Vulnerabilidades en la autenticación

Permiten a los atacantes obtener las credenciales de un usuario de una aplicación cuando:

- La aplicación permite especificar contraseñas poco seguras
- No se limita el número de autenticaciones fallidas
- La contraseña se almacena de forma poco segura
- La contraseña se transmite a través de medios poco seguros

Si un usuario tiene una contraseña poco segura es posible que un ataque de fuerza bruta de diccionario logre averiguarla. Para evitar esto se recomienda las siguientes políticas:

- Longitud mínima de la contraseña
- Aumentar la complejidad de la contraseña combinando distintos tipos de caracteres como letras mayusculas y minúsculas, simbolos y números.
- Validar que la contraseña no esté incluida dentro de una lista de contraseñas conocidas como poco seguras.

También es buena práctica no desvelar que parte de la autenticación es incorrecta. No se recomienda mostrar un mensaje diciendo que el usuario no existe o la contraseña no es correcta. Esto pueda hacer que un atacante pueda deducir nombres de usuario o correos electrónicos, por ello se recomienda usar un mensaje genérico que simplemente diga que la autenticación ha fallado.

Por otro lado, las contraseñas NUNCA se deberían de almacenar en texto plano ya que si un atacante puede acceder a las bases de datos internas puede hacerse fácilmente con las contraseñas como fue el caso del hackeo a playstation network. Facebook fue multada con 91 Millones de euros por almacenar las contraseñas en texto plano.

Hash

Para no almacenar las contraseñas en claro, una tecnica muy típica es la de Hash and Salt. Un hash es un algoritmo matemático que transforma un bloque arbitrario de datos en una nueva serie de caracteres de longitud fija. Este tipo de funciones son unidireccionales y no reversibles, por lo que tratan de garantizar que con los datos de salida no se puedan obtener los de entrada. Estas funciones se usan para:

- Asegurar que un fichero no se ha modificado durante una transmisión
- Firmar digitalmente un documento
- Hacer ilegibles las contraseñas

Los algoritmos de cifrado son revesibles, lo que implica que si un atacante obtiene la clave de cifrado y la base de datos de usuarios, puede obtener las claves. Como las funciones hash no son reversibles, son más recomendadas para el almacenaje de credenciales. A veces se combina cifrado simétrico con una función hash. Los algoritmos más conocidos son MD5, SHA1, SHA2, bcrypt, scrypt y PBKDF2, siendo los 3 últimos los más segirps ya que MD5 y SHA1 son bastante vulnerables. A la hora de procesar una contraseña se busca que el algoritmo hash sea lo más lento posible para alargar el tiempo necesario por un atacante durante un ataque.

Las funciones hash son vulnerables a la técnica de rainbow tables que consiste en usar diccionarios de hash para hacer un ataque. Para mitigar estos ataques se combinan los hash con fragmentos aleatorios (Salt) que se concatenan a la contraseña para generar siempre un hash difícil de predecir:

1. Antes de almacenar la contraseña se concatena con un fragmento aleatorio
2. La concatenación resultante se procesa con la función hash
3. Se almacenan tanto el hash como el salt.

Durante el proceso de verificación de contraseña se hace lo siguiente:

1. Se obtiene el salt usado
2. Se concatena la contraseña y el salt y se les aplica la función hash
3. Se compara el hash con el valor almacenado en la base de datos.

PBKDF2

Permite especificar un salt y también el número de iteraciones que se realizan para calcular el hash final.

Bcrypt

Diseñado para ralentizar los ataques de fuerza bruta. Incluye un valor de trabajo configurable (round) que cuanto mayor sea, mayor tiempo tardará el algoritmo en calcular el hash. En la mayoría de sus implementaciones añade un salt aleatorio de forma automática.