



Aula 2

Semana 2

Atividade de Criptografia

AN36B/EC38D/IF64J

Segurança e Auditoria de Sistemas

Lucas Sampaio

Palavras Chave!

Python 3, Criptografia, Algoritmos de criptografia simétrica, sockets, sniffer, Wireshark

Importante!

O material da disciplina está disponível no Moodle da instituição. Para maiores instruções de como instalar o Python e o Wireshark na sua máquina acesse: www.google.com

Introdução

Esta atividade tem como objetivos:

- Criar uma aplicação de comunicação escrita peer-to-peer (P2P)
- Garantir a confidencialidade da comunicação
- Utilizar a arquitetura cliente-servidor
- Utilizar Sockets
- Verificar se a política de segurança é satisfeita por meio de um sniffer de rede.

Ferramentas

São ferramentas necessárias para a realização da atividade:

- Python 3;
- Alguma biblioteca de criptografia do Python (cryptography, por exemplo);
- Wireshark (ou outro sniffer de rede);

Metodologia

Será providenciado no Moodle o código fonte de uma aplicação em Python 3 funcional que estabelece comunicação entre diferentes 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. Para verificar a funcionalidade do sistema sugere-se que você abra um terminal para cada instância (primeiro servidor e depois cliente) e ao final em um novo terminal execute um novo cliente:

```
#Terminal 1
```

```
>>python Server.py
```

```
#Terminal 2
```

```
>>python Client.py
```

```
#Terminal 3
```

```
>>python Client.py
```

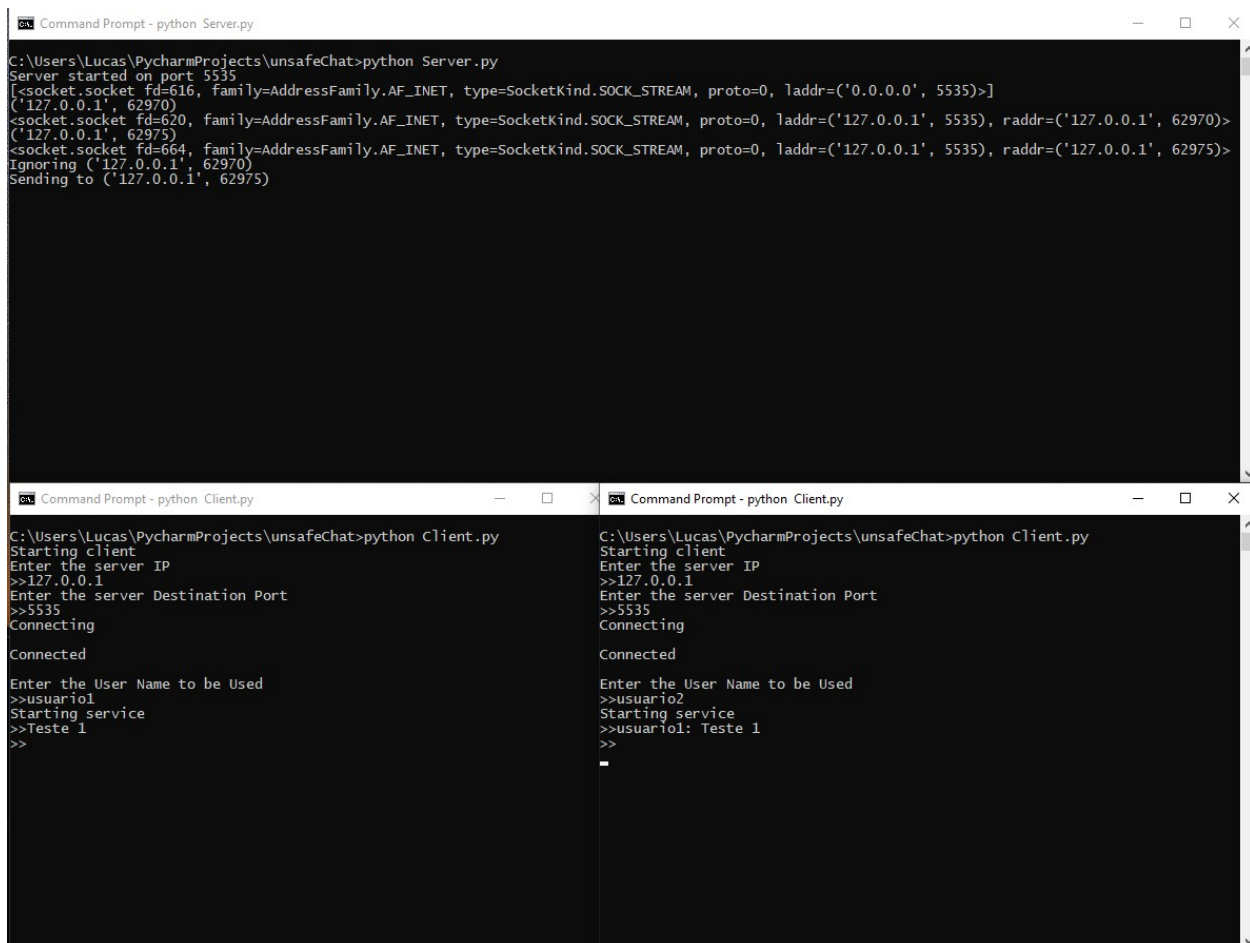
Assim quando enviar uma mensagem pelo cliente do Terminal 2 o do Terminal 3 também receberá a mensagem conforme a Figura 1 abaixo.

```
Command Prompt - python Server.py
C:\Users\Lucas\PycharmProjects\unsafeChat>python Server.py
Server started on port 5535
[<socket.socket fd=616, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('0.0.0.0', 5535)>]
('127.0.0.1', 62970)
<socket.socket fd=620, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 5535), raddr=('127.0.0.1', 62970)>
('127.0.0.1', 62975)
<socket.socket fd=664, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 5535), raddr=('127.0.0.1', 62975)>

Command Prompt - python Client.py
C:\Users\Lucas\PycharmProjects\unsafeChat>python Client.py
Starting client
Enter the server IP
>>127.0.0.1
Enter the server Destination Port
>>5535
Connecting
Connected
Enter the User Name to be Used
>>usuario1
Starting service
>>

Command Prompt - python Client.py
C:\Users\Lucas\PycharmProjects\unsafeChat>python Client.py
Starting client
Enter the server IP
>>127.0.0.1
Enter the server Destination Port
>>5535
Connecting
Connected
Enter the User Name to be Used
>>usuario2
Starting service
>>
```

Figura 1: Inicializando os terminais.



```
Command Prompt - python Server.py
C:\Users\Lucas\PycharmProjects\unsafeChat>python Server.py
Server started on port 5535
[<socket.socket fd=616, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('0.0.0.0', 5535)>]
('127.0.0.1', 62970)
<socket.socket fd=620, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 5535), raddr=('127.0.0.1', 62970)>
('127.0.0.1', 62975)
<socket.socket fd=664, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 5535), raddr=('127.0.0.1', 62975)>
Ignoring ('127.0.0.1', 62970)
Sending to ('127.0.0.1', 62975)

Command Prompt - python Client.py
C:\Users\Lucas\PycharmProjects\unsafeChat>python Client.py
Starting client
Enter the server IP
>>127.0.0.1
Enter the server Destination Port
>>5535
Connecting
Connected
Enter the User Name to be Used
>>usuario1
Starting service
>>Teste 1
>>

Command Prompt - python Client.py
C:\Users\Lucas\PycharmProjects\unsafeChat>python Client.py
Starting client
Enter the server IP
>>127.0.0.1
Enter the server Destination Port
>>5535
Connecting
Connected
Enter the User Name to be Used
>>usuario2
Starting service
>>usuario1: Teste 1
>>
```

Figura 2: Enviando uma mensagem.

Quando uma mensagem é enviada o comportamento esperado é registrado na Figura 2.

O exercício desta aula consiste em alterar o código fonte do programa de tal sorte que não seja possível observar o comportamento do sniffer de rede apresentado na Figura 3:

Para maiores detalhes de utilização do Wireshark consulte a documentação da aplicação. 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 o servidor e quantos clientes quiser.

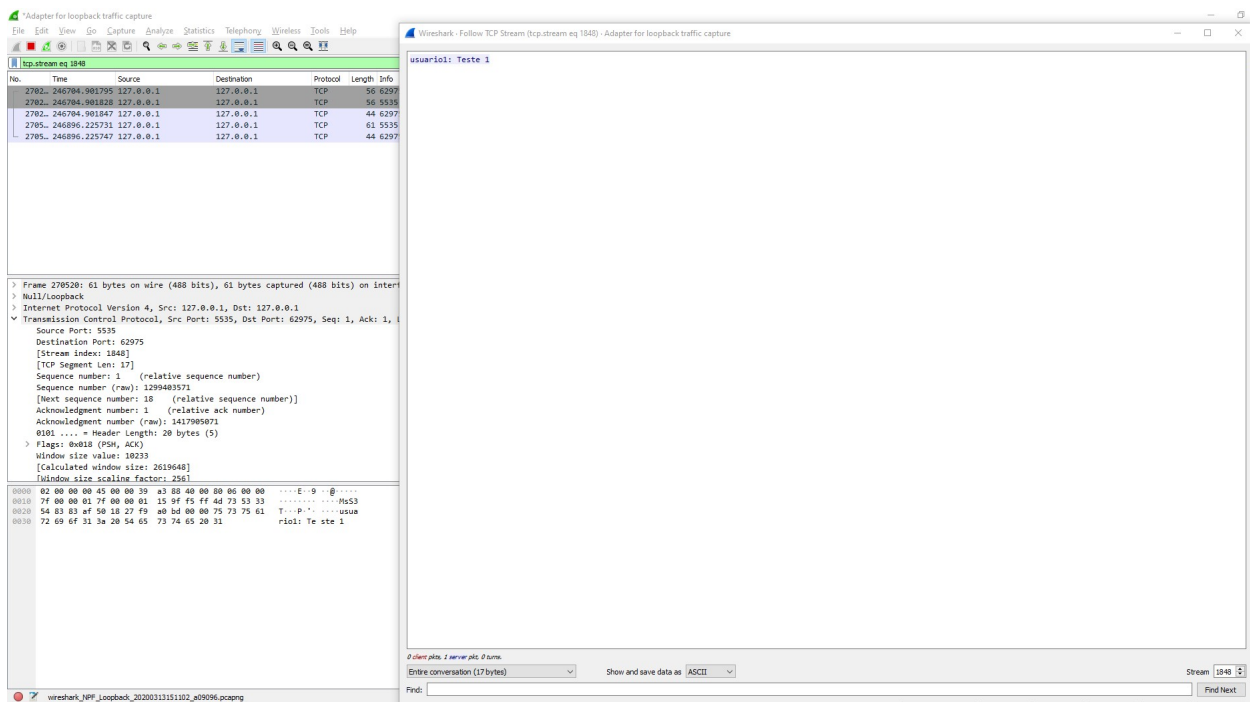


Figura 3: Observando a troca de pacotes e o texto pleno.

- 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 do Moodle, 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

Seu objetivo é 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.

Notas Importantes

- O código fonte pode ser alterado sem nenhum problema;

- É possível utilizar o programa fornecido via LAN/WAN desde que as configurações de sistema operacional, roteadores, etc. sejam feitas corretamente.
- Sinta-se livre para mudar a interface e a forma como o programa interage com o usuário.
- A definição de chaves de criptografia pode ser feita como você desejar. Lembre-se do que viu em aula à respeito de chaves de criptografia.

O que será avaliado

- Autenticidade do código.
- Configuração/Utilização das chaves.
- Número possível de conexões simultâneas.
- Sua análise das possíveis vulnerabilidades que sua solução apresenta.
- Relatório devidamente formatado explicando como foi solucionado o problema proposto pela atividade.

Leitura Complementar

- [google.com](https://www.google.com)
- [wireshark.org](https://www.wireshark.org)
- [python.org](https://www.python.org)
- https://www.tutorialspoint.com/python/python_basic_syntax.htm

Código Fonte Original da Aplicação

<https://github.com/grakshith/p2p-chat-python>