

Ivan Dendis, Martin Krivosudský, Lubomír Viluda – Hrochokobry

Obecné chyby a crashe:

1. Nešlo zkompileovat.

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))

2. Zaseknuti aplikace:

Když se zadá špatné heslo aplikace nadále nereaguje.

Když si Klienti vymění klíče (malokdy se povede), následuje prázdný řádek -> aplikace pada.

Výsledky statické analýzy:

V celém projektu paměť alokována c-style (MALLOC).

Client.cpp:

Memory leak – proměnná table

ProtocolMsg.h:

Index i by měl být kontrolován aby nevypadl z hranic pole.

Class buffer nemá kopírovací konstruktor.

Crypto.cpp:

Memory leak – proměnná input

Tests.cpp:

Memory leak – proměnná joined_table

ClientThread.cpp

Prefered prefix ++ před iterátorem

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 klíče z paměti. Útočník pak může v paměti dohledat klíč.

Proposed solution: Přemazat proměnnou key v destruktoru.

Hash:

Problem identification: C_x(code, implementation)

Severity: high

Practicability: easy (directly by external attacker)

Description of the problem: Nekontrolují se integrity zpráv, útočník může kdykoli změnit zprávu a nikdo to nepozná.

Proposed solution: Přidat hash na konec zprávy a byla zajištěna integrita zprávy.

Výměna klíčů:

Problem identification: C_x(code, implementation)

Severity: high

Practicability: middle

Description of the problem: Při výměně klíče dochází ke dvěma věcem. Zprvu randomu se nastaví vždy stejný seed před generováním což způsobí to že se vygenerují 2 úplně stejné klíče. Navíc klíče by se neměli jen tak přidat za sebe. A za druhé mělo by se posílat větší množství dat pro klíče ne pouze 16 bit za každého klienta.

Proposed solution: Při výměně klíče vymyslet jiný způsob vygenerování nového klíče a přidat nějakou hash funkci na propojení 2 klíčů v jeden

Strepy a Strncpy:

Problem identification: C_x(code, implementation)

Severity: high

Practicability: hard (directly by external attacker)

Description of the problem: Při generování klíče (v funkcích startRead() a ReceiveData2()) se používají funkce strepy a strncpy a to mi nepřijde vhodné pro pole dat. Při bližším zkoumání to má nepředvídatelné chování.

Proposed solution: Pro práci s poli dat používat funkci memcry.

Dešifrování a šifrování tabulky:

Problem identification: C_x(code, implementation)

Severity: easy

Practicability: hard (directly by external attacker)

Description of the problem: Při šifrování a dešifrování nepredpočítava (běží na stejném vlákně) a dochází ke značnému zpomalení operace šifrování a dešifrování. Navíc podle mě jsou potřeba vzlašt tabulky pro šifrování a dešifrování což zde není, takže komunikace nemůže fungovat.

Proposed solution: funkce pro predpočítávání tabulek pro šifrování a dešifrování by měla běžet na jiném vlákně.

Dešifrování a šifrování:

Problem identification: C_x(code, implementation)

Severity: none

Practicability: hard (directly by external attacker)

Description of the problem: Šifrování probíhá i při první výměně klíče a to na random datech protože funkce by měla mít 2 větve jednu pro to když přijde klíč a jednu pro to když přijdou data.

Odhlášení klienta po ztracení spojení:

Problem identification: C_x(code, implementation)

Severity: none

Practicability: hard (directly by external attacker)

Description of the problem: Jakmile klient vypne svoji aplikaci, server si stále drží údaj o tom že je online a ostatní klienti ho dostanou v online listu