una evaluación detallada de las configuraciones aplicadas, clasificadas según su cumplimiento, severidad y referencia al marco SCAP (Security Content Automation Protocol). El informe incluye:

- Número total de reglas evaluadas: 245.
- Reglas aplicadas correctamente: 238.
- Reglas no cumplidas: 7.
- Puntuación final: 94.43% de cumplimiento.

Interpretación del resultado

El resultado obtenido indica un nivel de conformidad alto y consistente con las recomendaciones del perfil CIS Level 1, diseñado para asegurar servidores en producción sin afectar su funcionalidad. Las 7 reglas no satisfechas corresponden a configuraciones que, por razones de seguridad operativa o contexto educativo, no se automatizan deliberadamente. Entre ellas se encuentran:

- Parámetros de políticas de contraseñas que deben definirse por el administrador según el reglamento de la institución.
- Valores de bloqueo de sesión, auditoría de login o tiempos de inactividad que requieren ajustes contextuales.
- Configuraciones que dependen de servicios no habilitados por defecto (por ejemplo, syslog remoto o certificados de infraestructura externa).

Estas omisiones no suponen un fallo del sistema, sino una decisión consciente de diseño, con el objetivo de mantener la flexibilidad operativa sin sacrificar el cumplimiento estructural del estándar. Además, el informe generado se guarda en la raíz del entorno compartido (/vagrant/usg_audit_report.html), permitiendo su revisión y trazabilidad posterior por parte del administrador.

Este resultado valida que la configuración automatizada de la máquina virtual, mediante el script provision.sh y el perfil USG-CIS aplicado, alcanza un nivel técnico de cumplimiento alto, demostrando la capacidad de *Aragog* para generar entornos seguros desde el primer despliegue.

7.3 Protección frente a ataques de fuerza bruta: Fail2Ban

Uno de los vectores más comunes de ataque en sistemas accesibles por red es la fuerza bruta contra el servicio SSH. Para mitigar esta amenaza, *Aragog* incluye la herramienta Fail2Ban, configurada para monitorizar los registros de autenticación del sistema (/var/log/auth.log) y reaccionar automáticamente ante patrones sospechosos de intento de acceso no autorizado.

Con el fin de validar esta funcionalidad, se llevó a cabo una simulación de ataque desde el host hacia la máquina virtual, replicando el comportamiento de un atacante externo que intenta forzar credenciales SSH incorrectas.(Ver Imagen 7.1)

Simulación del ataque

Desde el sistema host, se ejecutó un script que realizaba seis intentos consecutivos de acceso SSH utilizando el usuario admin, pero sin aportar una clave válida, lo que generó respuestas de denegación de acceso por parte del servidor

Imagen 7.1 – Intentos de conexión fallidos mediante SSH desde el host.

Activación del mecanismo de defensa

Tras alcanzar el umbral de intentos fallidos definido en la configuración de Fail2Ban, el sistema identificó el comportamiento como potencialmente malicioso. A continuación, se procedió automáticamente al bloqueo de la IP de origen (192.168.56.1) tanto a nivel de registro como de firewall, tal como se evidencia en la salida del cliente de control y en los logs del sistema (ver Imagen 7.2):

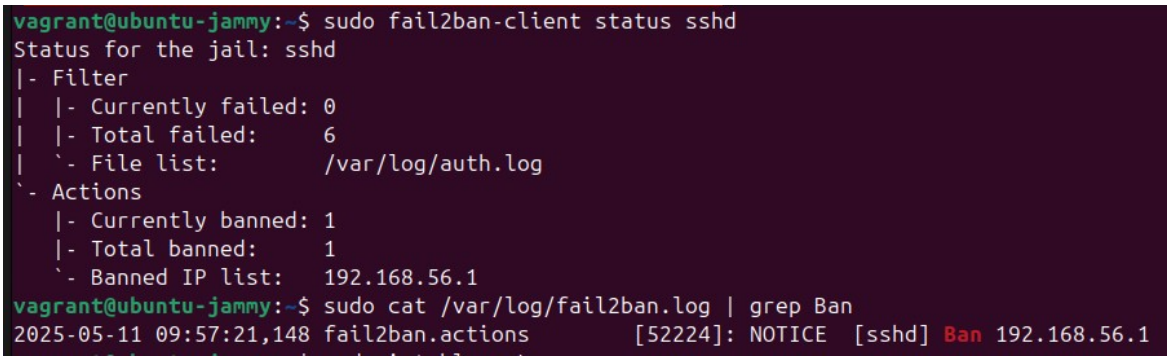


Imagen 7.2 – Estado del *jail* sshd mostrando la IP bloqueada, el número total de intentos y el evento registrado en fail2ban.log

Aplicación de reglas en iptables

Complementariamente, Fail2Ban gestiona la inclusión de reglas temporales en iptables para reforzar la defensa del sistema ante nuevas conexiones desde la IP atacante. La configuración activa del firewall refleja esta acción explícita (ver imagen **7.3**):

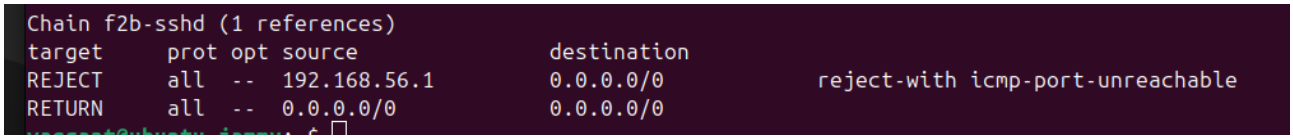


Imagen 7.3 – Reglas activas en iptables para el *jail* f2b-sshd

Valoración del resultado

La prueba realizada valida de forma concluyente la capacidad del sistema *Aragog* para detectar e impedir automáticamente intentos de intrusión por fuerza bruta, en este caso dirigidos contra el servicio SSH. La respuesta del sistema —bloqueando la dirección IP atacante tras un número determinado de intentos fallidos— se produce de forma completamente autónoma, sin requerir intervención administrativa directa, cumpliendo así con los principios de defensa proactiva exigidos en entornos educativos seguros y controlados.

Es importante destacar que esta prueba representa solo una de las múltiples reglas y escenarios que Fail2Ban es capaz de manejar. Gracias a su diseño modular y su integración con los registros del sistema, Fail2Ban puede adaptarse fácilmente para proteger otros servicios expuestos,

como FTP, SMTP, Apache, Nginx o cualquier aplicación que registre eventos de autenticación o abuso. La arquitectura flexible del sistema permite al administrador modificar o extender las políticas de bloqueo simplemente editando los archivos de configuración, sin necesidad de reinstalación ni recompilación del servicio.

En ese sentido, la demostración realizada no solo valida una configuración concreta, sino que evidencia el potencial de Fail2Ban como plataforma de defensa extensible, plenamente alineada con el enfoque modular y mantenible que define a *Aragog*. Esta prueba se asienta como una muestra puntual del amplio abanico de respuestas automatizadas que el sistema puede desplegar frente a amenazas reales.

7.4 Detección temprana de escaneos de puertos

Con el objetivo de validar las capacidades de detección activa de *Aragog* ante amenazas externas, se realizó una prueba simulada de escaneo de puertos desde el equipo host hacia la máquina virtual. Este tipo de escaneo es una técnica comúnmente utilizada por atacantes para identificar servicios expuestos y vulnerables.(Ver Imagen 7.4)

La herramienta utilizada fue **Nmap**, ejecutada con el siguiente comando:

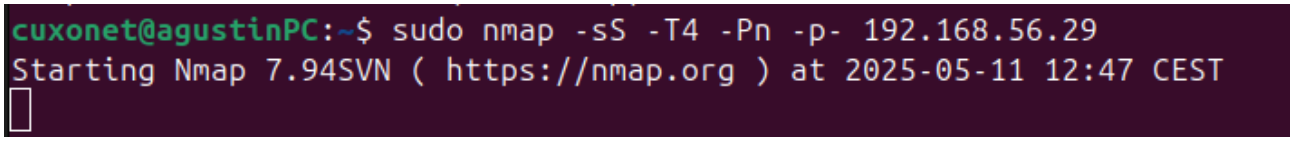


Imagen 7.4. Comando ejecutado desde el host (escaneo de puertos completo).

Durante el escaneo, PortSentry, configurado en modo avanzado (atcp), detectó la actividad como maliciosa y ejecutó automáticamente acciones de defensa: el bloqueo inmediato de la IP atacante mediante inserción en el archivo /etc/hosts.deny y su registro en el archivo de eventos bloqueados.(Ver imagen 7.5)

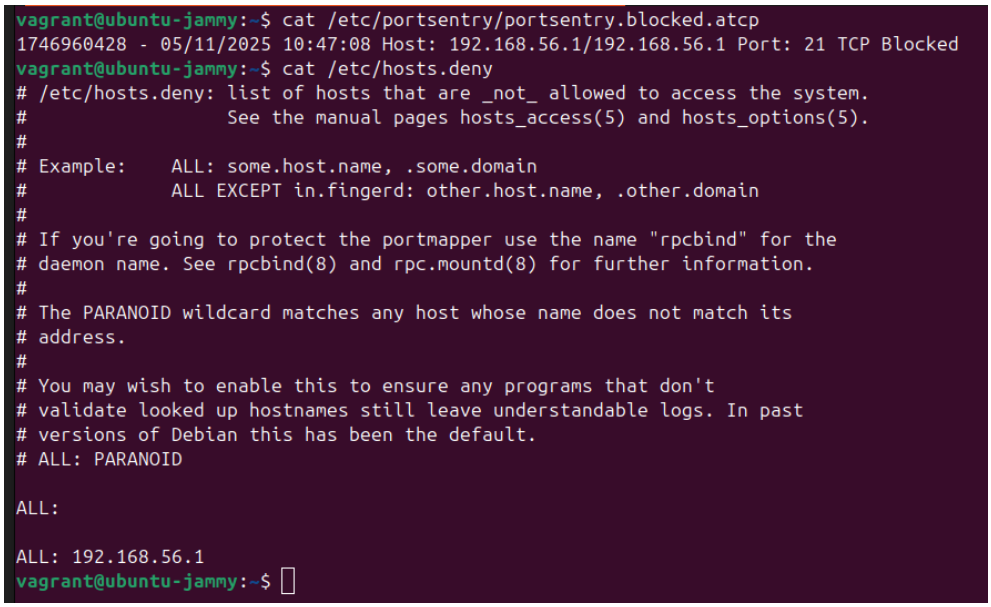


Imagen 7.5. PortSentry registra la IP 192.168.56.1 como origen del escaneo y la bloquea en el puerto 21 TCP, tambien se observa la confirmación del bloqueo en el archivo /etc/hosts.deny

Valoración del resultado

La prueba realizada demuestra con claridad la eficacia del sistema Aragog en la detección temprana de actividades de reconocimiento de red, concretamente escaneos de puertos —una técnica frecuentemente utilizada como paso preliminar a ataques más avanzados.

Gracias a la configuración en modo agresivo (atcp) de PortSentry y a su integración con mecanismos de respuesta automática como la modificación de reglas de iptables y la edición de /etc/hosts.deny, la reacción ante el intento de escaneo es inmediata y silenciosa. La IP atacante es bloqueada sin generar alertas visibles para el atacante, lo cual representa una defensa pasiva altamente efectiva.

Este resultado valida no solo la correcta configuración de PortSentry en el sistema, sino también su valor estratégico como medida de defensa proactiva. La posibilidad de definir reglas mediante ficheros externos permite ajustar la agresividad del sistema o adaptar su comportamiento ante distintos escenarios de amenaza, sin necesidad de reinstalación o reinicio de servicios.

Por tanto, esta prueba no solo confirma que el sistema puede detectar y bloquear escaneos de puertos con éxito, sino que también demuestra su capacidad de adaptación, escalabilidad y mantenimiento a largo plazo, factores esenciales en infraestructuras educativas donde las condiciones de red pueden variar con frecuencia y los intentos de exploración o abuso son comunes.

7.5 Detección y mitigación de ataques web

Con el objetivo de evaluar la capacidad del sistema Aragog para identificar y bloquear ataques web en tiempo real, se configuró un entorno web dentro de un contenedor, protegido por Apache2, el firewall de aplicaciones web ModSecurity v3, y el conjunto de reglas de seguridad OWASP CRS 3.3.2.

Se realizaron pruebas simuladas desde el equipo host hacia el contenedor, representando cuatro vectores de ataque comúnmente utilizados por actores maliciosos:

- Inyección SQL (SQLi)
- Cross-Site Scripting (XSS)
- Inclusión local de archivos (LFI)
- Ejecución remota de comandos (RCE)

Simulación de ataques

Para comprobar la capacidad de detección de amenazas en el contenedor Apache protegido por ModSecurity, se lanzaron cuatro ataques web comunes desde el equipo host utilizando curl. Cada solicitud fue diseñada para simular un tipo específico de ataque

Inyección SQL (SQLi)

curl "http://192.168.56.29:8080/?id=1%27%20OR%20%271%27=%271"

Cross-Site Scripting (XSS)

curl "http://192.168.56.29:8080/?q=<script>alert('xss')</script>"

Ejecución remota de comandos (RCE)

curl "http://192.168.56.29:8080/?cmd=ls%20/etc"

Inclusión local de archivos (LFI)

curl "http://192.168.56.29:8080/?page=../../../../etc/passwd"

(Ver imagen 7.6)

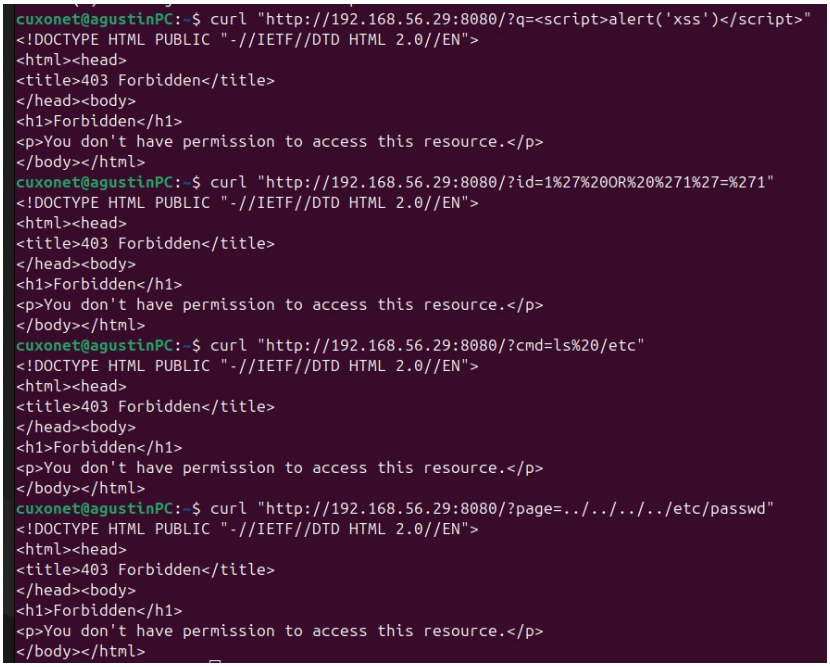


Imagen 7.6. Ataque del host al servidor apache.

Detección y respuesta del sistema

Las solicitudes maliciosas fueron correctamente identificadas y bloqueadas por el firewall de aplicaciones web. ModSecurity, en conjunto con las reglas del OWASP Core Rule Set, actuó durante la fase 2 del procesamiento HTTP, aplicando una serie de mecanismos defensivos:

- Análisis profundo del contenido y parámetros de las solicitudes.
- Evaluación de patrones de ataque mediante firmas predefinidas.
- Asignación de una puntuación de riesgo (*anomaly score*) a cada solicitud.
- Bloqueo inmediato si la puntuación total superaba el umbral de tolerancia (≥5).

En todos los casos evaluados, el sistema respondió con un código HTTP 403 Forbidden, denegando el acceso al recurso solicitado. Los eventos fueron registrados con detalle en los logs de auditoría de ModSecurity, donde se observan(Ver imagen 7.7):

- La activación de reglas específicas, como 930120 para intentos de inclusión de archivos locales (LFI), y 932160 para intentos de ejecución remota de comandos (RCE).
- La asignación de un *Inbound Anomaly Score* de 43, que excede ampliamente el umbral definido por la política de seguridad.
- La clasificación detallada del tipo de amenaza mediante etiquetas (attack-lfi, attack-rce, etc.) y estándares de cumplimiento como PCI.

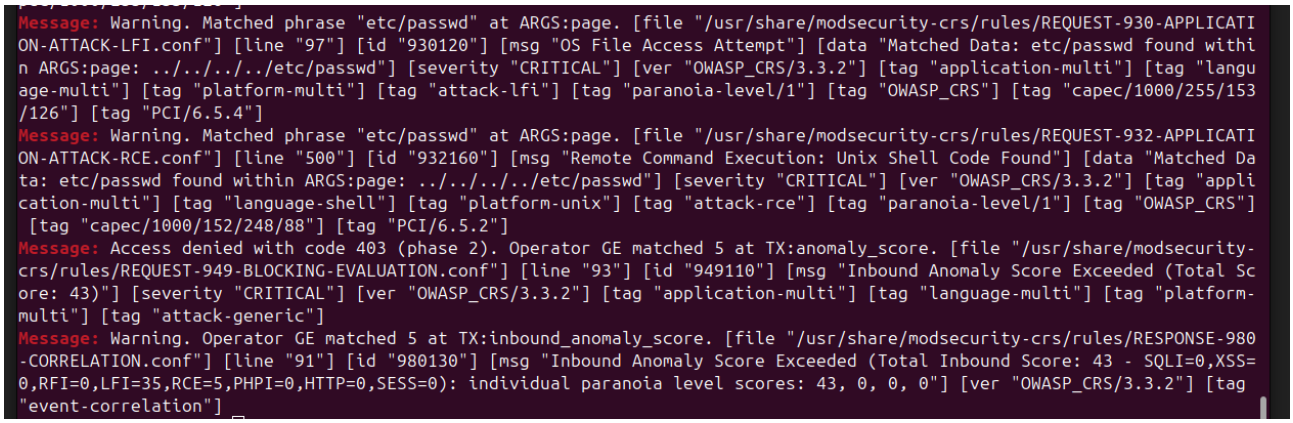


Imagen 7.7. Logs de auditoria ModSecurity.

Valoración del resultado

La ejecución de estas pruebas demuestra con claridad la eficacia del entorno contenedorizado del sistema Aragog en la detección y mitigación de amenazas web. El uso de ModSecurity, en combinación con OWASP CRS, proporciona una capa de defensa activa capaz de responder automáticamente ante ataques, sin necesidad de intervención manual ni interrupción del servicio.

A diferencia de soluciones puramente reactivas, este sistema actúa en el momento de la solicitud, evaluando el contenido y los patrones del tráfico HTTP en tiempo real. Esta capacidad de inspección profunda es crítica para prevenir ataques como SQLi o XSS, que muchas veces no generan errores visibles pero pueden comprometer gravemente los datos.

Además, la arquitectura del contenedor permite mantener una separación clara entre el entorno de aplicación y el sistema base, lo que añade una capa adicional de aislamiento ante eventuales vulnerabilidades.

Este experimento valida que el sistema no solo es capaz de detectar ataques web complejos, sino también de responder de forma inmediata y autónoma, bloqueando el acceso sin exponer detalles técnicos al atacante. En contextos educativos o redes semiabiertas, donde los intentos de explotación son frecuentes, este nivel de protección es esencial.

El sistema demostró ser:

- Eficiente en la detección de amenazas mediante reglas especializadas.
- Flexible y escalable gracias a su implementación en contenedores.
- Fácilmente mantenible, al centralizar los logs y las reglas en ubicaciones conocidas.

7.6 Test de Auditoria

Una parte fundamental de la seguridad en sistemas automatizados es su capacidad para autoevaluarse y detectar configuraciones inseguras o desviaciones de los estándares establecidos. En Aragog, este aspecto se cubre mediante herramientas de auditoría automática que validan el cumplimiento normativo desde la fase de provisión inicial.

Como ya se mencionó, el sistema genera automáticamente un informe HTML al finalizar el aprovisionamiento, utilizando la herramienta Ubuntu Security Guide (USG) y el perfil CIS Level 1 Server Benchmark. El resultado alcanzado fue un cumplimiento superior al 94%, lo que refleja una alta conformidad con las recomendaciones internacionales de seguridad (ver imagen 5.3).

Adicionalmente, se aplicó un escaneo con Lynis, herramienta externa que permite comparar el estado de seguridad entre sistemas. En una instalación básica de Ubuntu sin medidas de hardening, el puntaje fue de 62 puntos, indicando una configuración débil. (ver imagen 7.8)

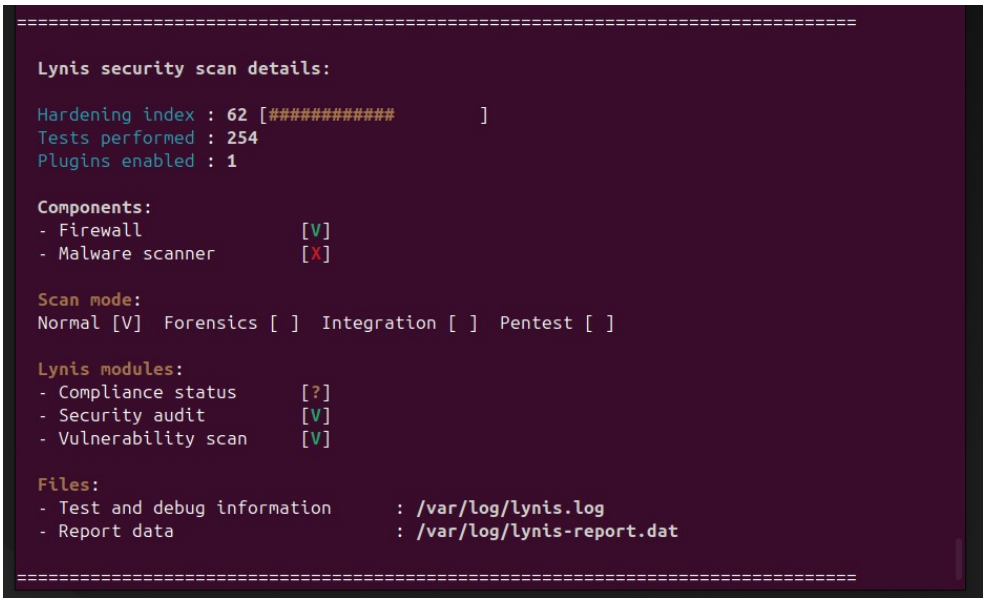


Imagen 7.8. Scanner Lynis Servidor No seguro.

En cambio, la máquina generada con Aragog obtuvo 82 puntos, alcanzando el nivel verde de seguridad (ver imagen 7.9). Este resultado se logró sin intervención manual, destacando la eficacia del enfoque automatizado.

Cabe mencionar que muchas advertencias en el análisis de Lynis no corresponden a vulnerabilidades críticas, sino a recomendaciones opcionales. Lograr un nivel alto de seguridad desde el inicio garantiza entornos más robustos, auditables y listos para producción.

El índice de hardening (Hardening Index) proporcionado por Lynis se utiliza como métrica cuantitativa de la postura de seguridad. Un valor por encima de 80 representa una configuración con buenas prácticas de seguridad implementadas.[24]

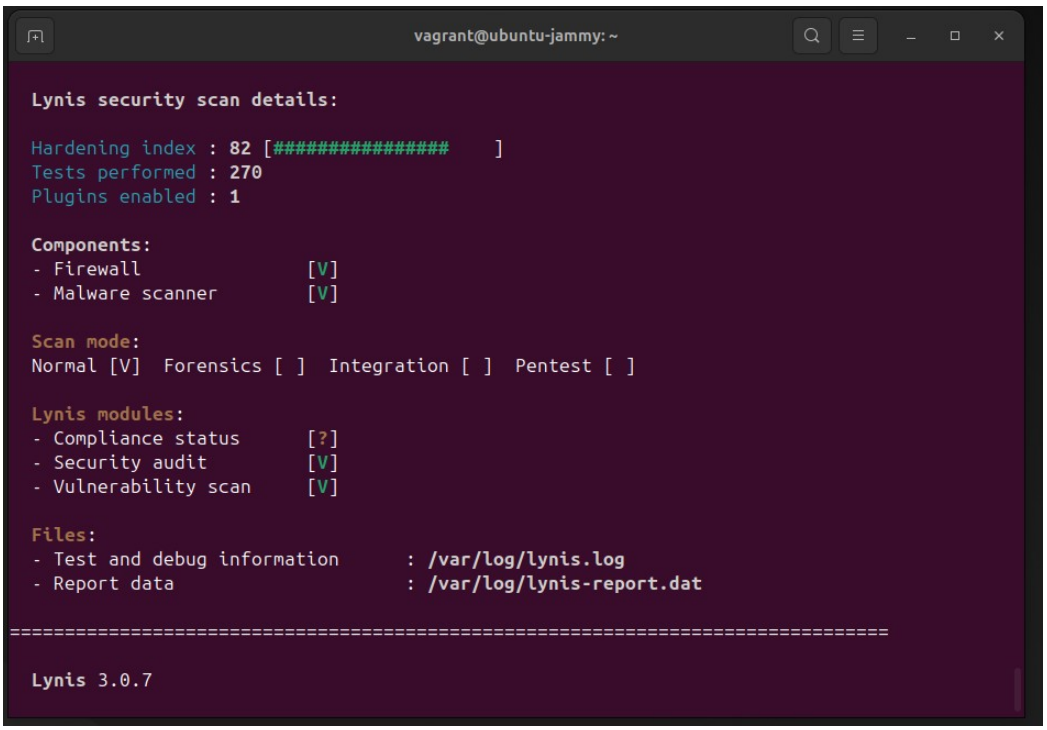


Imagen 7.9. Scanner Lynis ARAGOG

Para fortalecer aún más la validación, se utilizó Nessus Expert, una de las soluciones de escaneo de vulnerabilidades más avanzadas y utilizadas en entornos corporativos. A través de una prueba de red externa con su perfil predeterminado, el contenedor desplegado con Aragog no arrojó vulnerabilidades críticas, altas, medias ni bajas (ver imagen 7.10). Esta validación desde una herramienta externa refuerza la confiabilidad del sistema ante escaneos reales realizados por auditores o atacantes.

La combinación de estas pruebas demuestra que el entorno generado por Aragog cumple con requisitos técnicos sólidos en seguridad, tanto a nivel interno como externo, y está preparado para ser utilizado en contextos exigentes sin comprometer la integridad de los servicios o de los usuarios

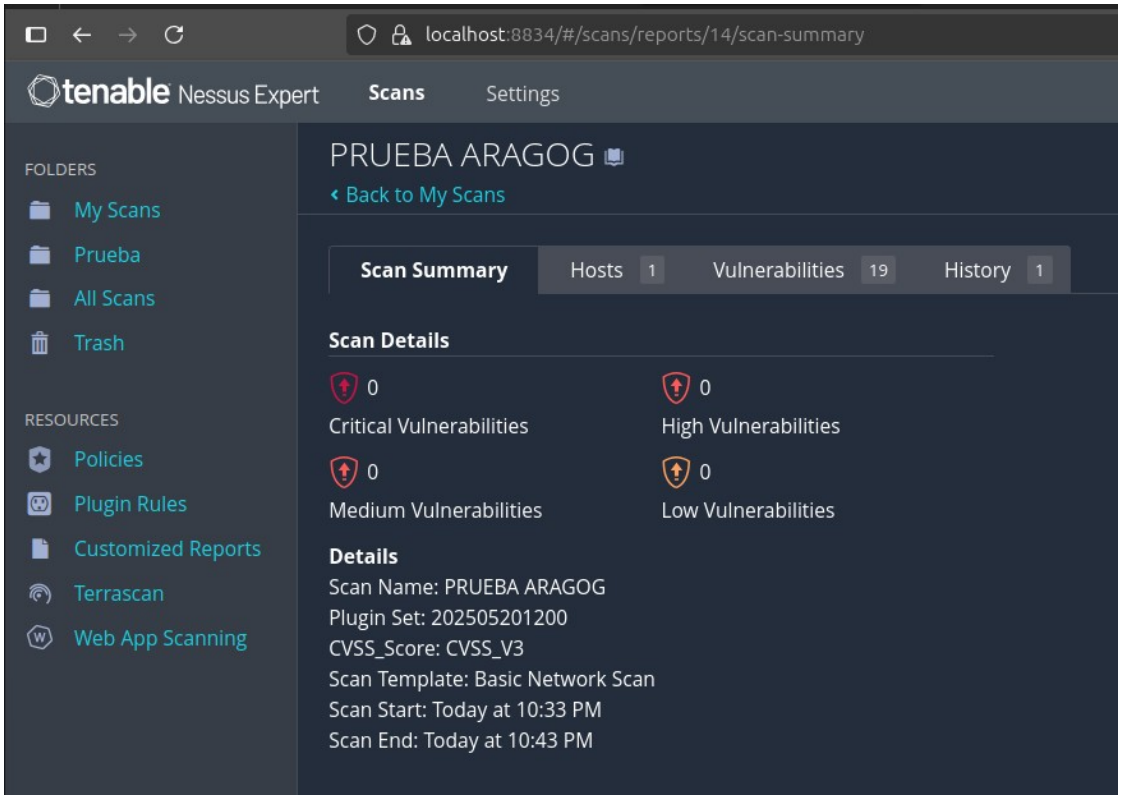


Imagen 7.10. Scanner Nessus Expert ARAGOG

Para validar la efectividad del sistema, se realizó el mismo escaneo con Nessus Expert sobre una máquina sin hardening. El resultado evidenció 6 vulnerabilidades altas y 1 media, en contraste con el entorno Aragog, que no presentó ninguna.(Ver imagen 7.11)

Esta comparación demuestra que las medidas automatizadas aplicadas por Aragog tienen un impacto real y efectivo en la reducción de riesgos de seguridad, confirmando su valor como entorno seguro y fiable para formación y despliegue web.

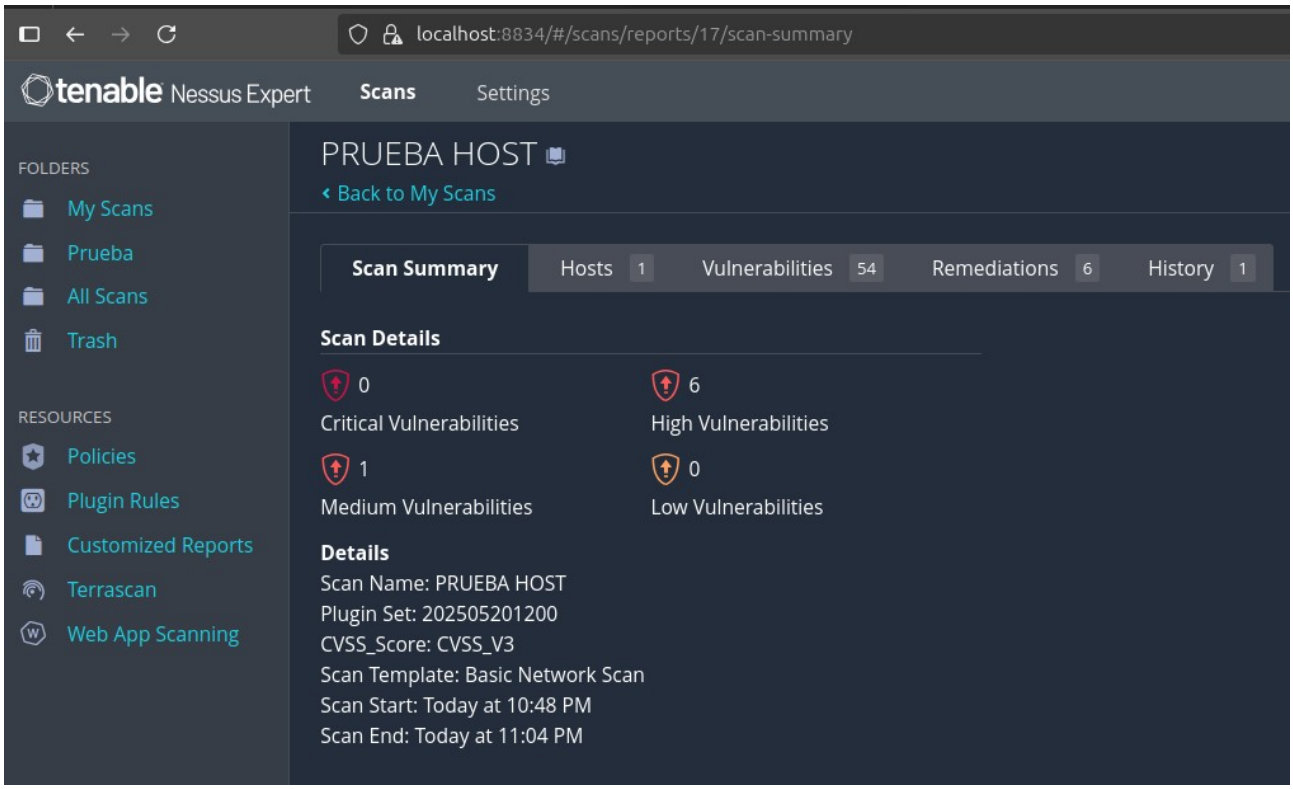


Imagen 7.11. Scanner Nessus Expert a MV no segura.

7.7 Comparativa de Seguridad: ARAGOG vs Linux Base

Herramienta / Escaneo	ARAGOG (Automatizado)	Servidor Linux Base
Lynis (Hardening Index)	82	62
USG - CIS Level 1 Audit	95%	No se puede ejecutar.
Tiger Report	Sin errores ni advertencias	Múltiples advertencias
Nessus (vulnerabilidades)	0 críticas, 2 medias	1 críticas, 6 Altas
Firewall activado	Sí (UFW configurado)	No
SSH endurecido	Sí (config personalizado)	No
Auditoría del sistema	Activa	Inactiva

Imagen 7.12. Comparativa

8. Conclusiones y Trabajo Futuro

8.1 Conclusión de resultados

Los resultados han validado la eficacia de Aragog ante ataques reales, como fuerza bruta, escaneo de puertos o inyecciones web, además de demostrar su conformidad con normas de seguridad a través de pruebas de autoevaluación. Destacan los autotests automáticos implementados en el script de provisión, así como escaneos externos con Lynis y Nessus, que confirman el alto nivel de protección frente a sistemas no endurecidos.

Gracias a su diseño modular y mantenible, Aragog se presenta como una plataforma robusta y adaptable, útil tanto en docencia como en investigación práctica sobre ciberseguridad. Como mejora futura, se proyecta una interfaz web de gestión que facilite el despliegue, la asignación de accesos y la supervisión de contenedores de forma visual y segura.

En definitiva, Aragog cumple con los objetivos planteados y sienta las bases para una gestión automatizada y segura de entornos web en el ámbito educativo.

8.2 Trabajo Futuro

Como línea de evolución natural, el sistema Aragog podría ampliarse mediante el desarrollo de una plataforma web de administración centralizada, orientada a mejorar la experiencia del administrador y permitir una gestión más eficiente, escalable y segura de los entornos de cada alumno.

Esta interfaz, basada en frameworks modernos como Django (Python) permitiría:

- Gestión de usuarios: altas, bajas y modificación de datos de los estudiantes a través de formularios seguros.
- Asignación dinámica de contenedores: integración con scripts existentes para lanzar entornos web de forma automatizada.
- Control de estado: visualización en tiempo real de qué contenedores están activos, qué recursos utilizan y desde qué IPs están accediendo.
- Gestión de credenciales: generación y envío automático de claves de acceso vía correo electrónico cifrado.
- Integración con bases de datos: almacenamiento estructurado de logs, acciones del administrador, historial de despliegues y auditorías.
- Alertas de seguridad: notificaciones automáticas cuando Fail2Ban o PortSentry detecten intentos de intrusión.

- Dashboard visual: resumen del estado global de la infraestructura, número de usuarios activos, vulnerabilidades detectadas o contenedores en uso.

Además, podrían explorarse futuras integraciones con sistemas de autenticación federada (LDAP, SSO), soporte para API RESTful que permita interacción con otras herramientas universitarias o de análisis, e incluso mecanismos de backup automático de configuraciones y datos de usuario.

Todo este desarrollo debería mantener el principio rector del proyecto: automatización con seguridad, garantizando que las nuevas funcionalidades no introduzcan vectores de ataque adicionales ni comprometan la robustez del sistema actual.

9. Bibliografía

BIBLIOGRAFÍA

[1] Center for Internet Security, “CIS Benchmarks,” [Online]. Available: https://www.cisecurity.org/cis-benchmarks [Accessed: 18-May-2025].

[2] ModSecurity Project, “ModSecurity Web Application Firewall,” [Online]. Available: https://modsecurity.org [Accessed: 18-May-2025].

[3] Fail2Ban Project, “Fail2Ban: Log monitoring and intrusion prevention,” [Online]. Available: https://www.fail2ban.org [Accessed: 18-May-2025].

[4] Canonical, “Ubuntu Pro with CIS Benchmark,” [Online]. Available: https://ubuntu.com/pro [Accessed: 18-May-2025].

[5] Apache Software Foundation, “Apache HTTP Server,” [Online]. Available: https://httpd.apache.org [Accessed: 18-May-2025].

[6] F5 Inc., “NGINX Web Server,” [Online]. Available: https://www.nginx.com [Accessed: 18-May-2025].

[7] OWASP Foundation, “OWASP Top Ten Web Application Security Risks,” [Online]. Available: https://owasp.org/www-project-top-ten [Accessed: 18-May-2025].

[8] Psionic Technologies, “PortSentry: Port Scan Detection and Response,” [Online]. Available: https://sourceforge.net/projects/sentrytools [Accessed: 18-May-2025].

[9] Canonical, “Ubuntu Security Guide (USG),” [Online]. Available: https://ubuntu.com/security/certifications/docs/usg [Accessed: 18-May-2025].

[10] Cloudflare, “What is a Web Application Firewall (WAF)?,” [Online]. Available: https://www.cloudflare.com/es-es/learning/ddos/glossary/web-application-firewall-waf/ [Accessed: 18-May-2025].

[11] HashiCorp, “Vagrant by HashiCorp,” [Online]. Available: https://www.vagrantup.com. [Accessed: 18-May-2025].

[12] Docker Inc., “What is Docker?,” [Online]. Available: https://www.docker.com/resources/what-container [Accessed: 18-May-2025].

[13] Oracle, “VirtualBox,” [Online]. Available: https://www.virtualbox.org [Accessed: 18-May-2025].

[14] IES Granadilla, “Curso Docker 2021 – Organización Multiusuario,” [Online]. Available: https://iesgn.github.io/curso_docker_2021/sesion2/organizacion.html [Accessed: 18-May-2025].

[15] A. Roca, “Implementación de entornos virtualizados multiusuario para laboratorios educativos,” IES Ingeniero de la Cierva, Murcia, 2009. [Online]. Available: https://www.murciaeduca.es/iesingenierodelacierva/sitio/upload/DAI1_SIMR.2009.pdf [Accessed: 18-May-2025].

[16] Canonical, “UFW – Uncomplicated Firewall,” [Online]. Available: https://help.ubuntu.com/community/UFW [Accessed: 18-May-2025].

[17] The OpenBSD Project, “OpenSSH: Portable Secure Shell,” [Online]. Available: https://www.openssh.com [Accessed: 18-May-2025].

[18] Let’s Encrypt, “Free SSL/TLS Certificates,” [Online]. Available: https://letsencrypt.org. [Accessed: 18-May-2025].

[19] National Institute of Standards and Technology, “Security Content Automation Protocol (SCAP),” [Online]. Available: https://csrc.nist.gov/projects/security-content-automation-protocol [Accessed: 18-May-2025].

[20] Docker Inc., “Container Isolation and Kernel Namespaces,” [Online]. Available: https://docs.docker.com/engine/security/isolation/ [Accessed: 18-May-2025].

[21] OWASP Foundation, “ModSecurity Core Rule Set Docker Image,” [Online]. Available: https://github.com/SpiderLabs/owasp-modsecurity-crs [Accessed: 18-May-2025].

[22] OWASP Foundation, “OWASP Top 10 – 2021: The Ten Most Critical Web Application Security Risks,” [Online]. Available: https://owasp.org/Top10/ [Accessed: 18-May-2025].

[23] Agencia Española de Protección de Datos, “Guía sobre el cumplimiento del RGPD y la LOPDGDD,” [Online]. Available: https://www.aepd.es/es/guias-y-herramientas/guias [Accessed: 18-May-2025].

[24] CISOfy, “Lynis Security Auditing Tool,” [Online]. Available: https://cisofy.com/lynis/ [Accessed: 18-May-2025].

10. ANEXOS

10.1 Plan de gestión de riesgos

10.1.1 Metodología

Este Plan de Gestión de Riesgos es el plan para el conjunto de riesgos identificados para el proyecto Aragog. Según el PMBOK “El riesgo de un proyecto es un evento o condición incierta que, de producirse, tiene un efecto positivo o negativo en uno o más de los objetivos del proyecto, tales como el alcance, el cronograma, el costo y la calidad.” La metodología para la gestión del riesgo tiene dos aspectos diferentes:

1. **Metodología General**. Metodología para el conjunto del Plan de Gestión de Riesgos.

2. **Metodología de Gestión de Riesgos**. Metodología para el ciclo de vida de todos los riesgos y las interrelaciones entre los riesgos identificados.

10.1.2 Metodología General

La Metodología General describe el proceso de gestión de riesgos para el conjunto del proyecto: desde el inicio hasta el final de los trabajos relacionados con el proyecto.

Identificación inicial de riesgos.

Primer contacto con los riesgos del proyecto. Algunas técnicas útiles para la identificación inicial de riesgos son las siguientes:

- Tormenta de ideas
- Juicio de expertos
- Tabla DAFO

Elaboración del Plan de Gestión de Riesgos.

- Creación del Plan de Gestión
- Los criterios para la gestión de riesgos
- El Registro de Riesgos y la Hoja de Datos de riesgos para todos los riesgos priorizados.

Monitorización de Riesgos mientras el proyecto está avanzando:

Nombre del proceso	Descripción
Evaluación de riesgos regular	De vez en cuando, el riesgo debe ser evaluado con el fin de asegurar que no representa una amenaza para el avance del proyecto.
Actualización del Plan de Gestión de Riesgos	Muchos riesgos se mantienen durante todo el proyecto, pero no todos ellos. Algunos riesgos tienen un ciclo de vida más corto que la duración del proyecto. Por otro lado, a medida que el proyecto evoluciona, pueden aparecer nuevos riesgos u otros ya identificados, pero no importantes hasta el momento, pueden convertirse en una amenaza para el proyecto.
Actualización del plan del proyecto por decisiones de la evaluación de riesgos	Se deben tomar decisiones durante el proyecto con el fin de evitar o, al menos, minimizar el impacto de los riesgos.

Tabla. 10.1 Monitorización de riesgos

Informe Final de Riesgos.

Al final del proyecto, se elaborará un informe final que mostrará toda la evolución de los riesgos.

10.1.3 Metodología de Gestión de Riesgos

- Los aspectos más técnicos de la gestión de riesgos están involucrados en este aspecto de la metodología. Para ese proyecto definimos seis pasos:
- Planificar la Gestión de Riesgos: Es el proceso por el cual se define cómo realizar las actividades de gestión de los riesgos para un proyecto.
- Identificar los Riesgos: Es el proceso por el cual se determinan los riesgos que pueden afectar el proyecto y se documentan sus características.
- Realizar el Análisis Cualitativo de Riesgos: Es el proceso que consiste en priorizar los riesgos para realizar otros análisis o acciones posteriores, evaluando y combinando la probabilidad de ocurrencia y el impacto de dichos riesgos.
- Realizar el Análisis Cuantitativo de Riesgos: Es el proceso que consiste en analizar numéricamente el efecto de los riesgos identificados sobre los objetivos generales del proyecto.
- Planificar la Respuesta a los Riesgos: Es el proceso por el cual se desarrollan opciones y acciones para mejorar las oportunidades y reducir las amenazas a los objetivos del proyecto.
- Monitorizar y Controlar los Riesgos: Es el proceso por el cual se implementan planes de respuesta a los riesgos, se rastrean los riesgos identificados, se monitorizan los riesgos

residuales, se identifican nuevos riesgos y se evalúa la efectividad del proceso contra riesgos a través del proyecto

10.1.4 Conceptos Generales

Con el fin de evitar la ambigüedad, se deben definir tres conceptos utilizados en la Gestión de Riesgos:

- **Riesgo:** Es cualquier evento que pueda causar un efecto en el proyecto. Este efecto puede ser una amenaza o una oportunidad. Las amenazas deben ser evitadas, mientras que las oportunidades deben ser aprovechadas.
- **Factor de riesgo:** Son circunstancias asociadas con el riesgo que aumentan la probabilidad de que se produzcan los efectos descritos por dicho riesgo. Por lo general, los riesgos son difíciles de medir o controlar directamente, por lo que las decisiones se toman controlando sus factores con el fin de evitar, minimizar o potenciar sus efectos.
- **Indicadores de riesgo:** Son elementos asociados al riesgo y/o a sus factores, que pueden medirse y sirven como referencia para estimar la probabilidad de que ocurra el efecto del riesgo.

10.2 Herramientas y Tecnología

Tormenta de Ideas

Esta técnica se utilizará inicialmente para identificar la lista principal de riesgos. Consiste en una discusión orientada a los principales objetivos del proyecto y a todas las posibles situaciones que podrían salir mal.

Juicio de Expertos

Se empleará en conjunto con la tormenta de ideas para consolidar y validar la identificación de riesgos, aprovechando la experiencia técnica del equipo involucrado.

Evaluaciones Periódicas

De forma regular, se evaluarán todos los indicadores de riesgo según el cronograma establecido en la hoja de riesgos. Los resultados serán analizados en las reuniones de gestión correspondientes.

Reuniones

Se incluirá un punto específico en el orden del día de las reuniones de seguimiento, con el objetivo de tomar decisiones relacionadas con los riesgos identificados y sus posibles impactos.

Método Delphi

En caso de desacuerdo sobre la identificación o gestión de algún riesgo, se podrá aplicar el método Delphi como técnica de consenso estructurado para tomar decisiones fundamentadas.

Diagramas Causa-Efecto

Dado que las relaciones entre riesgos y factores desencadenantes pueden ser complejas, se podrán utilizar diagramas de causa-efecto (también conocidos como diagramas de Ishikawa) para representar gráficamente estas interacciones.

Cálculos Estadísticos

En los casos necesarios, se aplicarán métodos estadísticos para realizar evaluaciones cuantitativas de los riesgos, incluyendo estimaciones de impacto y probabilidad.

Otros Métodos

Se podrá recurrir a herramientas y modelos matemáticos adicionales que apoyen los análisis cuantitativos y la toma de decisiones en función de la criticidad del riesgo.

10.3 Roles y Responsabilidades

Autor del proyecto

Dirige y realiza el seguimiento de los riesgos además de resolver los problemas surgidos durante su manejo e integración de su gestión en la gestión del proyecto. Gestiona los riesgos no asignados a los siguientes responsables.

Consultor Técnico de Revisión

Se encarga de supervisar y gestionar los riesgos que atienden a las fases de investigación de la gestión y desarrollo de los proyectos.

Responsable de Sistemas

Supervisa y gestiona los riesgos relacionados con las infraestructuras cuyo uso se incluye en sus proyectos, ya sean propias o contratadas.

10.4 Presupuesto

El presupuesto de Gestión de riesgos se muestra a continuación.

Item	Concepto	Asignación (€)
1	Identificación de riesgos	55 €
2	Análisis y priorización de los riesgos	45 €
3	Planificación de los riesgos	45 €
4	Definición de planes de contingencia	80 €
5	Actualización y monitorización de los riesgos	80 €
	TOTAL	305 €

Tabla. 10.2 Presupuesto de Riesgos

10.5 Calendario

N/A

10.6 Categorías de Riesgo

A fin de poder identificar los riesgos y conocer la estructura de los mismos se categorizan dentro de una de las siguientes categorías, pudiendo pertenecer cada uno de ellos a más de una categoría:

Técnico

- Requisitos
- Tecnología
- Prestaciones y fiabilidad
- Calidad

Organizacional

- Dependencias del proyecto
- Recursos
- Financiación
- Personal.

Gestión del proyecto

- Estimación
- Planificación
- Control
- Comunicación

Externo

- Proveedores
- Usuario
- Tiempo
- Fenómenos

10.7 Definiciones de Probabilidad

Nivel de probabilidad	Rango porcentual	Descripción
Muy baja	0% - 20%	Es altamente improbable que el riesgo llegue a ocurrir.
Baja	20% - 40%	La posibilidad de que el riesgo se materialice es reducida.
Media	40% - 60%	Existe una probabilidad moderada de que el riesgo ocurra.
Alta	60% - 80%	Es bastante probable que la situación de riesgo se presente en el desarrollo del proyecto.
Muy alta	80% - 100%	El riesgo tiene una probabilidad muy elevada de afectar al proyecto.

Tabla 10.3 Definiciones de impacto por objetivos

10.8 Matriz de Probabilidad e Impacto

Probabilidad \ Impacto	Muy Bajo (0.05)	Bajo (0.15)	Medio (0.30)	Alto (0.55)	Crítico (0.90)
Muy Alta (0.90)	0.05	0.14	0.27	0.50	0.81
Alta (0.70)	0.04	0.11	0.21	0.39	0.63
Media (0.50)	0.03	0.08	0.15	0.28	0.45
Baja (0.30)	0.02	0.05	0.09	0.17	0.27
Muy Baja (0.10)	0.01	0.02	0.03	0.06	0.09

Tabla 10.4 Matriz

10.9 Planes de contingencia

10.9.1 Presupuesto

En caso de que resulte necesario modificar el presupuesto establecido y aprobado previamente al inicio del proyecto, dicha variación será evaluada y valorada por el Tutor del proyecto junto con el Administrador de Sistemas. Bajo ninguna circunstancia, esta modificación podrá superar el 5% del coste total previsto para el proyecto.

10.9.2 Planificación

Aunque la fecha de finalización del proyecto está programada para el 20 de mayo de 2025, se considera la posibilidad de ajustes en la planificación debido a los riesgos inherentes a este Cualquier ampliación de plazos será válida siempre que el trabajo adicional no implique un incremento superior al 5% del presupuesto aprobado, según lo indicado en el apartado anterior.

10.9.3 Formatos de la Documentación

Para la correcta gestión de la documentación del proyecto, se tomarán como referencia las siguientes normativas internacionales:

- UNE-ISO 31000:2010 – Gestión del riesgo
- UNE-EN 31010:2011 – Gestión del riesgo. Técnicas de evaluación del riesgo

10.9.4 Seguimiento

La estrategia para el seguimiento de los riesgos se basa en las siguientes directrices:

- Los riesgos identificados serán evaluados y su impacto recalculado mensualmente durante toda la ejecución del proyecto.

- Cada mes se llevará a cabo un análisis para detectar posibles nuevas amenazas que puedan surgir.
- La revisión de los riesgos se realizará mediante reuniones con las partes interesadas (stakeholders) involucradas o afectadas por dichos riesgos.