Estudiante 1: Vulnerabilidades de Autenticación y Rate Limiting

1. **Brute Force Protection** (`01-brute-force.test.js`)

   - Implementar rate limiting en el login

   - Añadir límite de intentos fallidos (máximo 5)

   - Implementar bloqueo temporal de cuentas

   - Archivos a modificar: `backend/src/controllers/authController.js`, `backend/src/routes/auth.js`

2. **Insecure CAPTCHA** (`06-insecure-captcha.test.js`)

   - Implementar validación segura del CAPTCHA

   - Evitar que se pueda manipular el resultado del lado del cliente

   - Archivos a modificar: `backend/src/controllers/captchaController.js`, `backend/src/routes/captcha.js`

Tests a verificar: `01-brute-force.test.js`, `06-insecure-captcha.test.js`

Estudiante 2: Inyecciones SQL

1. **SQL Injection** (`07-sql-injection.test.js`)

   - Reemplazar concatenación de strings en consultas SQL

   - Implementar prepared statements/queries parametrizadas

   - Validar y sanitizar todas las entradas de usuario

   - Archivos a modificar: `backend/src/controllers/productController.js`, `backend/src/config/database.js`

2. **Blind SQL Injection** (`08-blind-sql-injection.test.js`)

   - Proteger consultas que filtran por ID

   - Implementar validación estricta de parámetros numéricos

   - Archivos a modificar: `backend/src/controllers/productController.js`

Tests a verificar: `07-sql-injection.test.js`, `08-blind-sql-injection.test.js`

Estudiante 3: Command Injection y CSRF

1. **Command Injection** (`02-command-injection.test.js`)

   - Eliminar ejecución de comandos del sistema (`exec`, `spawn`)

   - Implementar alternativas seguras con librerías nativas

   - Validar y sanitizar entradas antes de procesar

   - Archivos a modificar: `backend/src/controllers/vulnerabilityController.js`

2. **CSRF Protection** (`03-csrf-protection.test.js`)

   - Implementar generación de tokens CSRF

   - Validar tokens en peticiones POST/PUT/DELETE

   - Configurar cookies con flags seguros (SameSite, Secure)

   - Archivos a modificar: `backend/src/server.js`, `backend/src/middleware/`, `frontend/src/services/api.ts`

Tests a verificar: `02-command-injection.test.js`, `03-csrf-protection.test.js`

Estudiante 4: Seguridad de Archivos

1. **File Inclusion (LFI)** (`04-file-inclusion.test.js`)

   - Prevenir path traversal (../../../)

   - Validar que solo se acceda a archivos permitidos

   - Implementar whitelist de archivos/directorios

   - Archivos a modificar: `backend/src/controllers/vulnerabilityController.js`

2. **File Upload Vulnerabilities** (`05-file-upload.test.js`)

   - Validar extensiones de archivo (whitelist)

   - Verificar tipo MIME real del archivo

   - Limitar tamaño de archivos

   - Renombrar archivos subidos para evitar ejecución

   - Archivos a modificar: `backend/src/controllers/uploadController.js`, `backend/src/config/multer.js`

Tests a verificar: `04-file-inclusion.test.js`, `05-file-upload.test.js`

Estudiante 5: Integración, Testing y Documentación

1. **Configuración y Testing**

- Verificar que Docker Compose funcione correctamente
- Ejecutar todos los tests de seguridad y reportar resultados
- Validar que la aplicación funcione end-to-end
- Crear scripts de verificación automatizada

2. **Documentación**

- Documentar cada vulnerabilidad encontrada
- Explicar las correcciones implementadas por el equipo
- Crear guía de "antes y después" con ejemplos de código
- Preparar README con instrucciones de ejecución y testing
- Documentar buenas prácticas de seguridad aplicadas

3. **Revisión de Código**

- Revisar los PRs/commits de los otros estudiantes
- Verificar que las correcciones sean completas
- Sugerir mejoras adicionales de seguridad

Entregables: Documentación completa, reporte de tests, validación de integración