CONFECCIÓN DE LA PLANTILLA DE EVALUACIÓN DE RIESGOS

Paso 1: Entender qué hace el programa

Antes de tocar la plantilla, hay que tener claro qué es lo que hace el programa:

- Es una tienda en consola que:
    - Registra usuarios con nombre, email y contraseña.
    - Guarda productos fijos con id, nombre y precio.
    - Permite que un usuario, indicando su email, haga pedidos y se guarden en una lista.
    - Muestra el listado de pedidos.

Con esto ya se ve que se tratan datos personales (nombre, email, contraseña) y datos de compras, así que tiene interés para RGPD y seguridad.

Paso 2: Rellenar "Asset/Process" y "Data Category"

Miramos cada parte importante del código y qué datos toca:

- Registro de usuarios (registrar_usuario): trata nombre, email, contraseña.
- Pedido (crear_pedido): asocia email/nombre con productos, cantidades y total.
- Listado de pedidos (mostrar_pedidos): enseña quién ha pedido qué.

En la plantilla se anota, para cada riesgo que definamos:

- Asset/Process: por ejemplo, "Registro de usuarios", "Gestión de pedidos".
- Data Category: "Identificativos (nombre, email)", "Credenciales (contraseña)", "Datos de compras".

Paso 3: Detectar amenazas y vulnerabilidades

Con el programa delante, se preguntan cosas muy básicas:

- ¿Qué podría salir mal? (amenazas, "Threat")
- ¿Qué fallo del diseño/código lo permitiría? (vulnerabilidad, "Vulnerability")

Ejemplos concretos que salen de leer el código:

- Contraseñas en texto claro en la lista usuarios.
- Se usa solo el email para encontrar al usuario; no se comprueba contraseña al hacer pedido.
- Los pedidos muestran nombre y producto, lo que revela hábitos de compra si se filtran.

De ahí salen riesgos como:

- "Robo de contraseñas / acceso a credenciales" (Threat) apoyado en "contraseñas en texto plano" (Vulnerability).
- "Suplantación de usuario para hacer pedidos" apoyado en "identificación solo por email sin autenticación real en crear_pedido".

Paso 4: Valorar Impact, Likelihood y Risk Level

Para cada riesgo, se valora:

- Impact: qué pasa si se materializa (daño para las personas o la organización).
- Likelihood: qué probabilidad hay que ocurra con el programa tal cual.
- Con esas dos cosas se fija un Risk Level (bajo/medio/alto).

Por ejemplo:

- Contraseñas en texto plano → Impacto alto (pueden usarlas en otros sitios), Probabilidad media/alta (basta con ver la lista) → Riesgo alto.
- Pérdida de pedidos (todo en memoria) → Impacto medio (pierdes histórico),
- Probabilidad alta (cerrar programa los borra) → Riesgo medio/alto.

Paso 5: Controles existentes y recomendados

Luego se mira qué protección hay ya en el código:

- Aquí prácticamente no hay controles (no hay hash, no hay autenticación para pedidos, no hay backups), así que "Existing Controls" suele estar casi vacío.
- En "Recommended Controls" se apuntan medidas básicas que sabes que faltan:
    - Hash de contraseñas, autenticación real antes de pedir, base de datos persistente, política de privacidad, etc.

Paso 6: Completar la plantilla con 2–3 riesgos claros

Con todo lo anterior, ahora sí se pueden rellenar filas. Por ejemplo, para este programa, algo como:

Risk ID	Asset/Process	Data Category	Threat	Vulnerability	Impact	Likelihood	Risk Level	Existing Controls	Recommended Controls	Responsible Person	Deadline
R1	Registro de usuarios	Credenciales (email, contraseña), identificativos	Robo de contraseñas y acceso a otras cuentas	Contraseñas guardadas en texto claro en usuarios	Alto – se pueden reutilizar en otros servicios	Media/Alta	Alto	Ninguno	Guardar solo hash de contraseñas; restringir acceso a la lista de usuarios	Responsable de desarrollo	fecha objetivo
R2	Creación de pedidos (crear_pedido)	Identificativos y datos de compra	Suplantación de usuario para hacer pedidos	Pedido asociado solo por email, sin autenticación real	Medio – pedidos a nombre de otro usuario	Media	Medio/Alto	Ninguno	Exigir login con contraseña antes de pedir; validar usuario autenticado	Responsable de desarrollo	fecha objetivo
R3	Lista pedidos en memoria	Datos de compra vinculados a email/nombre	Pérdida total del historial de pedidos	Datos solo en memoria, sin persistencia ni backups	Medio – pérdida de histórico y reclamaciones difíciles	Alta	Medio/Alto	Ninguno	Usar base de datos persistente; copias de seguridad periódicas	Responsable de TI	fecha objetivo

La lógica con la que se llega ahí es siempre la misma:

1. Entender qué hace el programa y qué datos mueve.
2. Nombrar los activos/procesos y categorías de datos.
3. Imaginar escenarios de “qué puede ir mal” y ver qué fallo del código lo permite.
4. Ponerles impacto y probabilidad sencillos.
5. Anotar qué ya existe y qué habría que añadir para reducir el riesgo.

Pedido asociado solo por email, sin autenticación real