**Ivan Dendis, Martin Krivosudský, Ľubomír Viluda – Hrochokobry**

**Obecné chyby:**

Nešlo zkompilovat.

Zmatek v .h souborech k polarssl + nešlo skompilovat na serveru v kuli loggum (ClientThread.cpp:195: error: C2678: binary '<<' : no operator found which takes a left-hand operand of type 'std::ostream' (or there is no acceptable conversion))

Leaky. Nesmazané klíče.

Jinak v kódu se nikdo nestará o smazání naalokované paměti a dochází k leakům.

**Výsledky staticke analizy:**

V celem projektu pamet alokována c-style (MALLOC).

Client.cpp:

Memory leak – promena table

ProtocolMsg.h:

Index i by mel byt kontrolován aby nevypadl z hranic pole.

Class buffer nemá kopírovací konstruktor.

Crypto.cpp:

Memory leak – promena input

Tests.cpp:

Memory leak – promena joined\_table

ClientThread.cpp

Prefered prefix ++ před interatorem

**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.

**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.

**Výměna klíčů:**

Problem identification: C\_x(code, implementation)

Severity: high

Practicability: middle

Description of the problem: Při výměně klíču docházi 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

**Strcpy a Strncpy:**

Problem identification: C\_x(code, implementation)

Severity: high

Practicability: hard (directly by external attacker)

Description of the problem: Při generování klíču(ven funkcích startRead() a ReceiveData2()) se použivaji funkce strcpy a strncpy a to mi nepřijde v hodne pro pole dat. Při bližším zkoumani to ma nepředvídatelné chovaní chování.

Proposed solution: Pro prácí s poli dat použivat funkce memcry.

**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 docházi 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.

**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.