CMP0565 - Reporte Proyecto Final: ChatApp

Christian Sánchez, Esteban Flores Andrade

Universidad San Francisco de Quito (USFQ)
Fecha entrega: 20 de Diciembre de 2017

El protocolo

El protocolo implementado tuvo cambios con respecto al protocolo propuesto inicialmente. Sobre las especificaciones técnicas, se cambió el tipo de hash de `PBKDF2` a `SHA-256` y el tamaño de llaves de la encriptación simétrica de `AES-256` a `AES-128`. Así, las especificaciones técnicas del protocolo implementado fueron:

- `RSA-2048`
- `AES-128`, específico: `AES/CBC/PKCS5Padding`
- Tipo de hash: `SHA-256`

Sobre la forma en que se guardaron las credenciales de usuario en la application, $P' = h\{h\{pwd\} + username\}$. En este caso, el símbolo " + " significa concatenación.

A continuación se presentan los cambios en cada paso del protocolo.

Login

Sobre el Login se realizaron los siguientes cambios:

- Se decidió no guardar en texto plano el nombre de usuario en el archivo de credenciales del servidor. En el protocolo implementado, el nombre de usuario se guarda de la forma: $h\{username\}$.
- Los cambios entre el protocolo de login propuesto y el implementado se pueden observar en las figuras 1 y 2.

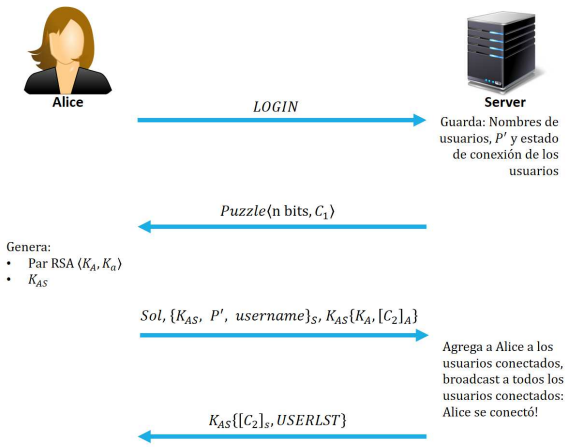

Figura 1: Protocolo de login (propuesto).

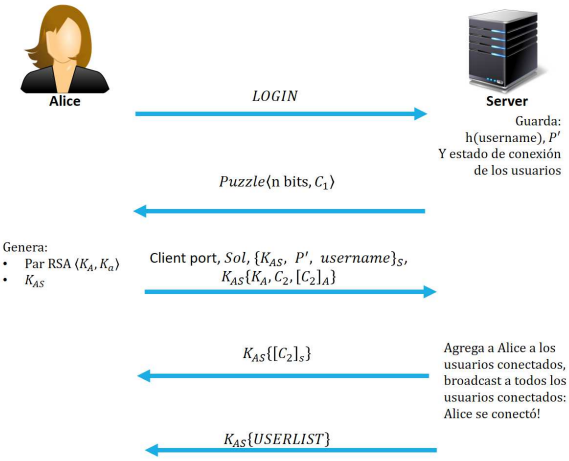

Figura 2: Protocolo de login (implementado).

Actualización de lista de usuarios conectados

Los cambios en la actualización de lista de usuarios conectados entre el protocolo propuesto y el implementado se pueden observar en las figuras 3 y 4.

Figura 3: Protocolo de actualización de lista de usuarios conectados (propuesto).

Figura 4: Protocolo de actualización de lista de usuarios conectados (implementado).

Nota importante: El cambio sucede porque existe un error en la implementación de actualización de usuario, esto debería ser mandado encriptado con la clave entre servidor y el usuario que la vaya a recibir, pero en ejecución, aunque se recibe la misma cadena de bytes que se envió no es posible desencriptar. Por ello el cambio entre el protocolo propuesto y el protocolo implementado.

Logout

El protocolo de logout no presentó cambios entre lo propuesto y lo implementado. El protocolo de logout se muestra en la figura 5.

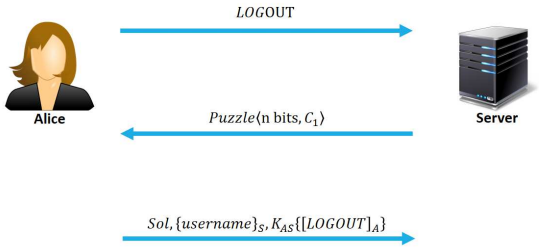

Figura 5: Protocolo de logout.

Solicitud de datos de otro usuario (ticket)

Los cambios en la solicitud de ticket entre el protocolo propuesto y el implementado se pueden observar en las figuras 6 y 7.

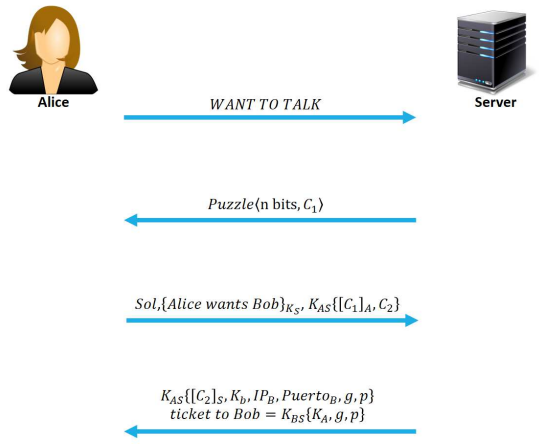

Figura 6: Protocolo de solicitud de ticket (propuesto).

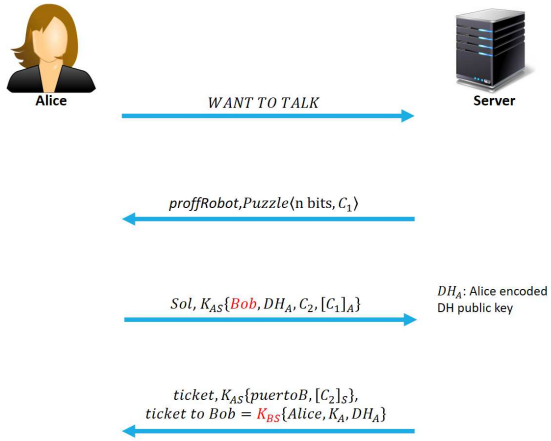

Figura 7: Protocolo de solicitud de ticket (implementado).

Comunicación entre usuarios

Los cambios en la comunicación entre usuarios entre el protocolo propuesto y el implementado se pueden observar en las figuras 8 y 9.

Nota importante: Para la conexión entre usuarios se pensó que el usuario no debería escuchar sin que alguien no quiera hablar con él, esto produce un error ya que cuando vamos a hablar entre el usuario A y B vemos que B tarda más en iniciar su puerto que A en enviar un mensaje lo cual produce un error al momento de tratar de iniciar `Diffie-Hellman`, pero una vez que B se pone a escuchar se pueden intercambiar mensajes sin problema entre los usuarios.

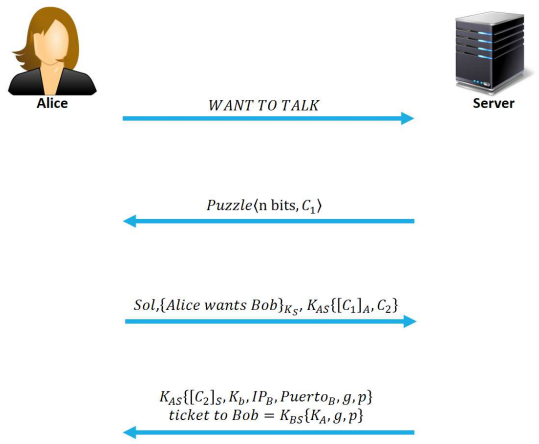

Figura 8: Protocolo de comunicación (propuesto).

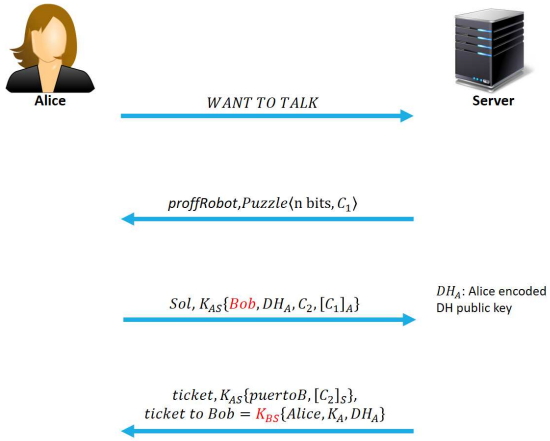

Figura 9: Protocolo de comunicación (implementado).

ATAQUES

En la siguiente sección se describen ataques que se encontraron sobre el protocolo implementado.

Login

Desde nuestra perspectiva, no logramos encontrar ataques conocidos considerables en el protocolo de login del programa. Sin embargo, creemos que los siguientes ataques (pequeños) pueden ser posibles.

-1. ***DOS en POW:*** Dado la forma de generar números primos para enviar a los clientes un `nonce`, puede ser posible inundar al servidor con solicitudes de números primos (solicitud LOGIN). Puede suceder que el servidor tarde más tiempo en generar números primos para la gran cantidad de solicitudes de login que el tiempo en que tarde en recibir estas solicitudes. En este caso, un ataque `DOS` es factible. Esto se repite para los protocolos de las figuras 5 y 7.

Solución: Implementar un mecanismo de generación de números primos eficiente. Que no tome una gran cantidad de poder de cómputo.

-2. ***Dificultad de POW:*** Ya que en el protocolo implementado la dificultad del `proof-of-work` esta "quemada.[en] código, el programa puede encontrar en un futuro problemas al tratar que los atacantes resuelvan un `POW` pues este, con el tiempo, puede llegar a ser débil.

Solución: Cargar la dificultad del `POW` desde un archivo de preferencias en el servidor.

-3. ***Ataque off-line en login:*** En el protocolo implementado no se define requerimientos mínimos para contraseñas. Por ello, si existe un usuario de alto perfil y un atacante que ha logrado descargar el archivo de credenciales del servidor y desea obtener la contraseña de este usuario alto perfil, puede intentar realizar hashes con el nombre de usuarios y las contraseñas más comunes (incluso algún diccionario) y comprobar el valor de `P'`. Este sería un ataque `off-line` que tendría sentido solo si se desea obtener credenciales de usuarios de alto perfil.

Solución: Definir requerimientos de contraseñas, como mínimo de caracteres, combinación de caracteres y al mismo tiempo limitar al usuario a crear contraseñas que no estén definidas en un diccionario.

-4. ***Contraseña usuario-servidor:*** Ya que la contraseña usuario-servidor es definida por la máquina del usuario, puede ser posible que un atacante modifique la funcionalidad de varias terminales de clientes, haciendo que estos definan siempre la misma clave usuario-servidor. Si un atacante logra infectar varias máquinas, todo el cifrado de información del sistema que se envíen con la clave usuario-servidor serán inútiles.

Solución: Modificar el protocolo para que la clave usuario-servidor la cree el servidor siempre que se inicie una instancia de un cliente.

Actualización de lista de usuarios conectados

-5. ***Enviar lista de usuario falsa:*** Trudy puede dar un nuevo elemento a la lista de usuarios disponible de un usuario A conectado y autentificado, pero al momento de querer hablar con dicha persona si no existe también en el servidor no se enviará mensajes lo cual nos evitaría hacer trabajo en vano para iniciar una conversación que no podría generarse. Además, la lista puede llegar al número máximo de registros gracias a este error y solo se limpiaría iniciando una nueva sesión

Solución: Buscar el error a más profundidad ya que la estructura para la transmisión de información protegida atravesó de la clave entre servidor y el usuario; esta implementado y trabaja para mandar la lista inicial de usuarios conectados antes de la persona que se conectó, pero para esta etapa en particular había problemas al tratar de descifrar.

Logout

Se cree que no existen ataques, que no hayan sido descritos anteriormente en esta sección, en el protocolo de logout. Sin embargo, se ha encontrado un detalle que puede ser optimizado. En el paso 3 de la figura 5, se puede evitar utilizar

cifrado asimétrico incluyendo el `username` en el bloque de cifrado simétrico.

Solicitud de datos de otro usuario (ticket)

En este protocolo también se cree que no existen ataques considerables. Sin embargo el procolo puede ser mejorado. La idea es que las generación de claves de `DH` se realicen en el servidor y no en el cliente. Esto debido a que la clave que generará Bob tendrá los mismos parámetros públicos que la llave de Alice. No tiene sentido para el atacante generar claves no seguras pues si ya tiene acceso a las credenciales y terminal de Alice, ya habrá obtenido las comunicaciones entre Alice y otros usuarios.

Comunicación entre usuarios

-6. ***Mensajes sin cifrado****:* Ya que se envía mensajes en plano porque el protocolo implementado no incluye `Diffie-Hellman`, se podrían enviar mensajes falsos a un usuario conectado el cual haya establecido ya una conversación con otro.

Soluciones:

- Terminar la implementación de `Diffie-Hellman` una vez que se ha iniciado el intercambio de mensajes y el puerto.
- Implementar una funcionalidad en la que una vez en que el usuario ingrese al sistema y sea autenticado empiece a escuchar en un puerto, así se evita el problema de iniciar este puerto al momento de tratar de hablar con otro usuario. Esto soluciona el problema con la lógica del código al intentar implementar `Diffie-Hellman`.

-7. ***Reflejo en comunicación****:* Ya que en esta parte del protocolo no se utiliza `nonces` ni `timestamps` , los mensajes son sujetos a ataques reflejo. Es decir, Trudy puede capturar un paquete de mensaje y volver a enviarlo haciendo creer al sistema que un mensaje ha sido enviado de nuevo en otro instante de tiempo. Este ataque no se considera crítico pues no afecta los requerimientos fundamentales del sistema.

Solución: Esta vulnerabilidad se podría solucionar si se agrega un `timestamp` a cada mensaje. Así el cliente sabrá que un cierto mensaje fue enviado/recibido en un instante dado. Cualquier reflejo significará una referencia al mensaje anterior enviado en el instante de tiempo marcado en el `timestamp`.