Projeto 2: Comunicações Seguras

Trabalho Elaborado por:

- Pedro Escaleira, 88821
- Rafael Simões, 88984

1. Índice

2. Introdução de algoritmos

Foi nos proposto a implementação de uma comunicação segura entre um cliente e um servidor. A comunicação resume-se à transferência de um ficheiro escolhido pelo cliente para o armazenamento num servidor.

Para esta comunicação ser classificada como segura são necessários mecanismos de encriptação do conteúdo do ficheiro para que haja comunicação entre o cliente e o servidor sem comprometer a confidencialidade e integridade dos dados.

Para isso foi nos proposto a implementação de vários protocolos consecutivos para conseguir atingir os objetivos referidos anteriormente.

Para uma melhor perceção e compreensão destes protocolos,seguidamente, cada um destes serão explicados individualmente e detalhadamente.

3. Negociação de algoritmos

a. Ao longo desta comunicação irão ser usados diferenciados algoritmos para diferentes objectivos.
   i. Algoritmo de troca de chaves
   ii. Algoritmos de cifra
   iii. Modo de cifra
   iv. Algoritmo de hashes
b. Neste caso foi implementado um algoritmo de trocas de chaves : **Diffie-Helman**
c. Os algoritmos relacionados com cifra tem como objetivo de compactuar para realização de encriptação simétrica
d. Algoritmos de hashes são usados para controlo de integridade de ficheiro , juntamente com o MAC (explicado em seguida) , e para a derivação das chaves geradas pelo algoritmo de Diffie-Hellman (este passo é necessário pois muitos dos algoritmos tem restrição no tamanho das keys)

1. O cliente tem um leque de algoritmos que suporta.
2. A primeira mensagem enviada é por parte do cliente e tem como conteúdo os algoritmos que este suporta
3. O servidor ao receber a mensagem com os algoritmos implementados pelo cliente verifica , de entre os recebidos, quais implementa, enviando uma mensagem para o cliente com os algoritmos filtrados.
   a. Caso a filtração não tenha algoritmos em comum, o servidor envia uma mensagem de erro ao cliente , terminando em seguida a ligação
   b. Caso a filtração tenha algoritmos em comum: **Passo 4**
4. O cliente escolhe que algoritmos usar de acordo com a mensagem recebida pelo servidor.
   a. Devido ao facto do *backend* (*default_backend*) ter problemas de compatibilidade com algumas combinações de cifras e modo de cifras (Ex: 3DES & GCM), o cliente tem um mecanismo de escolha com objetivo de evitar estas incompatibilidades
      i. Caso não seja possível realizar uma escolha de algoritmos, o servidor manda uma mensagem de erro para o cliente e a ligação termina
      ii. Caso seja possível escolher um algoritmo: **Passo 5**
5. Após o cliente finalizar o processo de escolha dos algoritmos, este informa o servidor enviando uma mensagem com os algoritmo selecionados.
6. O servidor confirma ao cliente que recebeu os algoritmos mandando uma mensagem de “OK”

7. Como este intercâmbio de mensagens, o processo de escolha de algoritmos encontra-se finalizado.

É o cliente a escolher os algoritmos para evitar *overhead* em termos de performance no servidor , pois um servidor pode servir vários clientes em simultâneo, com este comportamento o servidor restringe os algoritmos usados evitando algoritmos menos seguros, além disso a probabilidade da ligação continuar é maior pois permite uma maior escolha por parte do cliente (evitando a ligação terminar por incompatibilidade de algoritmos)

Sucesso:

Cliente:

```
user@pc:~$ python3 client.py client.py -r
2019-11-17 16:35:53 user-System root[5482] INFO Sending file:
/home/rafael/Desktop/sio_project_2/client.py to 127.0.0.1:5000 LogLevel: 20
2019-11-17 16:35:53 user-System root[5482] INFO Client algorithms: {'ciphers': ['ChaCha20', 'AES',
'TripleDES', 'Blowfish', 'ARC4'], 'modes': ['CBC', 'GCM', 'ECB'], 'hashes': ['SHA256', 'SHA512',
'MD5']}
2019-11-17 16:35:53 user-System root[5482] INFO Algorithms implemented by server : {'type':
'AVAILABLE_ALGORITHMS', 'data': {'ciphers': ['ChaCha20', 'TripleDES', 'AES'], 'modes': ['GCM',
'CBC'], 'hashes': ['SHA256', 'SHA512', 'MD5']}}
2019-11-17 16:35:53 user-System root[5482] INFO Chosen Algorithm DH_TripleDES_CBC_SHA256
2019-11-17 16:35:53 user-System root[5482] INFO Algorithm accepted from server
```

Servidor:

```
user@pc:~$ python3 server.py
Connection from ('127.0.0.1', 40968)
2019-11-17 16:35:53 user-System root[5482] INFO Client algorithms : {'ciphers': ['ChaCha20', 'AES',
'TripleDES', 'Blowfish', 'ARC4'], 'modes': ['CBC', 'GCM', 'ECB'], 'hashes': ['SHA256', 'SHA512',
'MD5']}
2019-11-17 16:35:53 user-System root[5482] INFO Algorithm picked by client DH_ChaCha20_GCM_SHA512
```

Erro:

Cliente:

```
user@pc:~$ python3 client.py client.py --cipher Blowfish
2019-11-17 16:35:53 user-System root[5482] INFO Sending file:
/home/rafael/Desktop/sio_project_2/client.py to 127.0.0.1:5000 LogLevel: 20
2019-11-17 16:35:53 user-System root[5482] INFO Client algorithms: Client algorithms: {'ciphers':
['Blowfish'], 'modes': ['CBC'], 'hashes': ['SHA512']}
2019-11-17 16:35:53 user-System root[5482] WARNING Got error from server: See server
```

Servidor:

```
user@pc:~$ python3 server.py
Connection from ('127.0.0.1', 41934)
2019-11-17 16:35:53 user-System root[5482] INFO Client algorithms : {'ciphers': ['Blowfish'],
'modes': ['CBC'], 'hashes': ['SHA512']}
2019-11-17 16:35:53 user-System root[5482] WARNING Invalid algorithm request!
2019-11-17 16:35:53 user-System root[5482] INFO Closing transport
```

Como para a troca de mensagens está a ser usado *json* para realizar o intercâmbio de dados. Foi adotado um formato de escolha em string com a combinação de todos os algoritmos escolhidos com o seguinte formato:

<Algoritmo de troca de chaves>_<Algoritmos de cifra>_<Modo de cifra>_<Algoritmo de hash>

Algoritmos implementados por cada entidade:

Server:

1. **Cifras :** ChaCha20, AES, TripleDES
2. **Modes:** CBC,GCM
3. **Hashes:** SHA256,SHA512,MD5

Client:

1. **Cifras :** ChaCha20, AES, TripleDES,Blowfish,ARC4
2. **Modes:** CBC,GCM,ECB
3. **Hashes:** SHA256,SHA512,MD5

4. Troca de chaves

a. No fim dos algoritmos estarem definidos e avaliados procede-se o protocolo de troca de chaves.
b. Para este protocolo foi implementado o algoritmo ***Difiie-Hellman*** (DH) , mais especificamente *Diffie-Hellman ephemeral* (DHE)
c. Considerando a natureza dos canais de comunicação entre cliente e servidor, o algoritmo DH é um método que permita a duas entidades partilharem uma chave secreta em canais de comunicação inseguros

O cliente começa por gerar os parâmetros de entrada necessários para o DH:

1. **p -** o valor primo de elevada dimensão
2. **q** - o valor do gerador

1. Com estes parâmetros e com o uso da DH implementada pela bibilioteca criptografica *Cryptography.io* o cliente gera uma chave pública e uma chave privada
2. A chave pública gerada pelo cliente e os parâmetros de entrada , p e q , são enviados para o servidor
3. O servidor com o DH e os parâmetros de entrada recebidos gera ,também, uma chave privada e pública
4. O servidor permuta a sua chave privada com a chave pública do cliente e obtém uma chave partilhada, que será usada para protocolos futuros
5. No fim de obter a chave partilhada o servidor manda a sua chave pública para o cliente
6. O cliente recebe a chave pública do servidor e este gera a chave partilhada com um processo igual ao do servidor , como explicado anteriormente.
7. Assim ambas as entidades têm uma chave partilhada e o processo de encriptação e desencriptação pode começar.

Apenas de referir que em ambas as entidades a chave partilhada passa por uma função de derivação de chaves(usando a função de síntese escolhida no processo de negociação) que é necessária pois esta chave necessita de ter dimensões fixas de acordo com os algoritmos usados

O Cliente toma a iniciativa de geração de chaves pois o gerador de parâmetros de entrada do DH , demora algum tempo (para length=1024) logo para não causar *overhead* em termos de performance no servidor o cliente gera os parâmetros e envia-os para o servidor

Cliente:

```
user@pc:~$
...
2019-11-17 16:35:53 user-System root[5482] INFO Initializing DH
2019-11-17 16:35:53 user-System root[5482] INFO Shared Key with DH :
b'\x1a&\xdc\xd7aP\xd5\tiAH[\xaa\xed_\x0c\x88}j\x15\xd2\xfa\xdc\x83\xe5\x1dmvA,t\xa6'
```

Servidor:

```
user@pc:~$
...
2019-11-17 16:35:53 user-System root[5482] INFO Shared_key with DH :
b'\x1a&\xdc\xd7aP\xd5\tiAH[\xaa\xed_\x0c\x88}j\x15\xd2\xfa\xdc\x83\xe5\x1dmvA,t\xa6'
```

5. Suporte para confidencialidade

a. Caso os protocolos de negociação de algoritmos e trocas de ficheiros estejam completos. O protocolo de suporte a confidencialidade pode iniciar.
b. Neste protocolo o fluxo de mensagens é bastante simples

1. Antes deste protocolo iniciar, o cliente manda uma mensagem para o servidor com o nome do ficheiro que este pretende enviar.
2. A partir deste momento o cliente manda *chunks* do ficheiro encriptado. Para encriptar o pedaço de ficheiro o cliente usa o algoritmo de cifra e o modo de cifra utilizando a chave partilhada.
3. O servidor vai recebendo estes pedaços, desencripta-os (usado os mesmo algoritmos referidos anteriormente utilizando também a chave partilhada) e guarda-os.

Cliente:

```
user@pc:~$
...
2019-11-17 16:35:53 user-System root[5482] INFO Sending file Name  to Server :
/home/rafael/Desktop/sio_project_2/client.py
2019-11-17 16:35:53 user-System root[5482] INFO Sending file to Server…
2019-11-17 16:35:53 user-System root[5482] INFO File transferred. Closing transport
2019-11-17 16:35:53 user-System root[5482] INFO The server closed the connection
```

Server:

```
user@pc:~$
...
2019-11-17 16:35:53 user-System root[5482] INFO Process Open: {'type': 'OPEN', 'file_name':
'/home/rafael/Desktop/sio_project_2/client.py'}
2019-11-17 16:35:53 user-System root[5482] INFO File open
2019-11-17 16:35:53 user-System root[5482] INFO Processing Data...
2019-11-17 16:35:53 user-System root[5482] INFO Processing Data...
                                        ....
2019-11-17 16:35:53 user-System root[5482] INFO Process Close: {'type': 'CLOSE'}
```

6. Suporte para integridade

a. Este protocolo é executado em conjunto com o protocolo acima referido.
b. Para suportar integridade do ficheiro o sistema implementa um mecanismo de *MAC* (Message authentication codes). MAC é uma ferramenta para calcular um código de integridade (e autenticidade) usando uma função *hash* criptográfica acoplada com uma chave (que neste caso é a chave partilhada)

1. Quando o cliente encripta um pedaço do ficheiro (processo do protocolo anterior) este calcula a síntese do conteúdo encriptado usando a chave partilhada usando o MAC da biblioteca do Cryptography.io., enviando o resultado da respetiva síntese para o servidor
2. Quando o servidor recebe um pedaço de informação do ficheiro encriptado, recebe também a síntese calculada anteriormente. Para verificar a integridade do ficheiro o servidor faz o processo de cálculo de síntese da informação do ficheiro encriptada recebida do cliente usando ,também, a chave partilhada.
3. No final do cálculo de síntese, o servidor compara o valor da síntese recebida do cliente e da síntese calculada.
    a. Caso estes valores sejam iguais a ligação continua , pois a integridade do ficheiro não foi comprometida
    b. Caso os valores sejam diferentes o servidor termina a ligação, pois a integridade do ficheiro foi comprometida.

Sucesso:

O mesmo output verificado no protocolo anterior

Erro:

Cliente:

```
user@pc:~$
...
2019-11-17 16:35:53 user-System root[5482] INFO Sending file to Server...
2019-11-17 16:35:53 user-System root[5482] WARNING socket.send() raised exception.
2019-11-17 16:35:53 user-System root[5482] INFO The server closed the connection
```

Servidor:

```
user@pc:~$
...
2019-11-17 16:35:53 user-System root[5482] INFO Processing Data...
2019-11-17 16:35:53 user-System root[5482] WARNING MAC authentication Failed
2019-11-17 16:35:53 user-System root[5482] INFO Closing transport
```

7. Mecanismo de rotação de chaves

a. Este protocolo é usado para fazer troca de chaves (neste caso trocar a chave partilhada) quando uma chave é usada um número determinado de vezes nos processos de encriptação/desencriptação.
b. Isto é feito, por boa prática, pois reduz a quantidade de conteúdo encriptado com uma determinada key para que a quantidade de material exposta por um único comprometimento de chave seja menor. Em suma, fortalece o processo de encriptação, pois dificulta o trabalho dos criptoanalistas para *crackear* a chave.

1. Quando o cliente manda o nome do ficheiro a ser transmitido para o servidor (após a negociação de algoritmos e troca de chaves), o servidor gera um valor limite para o uso da chave partilhada.
2. Posteriormente o servidor envia este valor para o cliente.
3. No processo de envio de pedaços de um ficheiro, por parte do cliente, para o servidor, o cliente atualiza uma variável que contabiliza o número de interações (iterações,neste caso) feitas com o servidor.
4. Caso o valor desta variável seja maior ou igual ao limite definido anteriormente, o cliente começa a troca da chave partilhada com o servidor (nesta fase o processo é igual , ao processo de troca de chaves com o uso de DH, como explicado anteriormente)
5. Quando ambas as entidades tiverem as novas chaves partilhadas definidas, o processo de envio de pedaços do ficheiro é retomado.

O cliente é que inicia a troca de chaves, pelo simples facto , de anteriormente ser o cliente a iniciar a troca de chaves com DH

Cliente:

```
user@pc:~$ python3 client.py some_file -r
...
2019-11-17 16:35:53 user-System root[5482] INFO Sending file Name  to Server :
/home/rafael/Desktop/sio_project_2/filme.mp4
2019-11-17 16:35:53 user-System root[5482] WARNING Setting limit for keys rotations (iterations) :
5000
2019-11-17 16:35:53 user-System root[5482] INFO Sending file to Server...
2019-11-17 16:35:53 user-System root[5482] INFO Change Key
2019-11-17 16:35:53 user-System root[5482] INFO Initializing DH
2019-11-17 16:35:53 user-System root[5482] INFO Shared Key with DH :
b'\xa4\x99\x92\xe3\x01\xd2\xa5\xae\xc2\x86\x99h\xa8\xddb\xd2\x9d\x07;^\x93~s\xc5
2019-11-17 16:35:53 user-System root[5482] INFO Sending file Name  to Server :
/home/rafael/Desktop/sio_project_2/filme.mp4
2019-11-17 16:35:53 user-System root[5482] INFO Sending file to Server...
                                        ....
```

Servidor:

```
user@pc:~$ python3 server.py
...
2019-11-17 16:35:53 user-System root[5482] INFO Process Open: {'type': 'OPEN', 'file_name':
'/home/rafael/Desktop/sio_project_2/filme.mp4'}
2019-11-17 16:35:53 user-System root[5482] WARNING Setting limit for keys rotations (iterations) :
5000
2019-11-17 16:35:53 user-System root[5482] INFO Processing Data...
2019-11-17 16:35:53 user-System root[5482] INFO Changing Key
2019-11-17 16:35:53 user-System root[5482] INFO Shared_key with DH :
b'\xa4\x99\x92\xe3\x01\xd2\xa5\xae\xc2\x86\x99h\xa8\xddb\xd2\x9d\x07;^\x93~s\xc5'
2019-11-17 16:35:53 user-System root[5482] INFO Processing Data...
                                        ....
```

Documentação

Opções adicionadas ao client.py

1. -r
    a. Opção para ativar a escolha de algoritmos aleatórios no processo de negociação de algoritmos.
    b. Caso esta opção não seja escolhida, os algoritmos escolhidos são:
        i. Algoritmo de cifra: TripleDES
        ii. Modo de Cifra: CBC
        iii. Algoritmo de hash: SHA512
    c. Usage- python3 client.py <file> -r
2. --cipher <algoritmo de cifra>
    a. Opção para especificar o algoritmo de cifra
    b. Usage: python3 client.py <file> --cipher AES
3. --mode <Modo de cifra>
    a. Opção para especificar o modo de cifra
    b. Usage: python3 client.py <file> --mode CBC
4. --synthesis <Algoritmo hash>
    a. Opção para especificar o algoritmo de hash
    b. Usage: python3 client.py <file> --mode SHA512
5. --dh_key_size <key_size>
    a. Opção para especificar o tamanho da chave gerada pelo gerador de parâmetros do DH
    b. key_size >=512
    c. Usage: python3 client.py <file> --mode 512

Opções adicionadas ao server.py

1. --limit
    a. Opção usada para definir o limite de decisão na troca de chaves
    b. Usage: python3 server.py --limit 5000