DCC023 Relatório de Trabalho Prático

Vinicius Silva Gomes – 2021421869

Data: 10/04/2024

O que foi implementado

Nesse trabalho prático foi implementado um `client` que se comunica com um servidor remoto e autentica usuários ou grupos de usuários a partir de seus números de matrícula e códigos de autenticação gerados através das autenticações individuais, respectivamente.

Para debugar o código, foi definida uma rotina que analisa a resposta enviada pelo servidor e verifica se o formato dela se parece com o formato esperado para erros retornados pelo servidor. Uma vez que um erro seja recebido, ou seja, houve algum problema com a mensagem enviada pelo `client` anteriormente (formato inválido, tamanho de pacote incorreto, etc), o programa dispara uma exceção e exibe o código do erro obtido.

No cenário onde o programa não obtenha uma resposta do servidor, foi definido um *timeout* de 10.0 segundos até que o programa tente reenviar essa mensagem que possivelmente se perdeu. O programa irá tentar reenviar a mensagem 3 vezes e irá encerrar caso não obtenha sucesso em nenhuma dessas tentativas.

Por fim, se a comunicação for bem sucedida, o programa exibe a informação enviada pelo servidor: SAS ou GAS gerado com a autenticação do usuário ou grupo de usuário, respectivamente; ou o bit que indica a validade do *token* SAS/GAS enviado na requisição.

Executando o código

O trabalho foi desenvolvido utilizando a linguagem de programação Python e alguns de seus módulos *built in* (*socket*, *struct*, etc). Para tanto, é recomendado que a instalação da linguagem seja feita através da documentação oficial[1].

Além disso, a versão da linguagem que foi utilizada durante o desenvolvimento do trabalho foi a 3.10.12. Portanto, é recomendado que a versão utilizada ao longo da bateria de testes seja igual ou superior a essa. Uma vez que o ambiente esteja configurado, o comando

```
python3 main.py <host> <port> <command>
```

deve ser utilizado para executar as funcionalidades desenvolvidas e realizar a comunicação entre o cliente e o servidor. Os argumentos específicos para o comando solicitado devem ser informados logo após o `<command>` na CLI.

Testes realizados para avaliação

Para a avaliação do trabalho, foi realizado todo o fluxo de requisições utilizando tanto o IPv4 quanto o *hostname* do servidor. Ambas as validações foram realizadas com o número de matrícula 2021421869 e *nonce* 44, através dos seguintes códigos:

```
# IPv4

% python3 main.py 150.164.213.243 51001 itr 2021421869 44
# Saida: 2021421869:44:3aab8528e3fe5b7c4770f5b10d7802e55ad3e4be20e2f44c2ce1f425f5be6749

% python3 main.py 150.164.213.243 51001 itv 2021421869:44:3
    aab8528e3fe5b7c4770f5b10d7802e55ad3e4be20e2f44c2ce1f425f5be6749
# Saida: 0

% python3 main.py 150.164.213.243 51001 gtr 1 2021421869:44:3
    aab8528e3fe5b7c4770f5b10d7802e55ad3e4be20e2f44c2ce1f425f5be6749
# Saida: 2021421869:44:3aab8528e3fe5b7c4770f5b10d7802e55ad3e4be20e2f44c2ce1f425f5be6749+7862
    d39b080c720541aad8f4c9d611834e373d9a5ae4bceed6353760d0bfa645

```

[1] Disponível em: https://www.python.org/downloads/.

```
% python3 main.py 150.164.213.243 51001 gtv 2021421869:44:3
    aab8528e3fe5b7c4770f5b10d7802e55ad3e4be20e2f44c2ce1f425f5be6749+7862
    d39b080c720541aad8f4c9d611834e373d9a5ae4bceed6353760d0bfa645
# Saida: 0
```

```
# Hostname

% python3 main.py pugna.snes.dcc.ufmg.br 51001 itr 2021421869 44
# Saida: 2021421869:44:3aab8528e3fe5b7c4770f5b10d7802e55ad3e4be20e2f44c2ce1f425f5be6749

% python3 main.py pugna.snes.dcc.ufmg.br 51001 itv 2021421869:44:3
    aab8528e3fe5b7c4770f5b10d7802e55ad3e4be20e2f44c2ce1f425f5be6749
# Saida: 0

% python3 main.py pugna.snes.dcc.ufmg.br 51001 gtr 1 2021421869:44:3
    aab8528e3fe5b7c4770f5b10d7802e55ad3e4be20e2f44c2ce1f425f5be6749
# Saida: 2021421869:44:3aab8528e3fe5b7c4770f5b10d7802e55ad3e4be20e2f44c2ce1f425f5be6749+7862
    d39b080c720541aad8f4c9d611834e373d9a5ae4bceed6353760d0bfa645

% python3 main.py pugna.snes.dcc.ufmg.br 51001 gtv 2021421869:44:3
    aab8528e3fe5b7c4770f5b10d7802e55ad3e4be20e2f44c2ce1f425f5be6749+7862
    d39b080c720541aad8f4c9d611834e373d9a5ae4bceed6353760d0bfa645
# Saida: 0
```

A quebra de linha no meio das Strings dos *tokens* é apenas para melhorar a visualização no relatório. O comando deve ser formatado todo em uma linha na hora de executar o programa, caso contrário haverá erros com relação a passagem dos parâmetros.

Limitações do Protocolo de Autenticação proposto

O Protocolo de Autenticação definido, apesar de funcionar para o que se propõem, apresenta falhas que podem ser exploradas com certa facilidade. A principal dessas falhas é a possibilidade de usuários mal-intencionados se passarem por outros usuários comuns na comunicação com o servidor.

O servidor recebe apenas o número de matrícula para gerar o *token* identificador do usuário (alterar o *nonce* faz com que o *token* gerado seja diferente, mas ele ainda está atrelado e representa o mesmo usuário na troca de mensagens). Assim, uma vez que esse número seja conhecido, é fácil autenticar usando essa matrícula e se passar por outro usuário ao comunicar com o servidor.

Para evitar essa situação, seria interessante que o protocolo apresentasse algum mecanismo que aumentasse a complexidade da autenticação, inserindo, por exemplo, uma etapa que verificasse alguma informação individual de cada usuário, como uma senha ou código único.

Por fim, seria desejável que houvesse alguma camada de criptografia na comunicação entre o cliente e o servidor. Isso previne que usuários mal-intencionados sejam capazes de capturar as informações que estão sendo trocadas ao longo da comunicação, protegendo os *tokens* de autenticação de vazamentos.