Klient:

409844-Knaur_Robin-ProjectsReview: (Skvelá analýza, ktorá dosť dopomohla k záverečnému riešeniu)

Problem:

2. Zaseknuti aplikace: Když se zadá špatné heslo aplikace nadále nereaguje. Když si Klienti vymění klíče(malokdy se povede), nasleduje prazdny řadek -> aplikace pada.

Riešenie:

Aplikacia sa už nezasekáva pri zadaní zlého hesla. Požiada sa o opakovanie. Klienti si už vždy vymenia hesla a aplikacia nepadá. (V tomto pripade a aj všetkých testovaných)

Problem:

Client.cpp: Memory leak – promena table

Riešenie:

```
57  Client::~Client()
58  {
59      if(m_SC != NULL)
60      {
61          m_SC.write("lo", 3);
62          m_SC.close();
63      }
64      if(m_p2p != NULL)
65      {
66          m_p2p.close();
67      }
68      if(m_p2pSSL != NULL)
69      {
70          m_p2pSSL.close();
71      }
72
73      delete tableOne;
74      delete tableTwo;
75  }
```

Problem:

Mazání klíče: Problem identification: C_x(code, implementation) Severity: high Practicability: easy (directly by external attacker) Description of the problem: Nikdo se nestará o smazání kliče z paměti.Utočník pak muže v pameti dohledat klíč. Proposed solution: Přemazat proměnout key v destruktoru.

Riešenie:

Kľúč sa maže(prepisuje na nuly) v deštruktore.

```
73      for(int i = 0; i < 128; i++)
74      {
75          key[i] = '\0';
76          tableOne->key[i] = '\0';
77          tableTwo->key[i] = '\0';
78      }
79
```

Problem:

Hash: Problem identification: C_x(code, implementation) Severity: high Practicability: easy (directly by external attacker) Description of the problem: Nekotrolují se intergrity zpráv, ukočník muže kdykoli zmeni zprávu a nikdo to nepozná. Proposed solution: Přidat hash na konec zprávy a byla zajistena integrita zprávy.

Riešenie:

Hash sa pridáva a následne kontroluje na druhej strane. Je použité encrypt then hash. Použil som SHA256 s jednoduchým určeným kľúčom len pre testovanie.

```
424     unsigned char hash[] = "Testing key";
425     sha256_hmac(hash, 11, (const unsigned char*)output, line.size(),output + line.size() , 0);
426
427     m_p2p.write( (const char*)output, line.size() + 32 );
```

```
177         bool testing = true;
178         for(int i = 0; i < 32; i++)
179         {
180             if(temp[i] != (unsigned char)buffer[length - 32 + i])
181             {
182                 testing = false;
183             }
184         }
185
186         if(!testing)
187         {
188             std::cout << "Hash doesnt match, msg could be changed." << std::endl;
189         }
190
```

Problem:

Výměna klíčů: Problem identification: C_x(code, implementation) Severity: high Practicability: middle

Description of the problem: Při výměně klíču dochází ke dvoum vecem. Zaprve randomu se nastavi vždy stejny seed před generováním což zapříčini to že se vygenrují 2 uplně stejné klíče. Navíc kliče by se nemeli jentak přidat za sebe. A Za druhé mělo by se posílat větší množstvi dat pro kliče ne pouze 16 bit za každeho klienta. Proposed solution: Při výmeně kliču vymyslet jiný způsob vygenrování noveho kliče a pridat nejakou hash funkci na propojeni 2 kliču v jeden

Riešenie:

Kľúče sa generujú pomocou funkcie ctr_drbg_random z polarSSL. Dĺžka kľúča je 128

```
entropy_init( &entropy );
if( ( ret = ctr_drbg_init( &ctr_drbg, entropy_func, &entropy,
    (unsigned char *) pers, strlen( pers ) ) ) != 0 )
{
    printf( " failed\n ! ctr_drbg_init returned -0x%04x\n", -ret );
    return;
}

if( ( ret = ctr_drbg_random( &ctr_drbg, part_key, 128 ) ) != 0 )
{
    printf( " failed\n ! ctr_drbg_random returned -0x%04x\n", -ret );
    return;
}
```

Kľúče sa navzájom vymenia a nakoniec sa zahashuju do výsledného 128bit kľúča.

```
220             key[i] = buffer[i + 4] ^ part_key[i];
221             part_key[i] = '\0';
222             buffer[i + 4] = '\0';
223         }
```

Problem:

Strcpy a Strncpy: Problem identification: C_x(code, implementation) Severity: high Practicability: hard (directly by external attacker) Description of the problem: Při generování klíču(v funkcích startRead() a ReceiveData2()) se používaji funkce strcpy a strncpy a to mi nepřijde vhodne pro pole dat. Při bližším zkoumani to ma nepředvídatelné chovaní chování. Proposed solution: Pro prácí s poli dat používat funkce memcry.

Riešenie:

Strcpy a Strncpy sa už vôbec nepouživájú. Pokiaľ je treba niečo skopírovať tak som použil std::copy (považoval som to za c++ náhradu memcpy)

```
171         std::copy(part_key, part_key + 128, temp + 4);
172         m_p2pSSL.write((const char*)temp, 132);
173
```

Problem:

Dešifrování a šifrování tabulky: Problem identification: C_x(code, implementation) Severity: eazy Practicability: hard (directly by external attacker) Description of the problem: Při šifrování a dešifrování nepredpočítava(beží na stejnem vlakně) a dochází ke značnemu spomaleni operace šifrování a dešifrování. Navíc podle me jsou potreba vzlašt tabulky pro šifrování a dešifrování což zde není, takže komunikace nemuže fungovat. Proposed solution: funkce pro predpočítavání tabulek pro sifrování a defifrování by male bežet na jinem vlákně.

Riešenie:

Tabulky pre šifropvanie aj dešifrovanie sú . Používam dve tabuľky, keď sa prvá využije do 50% tak sa začne na vlákne prepočítavať druhá, a predá sa jej counter z prvej. Keď sa tabuľka blíži ku koncu tak sa vymenia a tak stále dokola.

```
53      TableOne * toc = new TableOne(this);
54      connect(this, SIGNAL(tableOneSignal(prepare_table*)), toc, SLOT (TableOneComp(prepare_table*)));
55      toc->start();
```

```
370     if(prepare2)
371     {
372         std::cout << "preparing table 2, counter
373         tableTwo->counter = tableOne->counter;
374
375         emit tableOneSignal(tableTwo);
376         prepare2 = false;
377     }
```

```
5   TableOne::TableOne(QObject*)
6   {
7   }
8
9   void TableOne::TableOneComp(prepare_table * ta
10  {
11      ecb_prepare_table(table);
12  }
13
```

Problem:

Dešifrování a šifrování: Problem identification: C_x(code, implementation) Severity: none Practicability: hard (directly by external attacker) Description of the problem: Šifrování probiha i při prvnotni vymene klíču a to na random datech protože funkce by mela asi mit 2 vetve jednu pro to když prijde klič a jednu pro to když prijdou data.

Riešenie:

Použil som 2 rôzne sockety (QsslSocket a QTcpSocket) s rôznymi portami, takže výmana kľúčov prebehne cez SSLsocket, ktorý sa následne zatvorý a komunikácia následne prebieha na QtcpSockete.

Problem:

Problem identification: Deviation from initial design

Severity : High

Practicability :

Description of the problem : No client to client communication established. Symmetric encryption is not used.

Proposed solution :

Riešenie:

Všetko už funguje. (Testované zatial iba na localhoste).

Problem:

Problem identification: Improper ordering of function arguments

Severity : High

Practicability : In every call to Send Message

Description of the problem : crypto_crt/crypto.cpp:31 xor_table() function. Parameters do not match with arguments when used in qtClient/Client.cpp:203

Proposed solution :

Riešenie:

Dlho som sa trápil, a nevedel prísť na to prečo mi to padá...

Problem:

Problem identification: Memory Leak

Severity : high

Practicability :

Description of the problem : qtClient/Client.cpp:33 In client constructor a local variable is allocated memory and it is neither used nor freed.

Proposed solution : Code refactoring

Riešenie:

To bol pravdepodobne struct prepare_table ktorý je už spominaný vyššie. To isté sa týka toho istého leaku aj v QtClient/server.cpp

Posledná skupina: 410435-Plch_Matej-code_review

Spominaná generacia kľúča (vyriešené).

Threading tiež. Problémy typu velké písmená som neriešil.