**SecDevOps**

**1. Idea de la aplicación**

Página web para vender productos de ropa. Incluye sección de tienda, portal de usuario y carrito de la compra, pero no el proceso de pago.

**2. Tecnología de la aplicación**

Esta aplicación web se desarrolla en Visual Studio Code, utilizando lenguajes HTML, CSS y JavaScript.

**3. Vulnerabilidad a subsanar**

La página web redirecciona a una página de login al hacer clic en “portal de usuario”, aceptando todo tipo de redireccionamientos. Spammers y phishers utilizan los redireccionamientos abiertos de un sitio web para redireccionar a los usuarios desde la web original a su página web maliciosa, lo cual supone un problema de seguridad y confianza en los usuarios de tu sitio web.

Esta es la parte de mi código que redirecciona al portal de login.

Existe un botón en la sección de Portal de Usuario que no verifica el redireccionamiento.

Texto

Descripción generada automáticamente

Posteriormente, el botón redirecciona a una URL absoluta e independiente del sitio web.

Texto

Descripción generada automáticamente

Antes de subsanar esta vulnerabilidad, si un usuario hace clic en iniciar sesión dentro del Portal de Usuario, te redirige a esta página web.

Interfaz de usuario gráfica

Descripción generada automáticamente

**4. Vulnerabilidad ya subsanada**

Hay dos maneras de subsanar esta vulnerabilidad.

La primera opción es utilizar URLs relativos. En la sección del portal de usuario se puede incluir un enlace relativo que redirija a la página de inicio de sesión (/login).

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

La segunda opción, y la que yo he utilizado en mi código, es validar los redireccionamientos. Antes de redireccionar a una página, se realiza una validación para asegurarse de que la URL de destino es segura y está permitida.

La función esURLSegura verifica si el path de la URL cominenza por "/" (ruta relativa), "./" (ruta relativa en el mismo nivel de directorio) o "../" (ruta relativa hacia arriba en el árbol de directorios). Cualquier otro URL será considerado inseguro y se bloqueará el acceso. Además, para aumentar la seguridad del sitio web, se puede incluir una lista de paths permitidos, que no permite que el sitio web redireccione a otros paths, aunque estén dentro del propio sitio web. Si cumple los requisitos, el sitio web redireccionará a la URL de destino. Si no es así, mostrará una alerta de URL no segura.

**Texto

Descripción generada automáticamente**

Aquí se define la función mostrarMensajeNoSeguro, la cual impide que el usuario entre a la URL de redireccionamiento y muestra por pantalla que el sitio web al que accedes no es seguro.

**Texto

Descripción generada automáticamente**

Para demostrar que las condiciones funcionan, he creado cinco páginas. Index.html tiene la constante urlDestino definida como login.html, Index2.html como https://www.ejemplo.com/login e Index3.html como producto.html. Además he creado un formulario básico de inicio de sesión en login.html y la página producto.html para realizar pruebas, que tiene el mismo código que login.html.

Si accedemos al sitio web en index.html y hacemos clic en iniciar sesión, nos redirige a la página de inicio de sesión login.html, ya que este path es el único permitido.

Interfaz de usuario gráfica

Descripción generada automáticamente

Si, por el contrario, accedemos al sitio web con index2.html o index3.html aparece un mensaje de error ya que los redireccionamientos no son seguros.

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**5. Subida del código a un repositorio de github nuevo (no git101)**

https://github.com/marteag/pag-web

**6. Subida del documento con capturas de pantalla del proceso**

https://github.com/marteag/pag-web