Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC3253 — Criptografía y Seguridad Computacional 2021-1

NOMBRE: Matías Duhalde

Tarea 3

Pregunta 1

Parte a)

Las malas consecuencias de utilizar este tipo de autenticación pueden provenir de dos fuentes (asumiendo que el canal de comunicación es seguro), el mismo usuario que está creando y usando estas credenciales, y el servicio sobre al cual se accede:

- **Usuarios**: El problema principal va ligado en parte a la naturaleza humana. No somos buenos para recordar una gran cantidad de datos aleatorios, por lo que tendemos a utilizar contraseñas que tienen problemas de seguridad, como los siguientes:
  - No son completamente aleatorias sobre todo el espacio de caracteres. Es difícil recordar una secuencia de caracteres totalmente aleatoria, en la que sus elementos no tienen ninguna correlación entre sí. Se suele optar por contraseñas débiles que por lo general constan de sólo letras y números. También las letras generalmente forman palabras del idioma del usuario, por lo que se reduce aún más el espacio de posibilidades. En la práctica, al intentar obtener una clave por fuerza bruta, es mucho más probable que llegaremos antes a una contraseña correcta si concentramos los caracteres probados en ciertas combinaciones de palabras, a diferencia de probar todos los caracteres posibles.
  - Las contraseñas tienden a ser cortas, debido a que es difícil memorizar strings largos. Por lo tanto, en un ataque de fuerza bruta, se lograría romper mucho más rápido una contraseña corta que una muy larga.
  - Dado que en el último tiempo ha aumentado bastante la cantidad de servicios en línea que se acceden y usan este tipo de autenticación, es común repetir contraseñas entre estos sitios, para no tener que almacenar/recordar una contraseña distinta para cada servicio. Un ejemplo que expondría esta vulnerabilidad sería el caso de que algún servicio es *hackeado* y atacantes obtienen las contraseñas, se prueban los mismos nombres de usuarios y contraseñas en otros servicios que puedan estar usando los usuarios expuestos, para poder acceder a sus cuentas.
  - Si no se recurre a la memoria, entonces las contraseñas se almacenan (en texto plano) en otro lugar físico, como un cuaderno o un *post-it*, elementos que pueden encontrarse expuestos a ser vistos y abusados por otras personas.
- **Servicios**: Pueden incurrir en prácticas pobres y vulnerables para el manejo de credenciale:
  - Guardar credenciales en texto plano. Al registrarnos en una aplicación, como usuarios generalmente no tenemos cómo saber las prácticas que usa el servicio al que estamos accediendo para proteger nuestros datos, por lo que como usuario debemos confiar en que se están manejando de manera segura. No usar ningún algoritmo para ocultar las credenciales, deja expuestos los datos del usuario, permitiendo a cualquier persona con acceso al almacenamiento obtener contraseñas.
  - Usar encriptación en lugar de hashing. Debido a que las funciones de encriptación no son *one-way*, existiría forma de obtener la contraseña a partir del string almacenado, dando lugar a una vulnerabilidad. Por ejemplo, un empleado del servicio con suficiente conocimiento de la llave podría acceder a las contraseñas de los usuarios.

– Usar algoritmos que en la actualidad se consideran inseguros. Por ejemplo, si un servicio online, como un banco, tiene almacenadas las contraseñas con una función de hash considerada insegura, como `md5`, si se filtra la base de datos, le será mucho más fácil a los atacantes calcular los hashes y romper contraseñas. Además, si no se utiliza *salting*, las contraseñas quedarían muy expuestas a *rainbow table attacks*.

Parte b)

Una solución conversada en clases que solucionaría algunos problemas que tiene la autenticación por medio de *username/password*, es el uso de firmas digitales. También, la idea propuesta no busca reemplazar completamente la autenticación por nombre de usuario + contraseña, sino que complementar los dos sistemas entre sí.

Para implementar **autenticación** por firmas digitales, se debe usar criptografía asimétrica. Así, el usuario posee una llave pública y una privada. La llave pública se comparte con el servicio durante el proceso de registro, para que sus servidores sean capaces de verificar la firma.

Un usuario para poder *logearse* en la página, debe mandar un certificado al servidor, el cual es **encriptado con la llave privada** (junto con el nombre de usuario o alguna credencial que lo identifique), de tal manera que el servidor pueda **desencriptar el certificado con la llave pública** almacenada del usuario. Este certificado sólo podría ser generado por el usuario correspondiente, debido a que en la práctica es el único que conoce la llave privada.

Para el proceso de **autorización**, se puede usar el mismo sistema, pero incluyendo dentro del certificado una token temporal.

El método anterior permitiría eliminar todas las consecuencias negativas causadas por "errores humanos", debido a que no se debería crear ni recordar una contraseña, sino que se debe almacenar la llave privada, la cual es autogenerada por el algoritmo escogido, como *RSA* o *Diffie-Hellman*.

También, se eliminarían las consecuencias negativas relacionadas a los servicios, debido a que la seguridad del sistema recae en la llave privada, que es almacenada por el usuario, no los servidores. Si se fuera a filtrar de alguna manera la llave pública que guarda el servicio, no habría un problema grave, debido a que no permitiría a terceros generar nuevos certificados, sino que solamente desencriptarlos.

El problema principal que traería consigo este sistema, es que la llave privada quedaría "ligada" al dispositivo sobre el cual se creo, debido a que generalmente son extensas, y no pueden recordarse fácilmente como una contraseña, que se usa libremente entre dispositivos. Una solución a esto sería traspasar esta llave privada entre dispositivos como texto plano o un texto encriptado con otra llave conocida por el usuario, pero esto ya tiene asociado otros riesgos, como el robo de la llave. Para este mismo punto, también podrían configurarse dentro del servicio, más de un par de llave pública/privada por cada usuario, permitiendo que el usuario genere un par en otro dispositivo, y le traspase la llave pública al servidor (usando un canal seguro establecido por el par existente), y que este al momento de recibir un certificado de autenticación, intente desencriptarlo con todas las llaves que le correspondan al usuario.

También, existe la posibilidad de que un tercero intercepte los certificados enviados desde el cliente hasta el servidor, y los "clone" para hacerse pasar por el usuario. El servidor al recibir estos certificados clonados, podría desencriptarlos satisfactoriamente con la llave pública que posee. Para arreglar esto, se puede complementar el certificado con un *timestamp* y *HMAC*, que sólo permita realizar la acción asociada al certificado una vez, y dentro del marco de tiempo indicado. De esta manera, un tercero no podría reutilizar con el servicio un certificado interceptado.