**Plan de Seguridad para el Proyecto**

**1. Seguridad en el balanceo y alta disponibilidad**

* **HAProxy**
  + Configurar **SSL/TLS** con certificados válidos (Cloudflare o Let’s Encrypt).
  + Forzar **HTTPS** y deshabilitar protocolos inseguros (SSLv3, TLS 1.0/1.1).
  + Limitar **ciphers** a suites seguras.
  + Configurar protección contra:
    - HTTP request smuggling.
    - Slowloris (timeout y límites de conexiones).
  + Activar **rate limiting** para mitigar ataques de fuerza bruta o DDoS de baja intensidad.
  + **Preservar IP real del cliente en los logs**:
    - Usar cabecera X-Forwarded-For o PROXY protocol.
    - Configurar los servidores backend (Apache, NGINX o PHP) para registrar la IP real y no la del HAProxy.
* **Keepalived**
  + Asegurar que la comunicación VRRP esté restringida a la red interna.
  + Usar autenticación VRRP (AH) para evitar que nodos falsos tomen la IP flotante.

**2. Seguridad en Docker y Docker Swarm**

* Limitar los **privilegios** de los contenedores (--cap-drop y evitar --privileged).
* Definir **límites de CPU y RAM** para prevenir que un contenedor malicioso o defectuoso sature el host.
* Evitar montar volúmenes sensibles como /var/run/docker.sock en contenedores.
* Usar **redes internas** de Swarm para la comunicación entre servicios y bloquear acceso desde fuera.
* Escanear imágenes Docker:
  + Antes de subirlas al registry privado.
  + Usar herramientas como **Trivy o Clair** para buscar vulnerabilidades.
* Usar **versiones específicas** de imágenes, no latest.

**3. Seguridad en la base de datos**

* **MariaDB + InnoDB / Galera**
  + Activar **cifrado en tránsito** (SSL/TLS) entre nodos y hacia la aplicación.
  + Activar **cifrado en reposo** para datos sensibles.
  + Crear usuarios con privilegios mínimos necesarios.
  + Deshabilitar acceso remoto a root y usar contraseñas seguras.
  + Implementar **firewall interno** para que solo los nodos de aplicación accedan al puerto de MariaDB.
  + Monitorear replicación y logs para detectar inconsistencias o accesos sospechosos.

**4. Seguridad en el almacenamiento (GlusterFS)**

* Usar cifrado en tránsito (**TLS**) en GlusterFS.
* Restringir acceso solo a nodos autorizados (control de IPs).
* Activar autenticación con certificados.
* Monitorear integridad de archivos para detectar cambios no autorizados.

**5. Seguridad en la capa de aplicación (WordPress / Drupal)**

* Mantener el núcleo, plugins y temas actualizados.
* Eliminar plugins/temas no usados.
* Configurar permisos de archivos restrictivos (640 para archivos y 750 para directorios).
* Usar un **WAF** (puede ser el de Cloudflare o mod\_security).
* Desactivar edición de archivos desde el panel (DISALLOW\_FILE\_EDIT en WordPress).
* Proteger /wp-admin o /user con autenticación adicional.
* Activar plugins de seguridad:
  + WordPress: Wordfence, iThemes Security.
  + Drupal: Security Kit, Login Security.

**6. Seguridad en la red**

* **Firewall por capas**:
  + Capa host (ufw, firewalld, iptables).
  + Capa de red (Cloudflare WAF + reglas personalizadas).
* Bloquear puertos innecesarios.
* Segmentar la red interna (base de datos, almacenamiento, administración).
* Activar monitoreo de tráfico para detectar patrones anormales.
* Configurar nftables para:
  + Restringir acceso a puertos sensibles (SSH, DB, admin panels).
  + Permitir solo IPs internas autorizadas entre nodos.

**7. Protección contra ataques comunes**

* **SQL Injection, XSS, CSRF**: Mitigados desde la app y WAF.
* **Brute force**: Rate limiting en HAProxy y/o Cloudflare.
* **DDoS**: Protección de Cloudflare y limitación de conexiones en HAProxy.
* **Malware en archivos subidos**: Escanear con ClamAV u otro AV antes de almacenarlos en GlusterFS.
* **Fail2ban**:
  + Monitorizar logs de SSH, Apache, HAProxy, etc.
  + Configurar filtros personalizados para detectar patrones de ataque.
  + Integrar con nftables (banaction = nftables-multiport).
  + Establecer políticas de ban (ej: 3 intentos fallidos → 1h de bloqueo).

**8. Pruebas de seguridad**

* **Pentesting interno**: Usar OWASP ZAP, Nikto, sqlmap.
* **Pruebas de rendimiento bajo ataque**: Apache JMeter con escenarios simulando alta carga.
* **Análisis de vulnerabilidades**: OpenVAS, Nessus o similar.
* **Verificación de configuración segura**: Lynis o Docker Bench for Security.

**9. Monitoreo y alertas**

* Centralizar logs en **ELK Stack o Loki**.
* Configurar alertas (Prometheus + Alertmanager o Grafana).
* Revisar regularmente logs de HAProxy, Docker, MariaDB y GlusterFS.
* Activar detección de intrusiones (Fail2ban, Wazuh, CrowdSec).

**10. Plan de respuesta a incidentes**

* Procedimiento documentado para:
  + Aislar un nodo comprometido.
  + Restaurar backups (base de datos y GlusterFS).
  + Revocar credenciales comprometidas.
  + Notificar al equipo.
  + Registrar el incidente y acciones tomadas.