



Relatório Criptografia e Segurança em Redes

2023-2024

UNIVERSIDADE DO MINHO

JORGE SOUSA A94153

LUÍS PINHEIRO A58381

Índice

Índice.....	2
Aplicação de Chat.....	3
Modo 1: Integridade.....	4
Modo 2: Confidencialidade e Integridade.....	5
Modo 3: Confidencialidade, Integridade e Autenticidade	6

Aplicação de Chat

Mais uma vez recorreu-se ao software **Python 3** para desenvolver o trabalho prático.

O código da aplicação de chat do servidor e do cliente estão no mesmo ficheiro `main.py` porque estes partilham quase a totalidade do código. A aplicação ao ser iniciada por uma ponta, tenta ser o cliente, ao falhar cai no papel do servidor. A próxima inicialização vai ser bem-sucedida ao tentar ser o cliente.

De forma a lidar com o problema de bloqueio de entrada-saída, as funções de escrita na socket, que leem do *STDIN*, são sempre executadas noutra thread. A thread termina depois de ser inserido no *STDIN*, só o fim-de-linha, e envia pelo socket o mesmo caractere `\n` para, da mesma forma, o loop-infinito de leitura do socket terminar.

Modo 1: Integridade

Deforma a verificar a integridade das mensagens é produzida e enviada pelo transmissor a **hash** da mensagem que foi digitada. Como o tamanho da hash usado é constante, 32-bit, suficiente para provar a integridade numa aplicação simples. O recetor trunca 32-bits da mensagem tenta replicar a mesma hash com a mensagem que recebeu, caso seja bem-sucedido é assumido que a mensagem não sofreu alterações.

```
lepidos@asteel:~/CSR/Trabalho-Pratico2/src$ python3 main.py
CLIENT RUNNING
Please choose a security mode:
(1) Integrity;
(2) Simetric Encryption + Integrity;
(3) Assimetric Encryption + Integrity + Authenticity;
1
selected 1
Input message to send (empty to finish)
ola
[]

lepidos@asteel:~/CSR/Trabalho-Pratico2/src$ python3 main.py
SEVER RUNNING
Serving on ('127.0.0.1', 7777)
(type ^C to finish)
Client [1] selected mode 1
Input message to send (empty to finish)
Server reading socket
[1] : 'ola'

src      = 127.0.0.1
dst      = 127.0.0.1
\options \
###[ TCP ]###
sport    = 42598
dport    = 7777
seq      = 4070911317
ack      = 2709058583
dataofs  = 8
reserved = 0
flags    = PA
window   = 512
chksum   = 0xfe4b
urgptr   = 0
options  = [('NOP', None), ('NOP', None), ('Timestamp', (1048318037, 1048314084))]
###[ Raw ]###
load     = 'ola2fe04e524ba40505a82e03a2819429cc'

###[ Ethernet ]###
dst      = 00:00:00:00:00:00
src      = 00:00:00:00:00:00
```

Modo 2: Confidencialidade e Integridade

Dando continuidade ao que processo descrito no modo 1. Em vez de enviar a mensagem em texto+hash, neste modo enviamos o conteúdo da mensagem encriptado usando uma cifra simétrica e a hash do texto, outra vez de 32-bit.

O algoritmo **Fernet** faz um conjunto de operações com uma chave à mensagem. A chave possibilita tanto cifrar como decifrar neste algoritmo. Dando assim garantia de confidencialidade, isto é só quem possui a chave consegue reverter a cifra e obter o texto da mensagem.

```
key = Fernet.generate_key()
```

Sempre que uma ligação começa o cliente gera uma chave e envia ao servidor. Neste momento ainda não é possível usar como suporte a cifras simétrica. Esta é uma vulnerabilidade grave à confidencialidade, a qual as cifras assimétricas estão endereçadas a resolver. Ou assumir que ambos já tinham acordado uma chave antes de iniciar a aplicação e optar por fixar a uma chave como variável constante no código. Caso fosse esta a implementação deveria-se usar nonce para evitar ataques replay.

```
lepidos@asteel:~/CSR/Trabalho-Pratico2/src$ python3 main.py && clear
CLIENT RUNNING
Please choose a security mode:
(1) Integrity;
(2) Simetric Encryption + Integrity;
(3) Assimetric Encryption + Integrity + Authenticity;
2
selected 2
Client generating key and sending
Input message to send (empty to finish)
Client reading socket
ola
[]

lepidos@asteel:~/CSR/Trabalho-Pratico2/src$ python3 main.py && clear
SEVER RUNNING
Serving on ('127.0.0.1', 7777)
(type ^C to finish)
Client [1] selected mode 2
Input message to send (empty to finish)
Server reading socket
[1] : 'ola'
[]

src      = 127.0.0.1
dst      = 127.0.0.1
\options \
###[ TCP ]###
sport    = 55536
dport    = 7777
seq      = 2549295961
ack      = 4254060795
dataofs  = 8
reserved = 0
flags    = PA
window   = 512
chksum   = 0xfeac
urgptr   = 0
options  = [('NOP', None), ('NOP', None), ('Timestamp', (1048403863, 1048400938))]
###[ Raw ]###
load     = 'gAAAAABlXPfnQmXMrUKOPFcr_QV-cNP0bilZRTnmAaszdpqfIMehi0Q4SWPILXZBMeIoNur3R9t7D_bDk0e0xpT9wLCP772iNQ==2fe04e524ba40505a82e03a2819429cc'
###[ Ethernet ]###
dst      = 00:00:00:00:00:00
src      = 00:00:00:00:00:00
```

Modo 3: Confidencialidade, Integridade e Autenticidade

Neste modo, criptografia assimétrica, o algoritmo escolhido foi RSA, a integridade está aninhada no método de que garante a autenticidade.

```
signature = private_key.sign(ciphertext,
                             padding.PSS(mgf=padding.MGF1(hashes.SHA256()),
                             salt_length=padding.PSS.MAX_LENGTH), hashes.SHA256())
```

A confidencialidade é garantida pelo mecanismo assimétrico, agora são geradas duas chaves por cada um, uma para encriptar e outra para desencriptar. As chaves de encriptar são enviadas no início da comunicação. Assim cada um encripta a mensagem usando a chave que recebeu. Garantindo que só quem tem a correspondente chave de desencriptar consegue reverter a transformação. Assim nem o próprio transmissor da mensagem consegue decifrar o que cifrou, pelo que usou a chave que recebeu do outro lado da comunicação.

```

Please choose a security mode:
(1) Integrity;
(2) Simetric Encryption + Integrity;
(3) Assimetric Encryption + Integrity + Authenticity;

3
selected 3
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAACAMIIIBGKCAQEA+U4UxdDhRy7HnEd1IIn3
XwRko35QXa92zbpvd+r06vLY2ZeuYkNn9y/3gSckvBcVQHEgFnpTzYcAZKDL0ntq
SuXum7tbfq31FX0gXdzLpyYeIfIjsmzSf/ZWf91VlnWAGa+3s9110cPjiIwC5F
3uBlTgnyXIR6Wbu6rue7lZLywh7aV7sRhPwxJmJmNg64zmp6/n7rTehD3g2XL4D
Vn7kMeIyKD30PvovYZqh2VLG0H1ZoL11jKFvCXILfw7CKYGncxZDaAHmcZucK
y8F7325oXsS6vvEgkrT/oKL/LK5XNgSsKUYZ5Fnyj4drxqTeIHNM5Gt1/qzwVp04
ZwIDAQAB
-----END PUBLIC KEY-----

Input message to send (empty to finish)
Client reading socket
ola
[]

chcksum      = 0x5f21
src          = 127.0.0.1
dst          = 127.0.0.1
options      = \
###[ TCP ]###
sport        = 42958
dport        = 7777
seq          = 289211775
ack          = 859864705
dataofs      = 8
reserved     = 0
flags        = PA
window       = 512
chcksum      = 0x29
urgptr       = 0
options      = [['NOP', None), ('NOP', None), ('Timestamp', (1048507632, 1048504678))]]

###[ Raw ]###
load         = '\x08\xad\x18\x16\xees7W\x1xeaE\x1e_\x19\x1xf9f'\x19\xcd\x1d\x1cxbRor\x1b0-E-"'\xf75\x0b\xab\x5e\x1fnt\x1f5\x8d\xdc\x19\x0e
\x1f9\x1a3\x1a6\x0f\x1xb5 6\x184a<'\x19\x1xb40\x185\x1xaeYvu\x1db\x17V\x1e3\x1cc.U\x1e6\x1xb7<\x1b9\x1xf2\x1ed\x1xbdb%\x1b1K\x1f\x17np\x14C\x17f6\x1f4\x1a1\x03\x1xf9
H)\x1e8p0\x1xf62\x12\x1a78\x1xb0+\x19Z\x1bbL\x1f60\x1f8\x1f0\x1b8B\x186\x0e\x1cdrT\x1c6\x0e\x1a5\x1a8\x1bftP\x1f0n\x1c12\x1b50\x1d3\x16:\x1b8\x1a0\x197\x1de\x1d8
\x1d3\x1b8a\x18e\x05\x1eej\x1d6+\x1f5\x01P\x03\x0f\x1a\x10\x1x08.\x1xee\x1xb9\x1010Z\x1a6\x1x9925\x1ab\x1b8\x1d1\x1d2\x1d0\x03\x18d\x1x94\x1e\x00Xf\x19e\x1b2u\x1x9

```