**V2**: Authentication Verification Requirements

*V2.1.1: Passwords con mínimo de 12 caracteres*

*V2.1.2: Password con 64 o más caracteres son permitidas*

*V2.1.3: Passwords con espacios no son permitidas*

*V2.1.5: Passwords con caracteres ascii como emoticonos no son permitidas*

*V2.1.6: La funcionalidad de cambio de password requiere el usuario actual y la nueva password*

*V2.1.7: Chequear que la password ingresada no sea una de las comunes ya violadas, como 123456 o cosas por el estilo. Podemos chequearlo contra una lista interna nuestra o usar una api externa. Lo mejor y más fácil es que tengamos una lista interna, y ver que la password ingresada en el registro, o en el cambio de password no esté en esa lista.*

*V2.1.8: Tener un medidor de seguridad para verificar que la password sea fuerte. Esos que te obligan a agregar mayúsculas, minúsculas, caracteres especiales, y van pintando de rojo, amarillo y verde. Ver si hay algo ya hecho para copypastear de algún lado.*

*V2.1.9: No tener reglas de composición de contraseñas con caracteres permitidos. Por ejemplo no limitar la cantidad de números, o de mayúsculas o minúsculas o caracteres especiales.*

*V2.1.10: Que no haya requisitos de cambio de password periódico o impedimentos de contraseñas históricas.*

*V2.1.11: Verificar que funcione el “pegar”, los ayudantes de contraseña del navegador, y se permitan administradores externos de contraseñas.*

*V2.1.12: Verificar que el usuario pueda ver la contraseña escrita (con el ojito), o poder ver el ultimo carácter escrito antes que pase a ser un punto o asterisco.*

*V2.2.1: Verificar que haya controles de anti automatización. Esto es que intenten con fuerza bruta adivinar una password. Lo que habría que hacer acá es meter un captcha, e ir acumulando cuantos intentos de logueo fue haciendo erróneos, y si le pifió 5 veces seguidas por ejemplo, bloquearle la cuenta haciendo que no pueda loguearse hasta dentro de media hora por ejemplo. Lo otro mas complejo para hacer pero no se si vale la pena es bloquear la IP pero con lo de bloquear la cuenta para que no pueda loguearse en un tiempo alcanzaría.*

*V2.2.5: Verificar que cuando tengamos separados un proveedor de servicio de credenciales (CSP) y la verificación de la password de la aplicación, haya en el medio de los 2 endpoint un TLS. (Este no aplica)*

*V2.4.1: Guardar las passwords con hash y salt.*

*V2.4.2: El SALT que usemos, debe ser único por cada password que guardemos, y tiene que medir al menos 4 caracteres. Podemos tomar los primers 4 o últimos 4 caracteres dela password sin hashear y usar eso como salt. Es importante que se haya elegido un algoritmo de encriptación con salt, como por ejemplo AES256.*

*V2.4.3: Verificar que si usamos el algoritmo* PBKDF2, la cantidad de iteraciones para encriptar sea el máximo que permita el servidor, que suele ser al menos 100.000 iteraciones.

*V2.4.4: Verificar que si usamos el algoritmo* bcrypt, el factor que pongamos sea lo que el servidor se aguante. Tipicamente es 13.

*V2.4.5: Verificar que se haga al menos una iteración adicional en el algoritmo de encriptación, que el salt sea secreto y que sea generado por un generador de bits aleatorio aprobado y propocione la fuerza de seguridad minima. El salt se debe almacenar por separado de las contraseñas encriptadas, por ejemplo en un dispositivo tipo un modulo de seguridad (hardware).*

*V2.5.1: Verificar que en el recupero de clave, la clave no sea enviada en texto al usuario.*

*V2.5.2: Verificar que el recupero de claves basado en conocimiento o preguntas secretas no esté presente.*

*V2.5.3: Verificar que el recupero de clave no revele la contraseña actual de ninguna forma.*

*V2.5.4: Verificar que las cuentas compartidas o default no estén presentes. Por ejemplo root, admin, etc. Esto lo podemos meter en la lista de uno de los puntos de arriba, tipo claves conocidas no válidas.*

*V2.5.5: Verificar que si se reemplaza o cambia un factor de autorización, por ejemplo agregando algo más aparte del email y contraseña, el usuario sea notificado.*

● **V5**: Validation, Sanitization and Encoding Verification Requirements

*V5.1: Verificar que la aplicación tenga defensas frente a ataques del tipo de contaminación de parámetros HTTP, particularmente si el framework de la aplicación no hace distinción sobre el origen de los parámetros que llegan por get, post, etc.*

*V5.2: Verificar que el framework proteja contra ataques de asignación masiva de parámetros, como campos privados o similar.*

*V5.3: Verificar que todos los inputs sean validados usando validación positiva (lista blanca)*

*V5.4: Verificar que todos los datos sean verificados con los formatos correspondientes, por ejemplo caracteres, largo, y patrones. Como por ejemplo, si fuese tarjeta de crédito, los números validos, etc. En nuestro caso aplicaría que sea un email valido el email ingresado.*

*V5.5: Verificar que la url envíe solo a destinos que tengamos en nuestra lista blanca, y mostrar advertencias cuando se está dirigiendo contenido no autorizado. Por ejemplo si nos cambian la url, o si nos agregan cosas a nuestro post.*

● **V7**: Error Handling and Logging Verification Requirements

*V7.1: Verificar que la aplicación no loguee credenciales o detalles de pago. Los tokens de sesión deben ser guardados encriptados.(No usamos tokens )*

*V7.2: Verificar que la aplicación no registre otros datos sensibles definidas bajo normas de privacidad o políticas de seguridad relevante.*

*V7.3: Verificar que los logs incluyan logins satisfactorios y fallidos, fallos de deserialización, o sea cuando nos llegan cosas que no esperamos, y fallos de validación de los inputs de entrada.*

*V7.4: Verificar que los logs incluyan información necesaria que permitan una investigación detallada del lapso de tiempo en que las cosas pasaron.*