Tarea 3 criptografía

José Lara Hinojosa

Mayo 2021

Pregunta 1

a)

Posibles problemas de la autenticación

- Confiamos en los servidores y en la gente de dentro de estos
  Un problema es que si bien se puede tener seguridad en el trayecto de la información con encriptaciones y seguridad por https, la password como tal se puede leer en el servidor en el que se esta autenticando el usuario, por lo tanto, una persona mal intencionada con poder en el servidor podría saber la contraseña de un usuario antes de que esta se guarde en la base de datos como un hash, logrando así poder acceder a múltiples plataformas del cliente si es que este tenia una sola password para muchos servicios, en concreto el mail, que es el central para recuperar contraseñas, escalando a que el usuario podría llegar a perder cuentas bancarias, redes sociales, cuentas de universidad, etc.
  Otro ejemplo sería que un banco podría realizar transacciones al saber nuestra información y nosotros no podríamos defendernos ante esto, ya que ellos tienen todas las de ganar al tener toda nuestra información y controlar los logs de sus bds.

- Guardar las contraseñas sin hashear
  Cabe destacar que en el caso anterior existe seguridad a que los empleados al ver la base de datos no pueden saber cual es la contraseña, al estar hasheada, sin embargo, si estas se guardan sin hashear, todos los empleados con acceso a la bd podrían conocer las contraseñas de sus clientes, por lo que estaríamos confiando en estas personas (auditabilidad), y si se llegaran a filtrar estas bd, sería catastrófico, como lo que siempre pasa cuando se filtran correos electrónicos o contraseñas ya que se estaban guardando como texto plano y son llegar y usar para un malintencionado.

- Rainbow tables, problema de hashear sin salt
  De lo mismo anterior, también teníamos el problema de guardar las contraseñas como hash tal cual, sin añadir un numero random o salt, ya que al filtrarse en un caso

hipotético la base de datos de este servidor, un malintencionado podría tener pre-computadas tablas de hash (rainbow tables), viendo si alguna de la base de datos calza con alguna de estas tablas pre-computadas llegando a conocer la contraseña de algunas personas de la bd. También existe otro problema, si otros servicios también filtraron sus bd, el atacante podría revisar en estos y comparar los hashes del password para el usuario, sabiendo que tiene la misma contraseña en ambos servicios.
Esto se soluciona hasheando la password más un salt, así el atacante debería generar demasiadas posibilidades por cada posible password, haciéndolo muchísimo más difícil (si tiene acceso al salt entonces debería precomputar toda su bd de contraseñas posibles para cada usuario).
Si lo anterior no es suficiente se puede ocupar pbkdf2, hasheando muchas veces el hash con salt, haciéndolo casi imposible para el atacante.

Posibles problemas de la autorización

- Cookies y CSRF
  Los sitios almacenan sus tokens de autenticación en cookies del navegador, las cuales siempre se envían en los requests a sus paginas, por lo que, si una pagina hace request a otra pagina, en este caso un banco, y el usuario estaba logueado en ese banco, esto causaría que la pagina pudiera hacer request autorizados y sin permiso del usuario por atrás si es que lo quisiera, por ejemplo, transferirse mucho dinero de la cuenta del usuario.
  Otro problema es que el servidor debe almacenar todos estos tokens.

b) Solución al sistema usuario contraseña actual

Un diseño que solucionaría todos los problemas presentados sería que los usuarios tuvieran claves privadas y claves publicas guardadas en algún lugar seguro, además de una clave maestra para guardarse encriptadas y así añadir más seguridad, de tal forma que al loguearse en un sitio no se ocupe una contraseña como tal, si no que se ocupe usuario y clave publica para ese sitio más una firma delegada al navegador (el usuario pondría el certificado) y así, nadie podría falsificar la identidad del cliente, ni siquiera el servidor, ya que solo tendría su clave publica en la base de datos y el usuario.
Los problemas prácticos de esto es que estamos forzando una usabilidad mediocre al usuario a cambio de seguridad, ahora tendría que ocupar un gestor de contraseñas amigable el cual debe ser confiable y no guarda ninguno de nuestros inputs, solo las cosas encriptadas, en especial, los comandos en los que ponemos nuestra clave maestra para decriptar. También esta el problema de que pasa cuando queremos loguearnos en otros dispositivos, aquí la usabilidad sería aun peor, teniendo que conectar el celular para pasar estas contraseñas y desencriptarlas y todo el atado. También esta el problema de las aplicaciones actuales, todas deberían modificar sus sistemas de autenticación permitiendo este tipo de autenticación.
Para solucionar estos problemas se debería tener una aplicación confiable que no guarde nuestro input de clave maestra y almacene nuestras claves encriptadas, esta debe tener ver-

sión de android/ios y ser muy fácil de sincronizar con nuestro computador. Esto haría más amigable la experiencia. Para solucionar el otro problema es más difícil, ya que requeriría hacer un cambio en la mayoría de aplicaciones, y dependería más de la gente, pero podría ser un nuevo tipo de autenticación, no eliminando la por usuario contraseña y permitir esta de clave privada publica como opción, y siempre que se tenga activada la de privada publica verificar las firmas de las acciones y deshabilitar la usuario contraseña (así no forzamos a la gente que no quiere a tener este tipo de seguridad). Considerando los problemas de autorización el token podría reemplazarse por una firma del servidor con clave publica/privada, de tal manera que no tenga que guardar tokens, si no que revisa que viene del usuario y si esta bien firmado por el. También para los otros problemas como request de otros sitios pueden ocupar las cookies $SAMESITE$ y $HTTP_ONLY$.