Segurança e Auditoria de Sistemas AS36B-N16

Atividade de Comunicação Autêntica, Íntegra e Confidencial

Ferramentas

- Python 3.8.11;
- Biblioteca de criptografia do Python (cryptography);
- Wireshark;

Objetivos

- Criar uma aplicação de comunicação escrita peer-to-peer (P2P)
- Garantir a confidencialidade, autenticidade e integridade da comunicação
- Utilizar a arquitetura cliente-servidor
- Utilizar Sockets
- Verificar se a política de segurança é satisfeita por meio de um sniffer de rede.
- Garantir que todos os clientes consigam ver a informação apresentada na interface do terminal e que o fluxo TCP observado no Wireshark seja completamente indiscernível
- Usar criptografia simétrica, assimétrica e assinaturas

Procedimentos

Foi usado um código de uma aplicação em Python que estabelece comunicação entre dois clientes utilizando um servidor (nó central) por meio de sockets. A aplicação possui dois arquivos: o Serve.py e o Client.py. O primeiro é o servidor cujo IP para fins de teste é o 127.0.0.1 (seu próprio computador) e opera na porta 5535 (é possível alterar diretamente no script ou tornar a porta selecionável pelo usuário). Já o Cliente.py é utilizado para comunicar-se com o servidor como uma sala de bate-papo. Sugere-se que abra um terminal para cada instância sendo primeiro servidor, segundo o cliente até a etapa de inserir o nome do usuário, ao final em um novo terminal execute um novo cliente, assim quando enviar uma mensagem pelo cliente do Terminal 2 o do Terminal 3 também receberá a mensagem e vice e versa. Os códigos foram escritos de forma a fazer a comunicação entre apenas dois clientes.

Abaixo estão listados comandos e atalhos para observar o fluxo TCP da aplicação utilizada.

- Antes de ligar servidor e cliente inicialize o Wireshark e a captura de pacotes na interface de loopback (em alguns casos pode estar como lo ou 127.0.0.1).
- Abra um terminal para o servidor
- Abra um segundo terminal para o cliente e insira o usuário
- Abra um terceiro terminal para o cliente e insira o usuário
- Envie mensagens de um cliente para o outro.
- Pause a captura de pacotes.
- Na barra de filtros do Wireshark digite "tcp.port == 5535" desta forma apenas os pacotes cuja porta TCP 5535 (porta padrão configurada no arquivo, se você alterar a porta no código fonte deve também alterar a porta aqui) aparecerão na lista de pacotes.
- Selecione um pacote e aperte CTRL + ALT + SHIFT + T

Metodologia e Resultados

Fluxo geral de funcionamento do programa:

- O primeiro cliente envia a chave pública ao servidor e aguarda a conexão do segundo cliente
- O segundo cliente envia sua chave pública ao servidor
- Cliente 1 e cliente 2 trocam suas chaves públicas
- Um dos cliente digita e envia uma mensagem, antes de enviar é feita a troca das chaves simétricas e a assinatura
- O outro cliente recebe a mensagem e verifica a assinatura
- Após essas verificações é garantida a comunicação segura entre clientes

A figura abaixo mostra a troca de mensagem sem criptografia usando dois clientes:

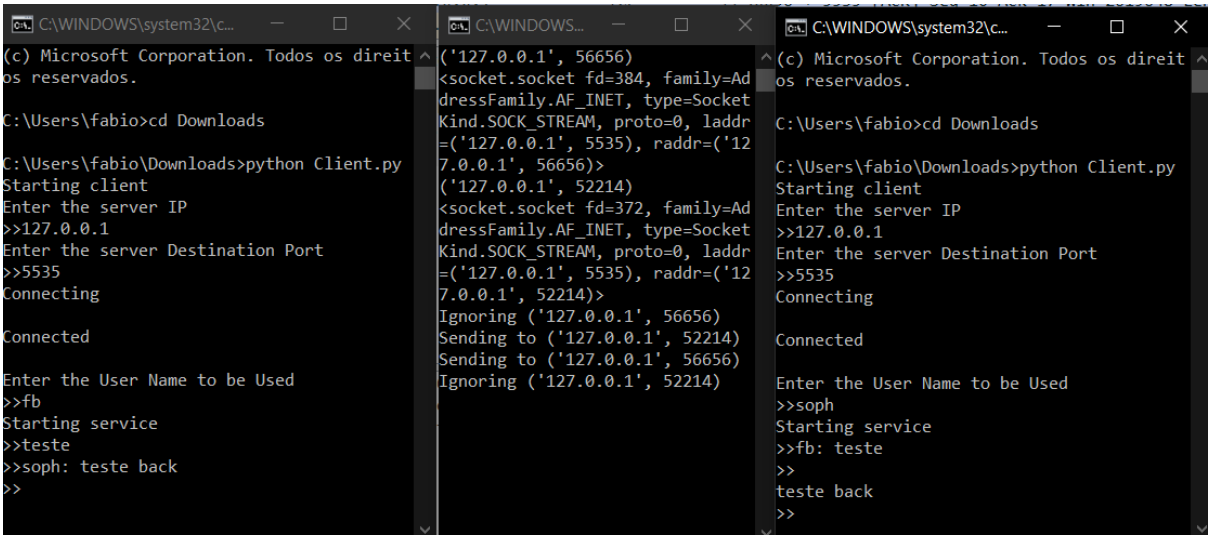

Figura 1

A figura 2 mostra através do wireshark que as mensagens do cliente 1 podem ser lidas pelo sniffer de rede pois não estão criptografadas:

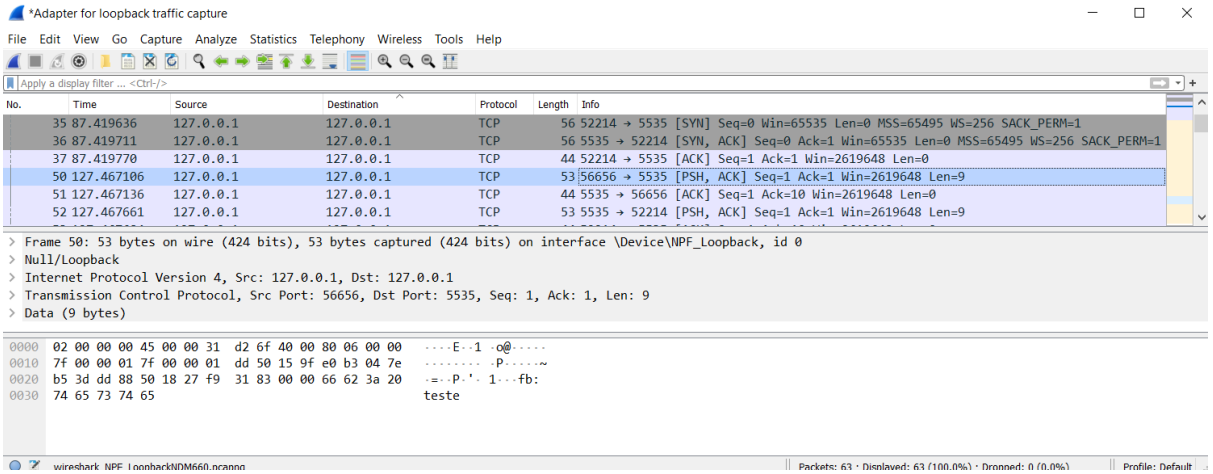

Figura 2

A figura 3 mostra através do wireshark que as mensagens do cliente 2 podem ser lidas pelo sniffer de rede pois não estão criptografadas:

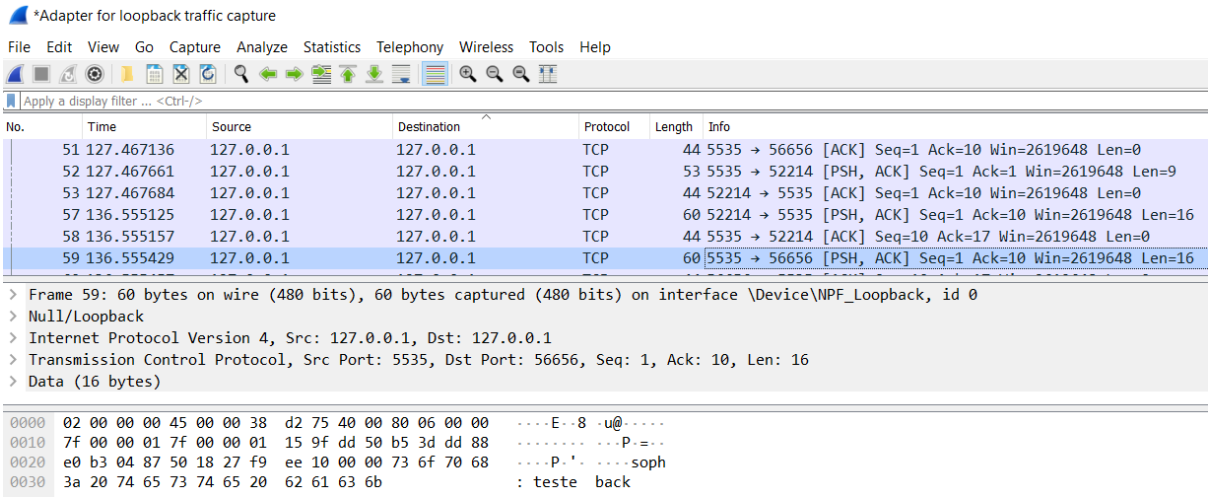

Figura 3

A figura abaixo mostra a troca de mensagem usando dois clientes já com a criptografia implementada:

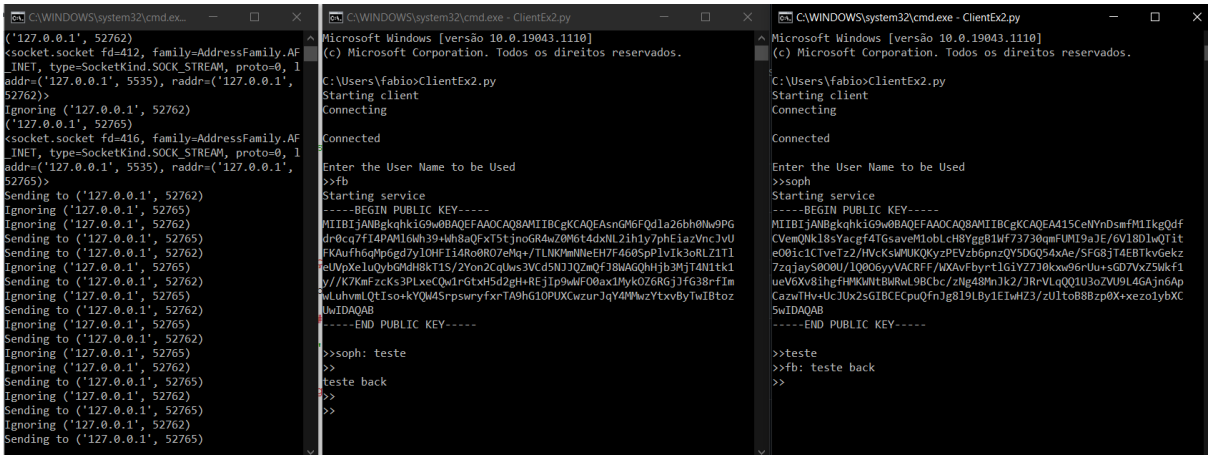

Figura 4

Por último na figura 5 mostra o conteúdo do pacote usando o comando “tcp.port == 5535”:

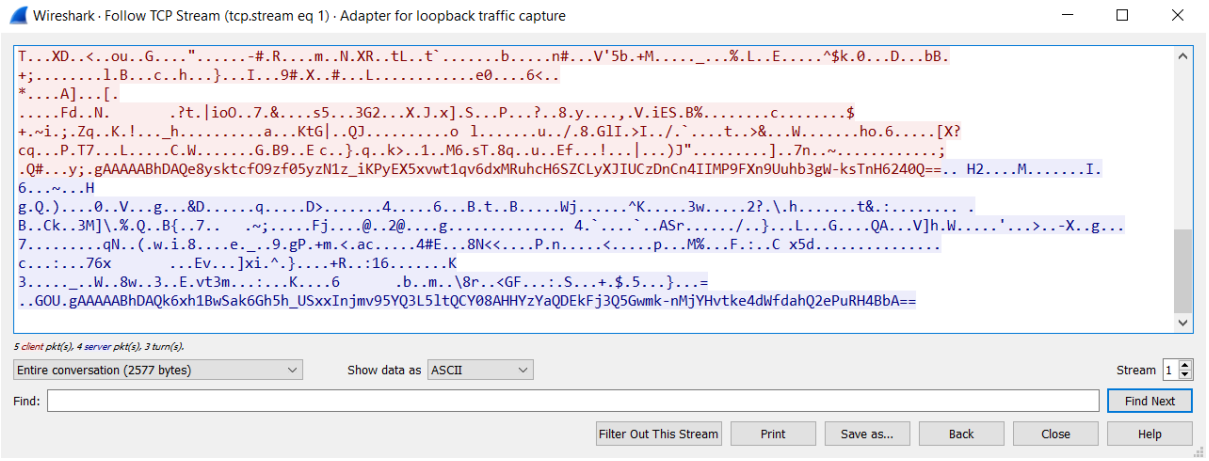

Figura 5

A seguir o fluxo de chamada das funções:

- A main starta a função run() -> chama a função connect passando o host e a porta para conectar no server central
- Instancia a classe Server() -> chama a função initialise passando a variável receive -> chama a função gera_chave_rsa que (gera as chaves públicas, privadas, serializa a chave pública, inicializa variáveis)
- Chama a função run() da classe Server que em um primeiro momento não executa nada até que o segundo cliente se conecte
- Envia a chave pública do remetente para o servidor central
- O while chama a função client() na classe Cliente até o momento que o servidor central envia a chave pública do cliente destinatário para o remetente, que é recebido por meio do primeira condicional da função run() da classe Server
- A primeira condicional da função client() verifica o recebimento da chave pública do destinatário
- Em seguida libera para o usuário enviar uma mensagem e quando o usuário envia entra na segunda condicional da função client() que faz a troca de chaves simétricas, das assinaturas e envia ao destino
- A partir da segunda as condicionais da função run() do Server são responsáveis respectivamente por: decriptar a mensagem recebida e coletar a chave simétrica do destinatário; Verificar se a assinatura não foi alterada demonstrando uma adulteração da chave simétrica; Decriptar e printar a mensagem

Conclusão

A criptografia assimétrica em conjunto com a simétrica garante um melhor desempenho devido a criptografia assimétrica ser menos custosa computacionalmente. A funcionalidade de assinatura usando os algoritmos RSA agrega autenticidade na troca das mensagens.

Este programa comparado ao código de aplicação de chave simétrica não precisa gravação da mesma em texto pleno ou em arquivo, porém, tem como desvantagem a comunicação entre somente 2 clientes.

E como melhoria sugere fazer o envio da assinatura concatenado a mensagem; Reduzir a quantidade de variáveis de flag; Evitar redundância da chamada da função de troca da chave pública; Adicionar mais eventos de verificação try/catch.

Bibliografia

FERNET. Symmetric encryption. Disponível em: https://github.com/pyca/cryptography/blob/main/docs/fernet.rst. Acesso em: 12 de agosto de 2021.

GITHUB. Código Fonte Original da Aplicação. Disponível em: https://github.com/grakshith/p2p-chat-python. Acesso em: 12 de agosto de 2021.