Article

Atividade de Criptografia

Pedro Bernardi Alves [1]

[1] Engenharia de Computação; (Pedro Bernardi) pedroalves@alunos.utfpr.edu.br

1. Introdução

Configurar um servidor de sala de bate-papo simples, que permita que vários clientes se conectem com confidencialidade, autenticidade e integridade na comunicação entre os meios.

2. Método

Através do socket é aberto um canal de comunicação bidirecional e que estabelece a comunicação entre um servidor e clientes. Assim, configurado um soquete em cada extremidade, que permite que um cliente interaja com outros clientes por meio do servidor. O soquete no lado do servidor se associa a alguma porta de hardware no lado do servidor. Qualquer cliente que tenha um soquete associado à mesma porta pode se comunicar com o soquete do servidor.

A utilização de thread, para que, toda vez que um usuário se conecta ao servidor, um thread separado é criado para esse usuário e a comunicação do servidor para o cliente ocorre ao longo de threads individuais com base em objetos de soquete criados para a identidade de cada cliente.

Foi utilizado a biblioteca cryptography do python[1], que fornece funções de alto nível e interfaces de baixo nível para algoritmos criptográficos comuns. Que fornece o ECDH(Elliptic Curve Diffie–Hellman Key Exchange), um conjunto de algoritmos para geração de chaves, criptografia e descriptografia, para criptografia assimétrica. E AES (Advanced Encryption Standard) com 128 bits no modo CBC (Cipher Block Chaining) para algoritmo simétrico.

O fluxo entre cliente e servidor acontece após o servidor ser iniciado, e posteriormente um cliente. Devemos primeiro devemos separar em duas etapas, a primeira é a troca de chaves, e a segunda o envio de mensagem. Para cada etapa temos um algoritmo diferente, no caso para a troca de chaves, deveremos ter um algoritmo assimétrico, e para troca de mensagem um algoritmo simétrico. Essa escolha se deve a sua agilidade, o algoritmo simétrico é menos custoso computacionalmente, assim sendo uma melhor escolha para troca de mensagens.

O próximo passo é um esquema de acordo de chaves, que permite que duas partes, cada uma com um par de chaves público-privadas de ECDH, estabeleçam um segredo compartilhado em um canal inseguro. Assim que o servidor é iniciado, geramos a chave privada e publica do servidor, e posteriormente é feito a serialização dos dados da chave para bytes. Por fim a chave publica é enviada e verificada por assinatura para todos clientes conectados ao socket.

O cliente faz o mesmo processo acima para gerar a chave publica e privada, e sua serialização, e também faz o envio da chave publica. E assim, ele aguarda o recebimento da chave publica do servidor. O servidor verifica se recebeu a chave publica do cliente com o recebimento da assinatura, aqui é feito um processo para muitos clientes, logo é gravado a chave publica de determinado cliente com seu endereço especifico em uma lista.Finalmente, a etapa da troca de chave entre servidor e clientes é finalizada.

Temos agora o processo de criação das chaves compartilhadas. No cliente o processo começa pelo carregamento da chave publica serializada do servidor. Depois é feito a

Citation: Atividade de Criptografia. *Appl. Sci.* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

derivação com a chave publica do servidor para criar a chave compartilhada entre o cliente especifico e servidor. O mesmo processo acontece para o servidor.

A troca de mensagens é garantido pois temos as chaves compartilhadas, do servidor e clientes. Agora, o cliente digita uma cadeia de caracteres a ser passado para encriptação. O algoritmo simétrico é presente nessa etapa, porém primeiro é necessário criar um nonce único para o modo CBC do algoritmo. Assim é combinado o algoritmo AES com o modo CBC, e feito a encriptação do texto pleno recebido e enviado ao servidor. O servidor ao receber, faz o mesmo processo para descriptografia, resultando no texto pleno. Com o texto pleno, através da lista de chaves publicas dos endereços conectados pelo socket, ele faz a criptografia com tais chaves para o envio a todos clientes conectados. Finalizando assim a etapa de troca de mensagens, e aguardando a próxima.

3. Procedimento

Para observar o comportamento do sniffer de rede, deve-se utilizar o software Wireshark. A execução no linux deve-se ser feito da seguinte forma:

```
#Terminal 1
sudo Wireshark
```

Após executado, entre na captura de pacotes na interface de loopback ou 127.0.0.1. O segundo passo começa com a execução do server e clientes, para isso abra os terminais indicados em novas instancias, obedecendo a ordem de server primeiro e depois os n clientes:

```
#Terminal 1
python Server.py

#Terminal 2
python Client.py

#Terminal 3
python Client.py
```

Figure 1. Inicialização do servidor

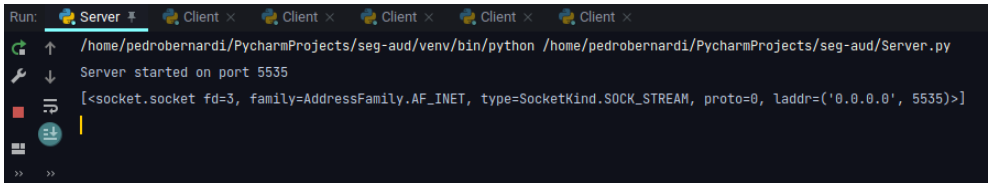
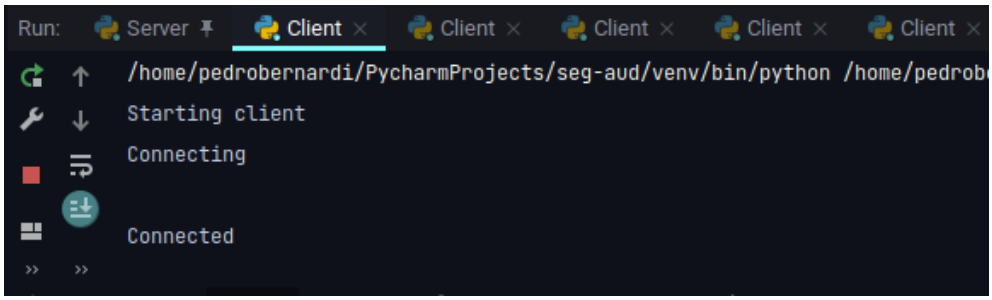


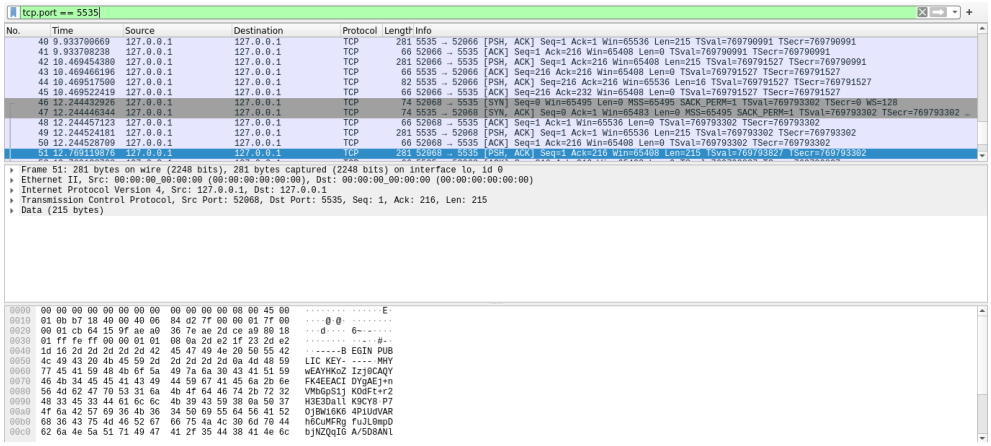
Figure 2. Inicialização dos clientes



Os clientes devem executar no IP/Porta padrão definidos, sendo o ip "127.0.0.1" e a porta "5535". A escolha do nome de usuário é feito em seguida. Em seguida envie mensagens por um cliente que executou.

Agora voltando para o software Wireshark, Na barra de filtros digite "tcp.port == 5535" para focar a apenas nos pacotes da porta 5535. E por fim, selecione um pacote e aperte CTRL + ALT + SHIFT + T, aqui temos os dados capturados na comunicação entre o servidor e clientes.

Figure 3. Wireshark com filtro e observando a troca de dados



4. Resultado

A partir da troca de chaves e mensagens, o seu fluxo é capturado pelo software wireshark, a fim de averiguar se os pacotes com os dados possuem integridade e autenticidade.

Figure 4. Troca de chaves e assinatura com integridade e autenticidade

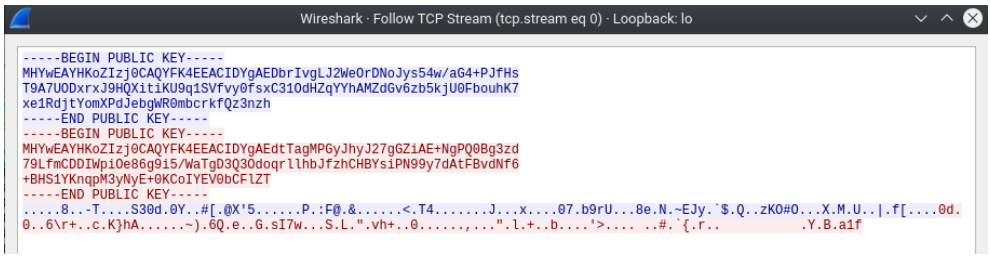
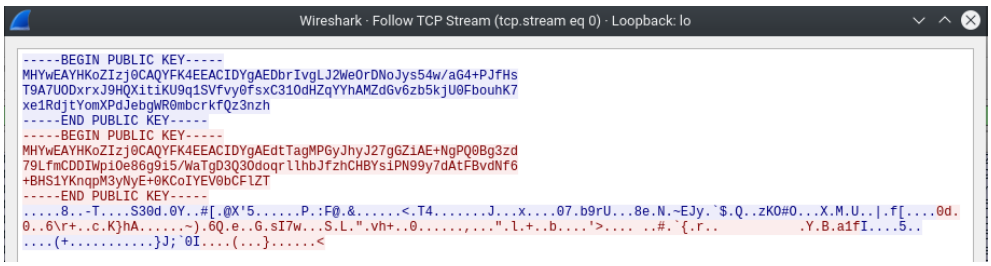


Figure 5. Troca de mensagens com integridade e autenticidade



5. Conclusão

Restringir o acesso aos dados estritamente aos meios autorizados, proteger as informações contra alterações não autorizadas e estar disponível para clientes autorizados interruptamente são os desafios de segurança tidos como objetivo nesse trabalho. Esse escopo foi atingido com a utilização de criptografia e descriptografia, fornecidos por algoritmo simétrico e assimétrico.

Há um número significativo de potenciais vulnerabilidades a curvas elípticas, como ataques de canal lateral e de twist-security[2]. Esses ataques ameaçam invalidar a segurança

que o ECC pretende fornecer às chaves privadas. Os ataques de canal lateral geralmente resultam em vazamentos de informações. Uma mitigação para ambos ataques, seria uma melhor validação dos parâmetros utilizados na atividade. Vulnerabilidades adicionais podem ser causadas pela falta de testes funcionais e de unidade, não garantindo a total funcionalidade para os diversos casos de uso.

References

1. "Symmetric encryption — Cryptography 38.0.0.dev1 documentation," Cryptography.io, 2022.
2. Cho, S., Jin, S. & Kim, H. Side-channel vulnerabilities of unified point addition on binary huff curve and its countermeasure. *Applied Sciences*. **8**, 2002 (2018)