Spolehlivá a zabezpečená komunikace v rámci systému pro zákonné odposlechy

FIT VUT Technický report

Michal Kajan, Karel Koranda, Libor Polčák

Technický report č. FIT-TR-2012-007
Fakulta informačních technologií, Vysoké učení technické v Brně

Last modified: 11. prosince 2012

Spolehlivá a zabezpečená komunikace v rámci systému pro zákonné odposlechy

Michal Kajan, Karel Koranda, and Libor Polčák

Vysoké učení technické v Brně, email: {ikajan,ipolcak}@fit.vutbr.cz,
xkoran01@stud.fit.vutbr.cz

Abstrakt Tato technická zpráva popisuje způsob zajištění spolehlivého a bezpečného přenosu dat ve vyvíjeném systému pro zákonné odposlechy v rámci projektu Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace. Součástí zprávy je analýza dostupných přenosových protokolů a možností zabezpečení komunikace pomocí šifrování. Na základě analýzy přenosových protokolů bude spolehlivosti přenosu dosaženo pomocí protokolu TCP. Dále následuje analýza vlastností bezpečnostních protokolů využitelných pro zabezpečení přenášených dat. Z těchto možností byly vybrány protokoly TCP pro spolehlivý a SSH pro zabezpečený přenos dat. V současnosti je vyvíjen model komunikace, který bude později sloužit jako vzor pro implementaci komunikace pro mikro-sondu.

1 Úvod

Při vyšetřování Internetové kriminality je nutné zajistit dostatečně průkazné důkazy, aby byly nezpochybnitelné při soudním řízení. Při odposlechu síťových dat je tedy nezbytné zajistit, aby byla kopie všech přenášených dat doručena ze sondy, která data odposlouchává přes další prvky systému pro zákonné odposlechy (Lawful Interception System – LIS) až k orgánům činným v trestním řízení (Law Enforcement Agency – LEA).

Pro zachování soukromí odposlouchávaných osob, ale také aby nebylo možné zmařit vyšetřování podezřelých osob prozrazením vykonávání odposlechu, není možné odposlechnutá data přenášet v čitelné podobě. Proto je nutné odposlechnutou komunikaci zabezpečit šifrovacími mechanismy.

Tato technická zpráva se zabývá zajištěním spolehlivého a zabezpečeného přenosu mezi jednotlivými prvky LIS umístěného v síti poskytovatele připojení k Internetu, či poskytovatele služeb. Cílem technické zprávy je nalézt vhodné prostředky pro zajištění bezpečného a spolehlivého přenosu dat z mikro-sondy do sběrného stanoviště v rámci LIS vyvíjeného v rámci projektu Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace.

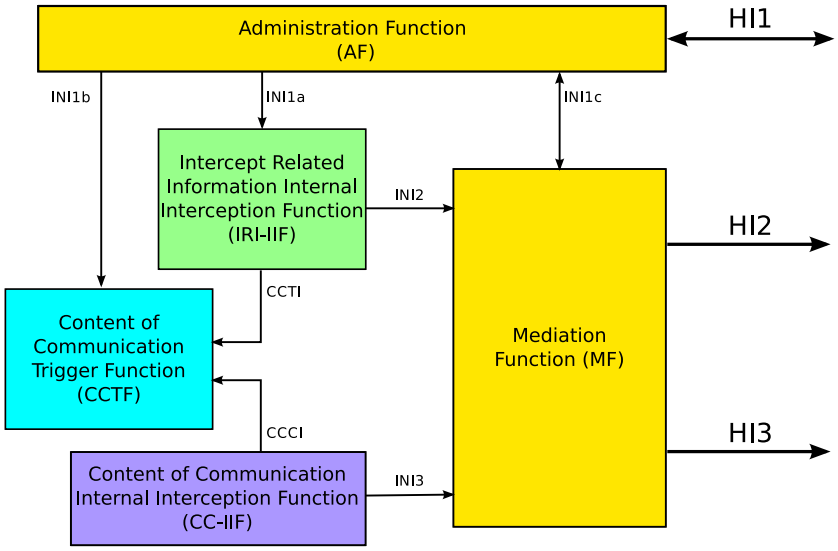
Sekce 2 popisuje požadavky a doporučení specifikované v normách ETSI. Popis protokolů vhodných pro zajištění transportu dat ze sondy je obsažen v sekci 3. Protokoly zajišťující zabezpečení přenášených dat jsou popsány v sekci 4.

Plánovaný výsledek projektu Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace a výběr protokolu pro spolehlivou a zabezpečenou komunikaci popisuje sekce 5. Dále je v této sekci popsán softwarový model protokolu TCP, který slouží pro navržení implementace protokolu TCP pro mikro-sondu a shrnut postup pro hardwarovou implementaci. Poslední sekce 6 shrnuje obsah práce a nastiňuje budoucí směrování práce.

2 Požadavky norem pro zákonné odposlechy

Tato sekce se zabývá referenčním modelem architektury ETSI pro zákonné odposlechy. Cílem sekce je podat čtenáři stručné informace o jednotlivých částech LIS a rozhraních mezi nimi. Podrobnější informace o normách ETSI pro zákonné odposlechy jsou obsaženy v technické zprávě FIT-TR-2012-008 [35].

Referenční architektura ETSI [12] zachycená na obrázku 1 se skládá z pěti spolupracujících částí (bloků): *Administration Function* (AF), *Internal Interception Function Internal Interception Function* (IRI-IIF), *Content of Communication Trigger Function* (CCTF), Content of Communication Internal Interception Function (CC-IIF) a *Mediation Function* (MF). Bloky jsou propojeny rozhraními pojmenovanými jako *Internal Network Interface* (INI), *Content of Cummunication Trigger Interface* (CCTI) a *Content of Communication Control Interface* (CCCI).



Obrázek 1. Referenční model LIS publikovaný ETSI [12]

Tato technická zpráva se zabývá primárně rozhraním *Internal Network Interface 3* (INI3), které slouží k zasílání odposlechnutého obsahu komunikace. Ten je získáván v rámci CC-IIF zkopírováním paketů přenášených sítí a je zasílán MF, kde je dodatečně zpracováván, přeformátován a dále přeposlán LEA.

Samotná norma ETSI [12] nepředepisuje přímo komunikační protokol ani způsob zajištění spolehlivosti přenášených dat mezi prvky systému LIS. Norma pouze požaduje, aby generovaná data (v IRI-IIF) a zachycená data v (CC-IIF) byla přenesena beze ztráty a v případě zachycených dat beze zásahů do původních síťových paketů. Norma ETSI dále doporučuje použití specializovaných protokolů pro přenos dat do sběrného stanoviště (MF).

Norma ETSI předkládá pro spolehlivý přenos dat několik příkladů, jak využít standardizované protokoly pro zapouzdření přenášených dat. Mezi tyto protokoly patří protokol UDP a Real-Time Transport Protocol (RTP) a jejich spojení s mechanismy pro zajištění kvality služeb (QoS). Společnou vlastností těchto protokolů je však to, že ani jeden z nich není schopen zajistit spolehlivou komunikaci a ta musí být řešena jiným způsobem na úrovni aplikace nad těmito protokoly.

Norma ETSI [12] nevyžaduje použití konkrétního způsobu zabezpečení přenášených dat, pouze udává požadavek na jejich ochranu šifrováním. Příklady použitých protokolů pro šifrování přenášených dat je možné nalézt v normách [14,11], ve kterých je doporučeno použití protokolů TLS nebo IPsec. Tyto normy se však zabývají rozhraními HI pro přenos dat směrem k orgánům činným v trestním řízení.

Norma [13] představuje bezpečnostní model doporučený pro použití v systému LIS. Ten zahrnuje popis zabezpečení na úrovni rozhraní v systému, mechanismy omezení přístupu k prvkům a rozhraním LIS, zaznamenávaní událostí, typy útoků a ochranu proti nim a další. Nicméně pro ochranu dat doporučuje jejich šifrování a zajištění integrity bez konkrétních doporučení použitých algoritmů. Nechává prostor pro řešení tohoto problému na tvůrcích implementace systému LIS a doporučuje se řídit standardizovanými a obecně akceptovanými protokoly, které mají podporu v rozšířené a dostupné implementaci a také jsou považovány za dostatečně bezpečné v době implementace systému LIS.

Normy ETSI jsou tedy benevolentní a nechávají na implementátorovi konkrétního LIS, aby si vybral vhodné protokoly pro zajištění spolehlivého a bezpečného přenosu dat. V následujících sekcích 3 a 4 představíme protokoly vhodné pro tyto účely.

3 Zajištění spolehlivosti komunikace

V následujícím textu budou přiblíženy protokoly a jejich vlastnosti, které se používají na transportní či vyšší vrstvě na přenos dat a vycházejí z doporučení normy ETSI [12] pro přenos zachycených dat od bloků IRI-IIF a CC-IIF. Přiblížíme si protokoly UDP a jeho varianty, protokol RTP a protokol TCP.

3.1 Protokol UDP

Protokol UDP byl definován organizací Internet Engineering Task Force (IETF) v RFC 768 [37]. UDP neposkytuje žádné mechanismy pro vytvoření spojení, neobsahuje žádný způsob detekce ztráty paketů, znovu zasíláni ztracených dat, řízení toku odesílaných dat, ani doručování ve správném pořadí. Ve své hlavičce obsahuje pouze identifikaci zdrojového a cílového portu, délky přenášených dat a kontrolní součet. UDP bývá implementován v jádře běžných operačních systémů a umožňuje tak aplikacím s důrazem na rychlost doručení dat realizovat síťovou komunikaci.

3.2 Varianty protokolu UDP

Pro svou jednoduchost bylo pro UDP navrženo několik rozšíření, jejichž vlastnosti budou popsány v následujících odstavcích. Analýza pokrývá několik protokolů, jejichž vlastnosti byly studovány v kontextu přenosu dat v LIS.

UDP Lite

The Lightweight User Datagram Protocol (UDP-Lite) [29] byl navržen pro aplikace tolerující částečně poškozené datové pakety (chyba zjištěna během výpočtu kontrolního součtu vede k zahození celého datagramu UDP). Protokol UDP-Lite zavádí mechanismus částečného pokrytí dat kontrolním součtem — data se rozdělí na část pokrytou a nepokrytou kontrolním součtem. Pokrytí je definováno v hlavičce datagramu. Chyba zjištěna v nepokryté části pak nevede k zahození všech příchozích dat. Je možné definovat úplné pokrytí dat kontrolním součtem, v tom případě je chování identické standardnímu protokolu UDP. Takové chování může být výhodné pro multimediální aplikace. Tento způsob přenosu dat však ovlivňuje nakládání s daty již od linkové úrovně, vyžaduje úpravy v jádře operačního systému či ovladačů síťového rozhraní.

Reliable Data Protocol

Reliable Data Protocol [43,34] je dalším příkladem vylepšení původního protokolu UDP. Spočívá v rozšíření o některé vlastnosti typické pro protokol TCP [36]. Není však schválen ani navržen pro proces standardizace, je určen pouze pro experimentální účely. Také nejsou běžně dostupné implementace RDP.

Tsunami UDP

Nejedná se o opravdový protokol ve smyslu jeho definice a standardizace. Je to *aplikační nadstavba* [3] nad protokolem UDP a TCP určena pro přenos dat vysokorychlostními sítěmi, ve kterých dochází ke vzniku velké latence mezi odesláním a doručením paketů. Komunikace je rozdělena do dvou částí: řídicí a datovou. Řídicí kanál se vytváří nad protokolem TCP, přičemž řídicí komunikace je velmi málo. Samotný přenos dat je pak realizován pomocí protokolu UDP. Protokol Tsunami UDP je určen pro soubory předem známé velikosti, které se přenášejí v blocích stejné velikosti. Pomocí žádostí přes řídicí kanál je možné využít i znovu

zaslání ztracených dat. V řídicím kanálu se také zasílají statistiky z dosavadního průběhu přenosu a odesílatel je používá k přizpůsobení rychlosti přenosu.

Protokol Tsunami UDP využívá standardní protokol UDP a nevyžaduje tedy zásah do ovladačů ani úpravy jádra operačního systému. Přenosy jsou realizovány s ohledem na zajištění základní integrity dat, prioritu je však možno změnit i s ohledem na rychlost.

Realtime Tsunami UDP

Oproti předchozí variantě se liší je minimálně — úpravy se týkají přenosu dat, u kterých není známa dopředu výsledná velikost. Používal se pro přenos dat navzorkovaných z rádiového signálu v satelitu observatoře [2]. Namísto klasického přístupu k souboru na straně odesílatele se přistupuje k unixovému *zařízení*, které je zdrojem nekonečného proudu dat.

Při zahájení přenosu příjemce specifikuje množství požadovaných dat a časovou značku počátku vzorkovaných dat. Odesílatel pracuje s kruhovým bufferem, který se při zastavení přenosu přepisuje novými navzorkovanými daty. Rychlost vzorkování je konstantní, během přenosu se data posílají neměnnou rychlostí.

3.3 Protokol RTP

Dalším protokolem zmiňovaným v normách ETSI pro použití v LIS je protokol Real-Time Transport Protocol (RTP [42], nezaměňovat s protokolem Cisco Reliable Transfer Protocol, který je součástí směrovacího protokolu EIGRP [1]). Protokol RTP využívá služeb *jiných transportních protokolů*. S protokolem pro výměnu dat byl současně navržen protokol pro řízení komunikace RTCP (je součástí specifikace RTP). Řídicí protokol slouží k přenosu řídicích dat, zasílání hlášení od příjemců statistických informací pro přizpůsobení charakteru odesílaní dat. Řídicí informace jsou vyměňovány obousměrně. RTP byl navržen speciálně pro účely přenosu audio a video dat pro IP sítě, celá specifikace protokolu je orientována na pojmy a mechanismy týkající se přenosu multimediálních dat.

V komunikaci RTP vystupují dvě komunikující strany — vysílač a přijímač. Úlohou vysílače je zachytávat audiovizuální data, dělit je do rámců pro zpracování enkodérem kódujícím podle určitého algoritmu a pak je rozdělit do RTP paketů.

Přijímač je z pohledu definice a implementace složitější. Po přijetí paketů ověřuje jejich platnost, řadí dle časové značky a ukládá do fronty příslušející danému vysílači. Z těchto paketů se vytvářejí původní rámce audio/video dat, ty jsou umístěny do další fronty pro odstranění časových mezer vzniklých přenosem sítí. U přijímače dochází k opravě chybných rámců, jejich následnému dekódování, úpravě časových informací potřebných pro přehrávání a následné prezentace do podoby přehrání zvukových stop nebo video dat. Mechanismus přenosů nad protokolem RTP se týká dat určených k okamžitému přehrávání, ne jejich uložení a následnému zpracování.

Součástí specifikace RTP jsou i další body komunikace, jako překladače a směšovače, které slouží ke konverzi mezi různými kodeky multimediálních dat a kom-

binování dat z více zdrojů do jednoho výstupu. Tyto body nejsou koncové body komunikace, mohou pouze tvořit součást komunikační cesty.

Jak již bylo řečeno, protokol RTP využívá jiných transportních protokolů. RTP pro přenos dat využívá protokol UDP (TCP není vhodný pro přenos multimediálních dat přehrávaných v reálném čase, jeho použití však nic nebrání a za jistých okolností je možné TCP využít). RTP rozšiřuje možnosti UDP protokolu o:

- detekci ztráty
- hlášení kvality příjmu odesílateli
- synchronizaci času, resp. přenášených multimediálních dat po přijetí (video a zvuk), ne časovou synchronizaci koncových bodů
- identifikace zdroje a obsahu
- značkování významných událostí v proudu dat

Jak je patrné z uvedeného výčtu, jediný bod, který se týká spolehlivosti přenosů je první a ostatní v LIS a pro charakter jeho dat nenacházejí využití.

Specifikace protokolu RTP se však orientuje na postupy opravy audio a video dat z pohledu použitých kódovacích a dekódovacích algoritmů. Tato data, na rozdíl od většiny ostatních dat, nejsou náchylná na částečnou ztrátu či poškození během přenosu, protože poškození dat často lidské smysly nedokáží rozeznat. Specifikace protokolu se orientuje na přenos dat pro jejich nepřerušený proces přehrávání, v případě jejich ztráty či poškození pak během rekonstrukce dat dekodér vyprodukuje výstup s nižší kvalitou. V naprosté většině případů to vyhovuje a je v souladu i s očekáváními uživatele, který raději pro multimediální data upřednostní výstup s nižší kvalitou, než žádný.

Jako jednoduchý příklad ochrany dat před poškozením možno zmínit použití tzv. dopředného opravného kódu, který funguje na principu přidání redundantních dat k přenášeným datům a umožňuje tak u příjemce detekovat a opravit poškozená data. Tento opravný kód je založen na použití *paritních slov*[1] mezi skupinou datových slov. Využívá se jednoduchá logická operace XOR pro výpočet chybějícího slova. Také se využívá několik postupů zasílání paritních paketů: jeden paritní paket vytvořený nad několika datovými pakety, nad dvojicí datových paketů nebo vícestupňové generování paritních paketů. Mechanismus znovu zaslání paketů není v protokolu RTP definován, pouze v tzv. rozšiřujícím profilu [33]. Podobně nepodporuje ani řízení zahlcení během přenosu.

Specifikace protokolu obsahuje i popis bezpečnostních mechanismů týkající se jak datových, tak řídicích přenosů. V případě datových paketů se šifruje celý jejich obsah. V případě paketů řídicích se na začátek přidává náhodný 4-bajtový prefix (z důvodu velkého počtu fixních položek v hlavičce, což by mohlo útočníkovi výrazně zjednodušit práci s prolomením šifrovacího klíče). Protokol předpokládá použití sdíleného klíče, jeho distribuce je ale mimo definici RTP a řeší se jinými protokoly. Každá implementace protokolu RTP by měla podporovat

[1] Paritní pakety se přenášejí v oddělených kanálech — jeden kanál je datový pro přenos paritních paketů, druhý pro řízení přenosu paritních paketů. Ve výsledku celou komunikaci pomocí RTP můžou tvořit až 4 kanály, na 4 oddělených portech.

šifrovací algoritmus DES, ale vzhledem k jeho slabinám se doporučuje podpora implementace algoritmů 3DES nebo AES.

Byl navržen [8] i rozšiřující bezpečnostní profil, který obsahuje několik změn oproti původně navrhovaným bezpečnostním mechanismům v RTP protokolu. Týká se např. odstranění náhodného prefixu u řídicích přenosů a standardní použití šifrovacího algoritmu AES. Distribuce klíčů je řešena pomocí protokolu SDP [19].

3.4 Protokol TCP

Spolehlivost u protokolu TCP je dána zajištěním hned několika vlastností: pozitivním potvrzováním pomocí číslování bajtů (každý bajt má své sekvenční číslo), použitím časovačů pro detekci ztráty dat na síti a jejich znovu odesíláním, přeskládáváním přijatých datových segmentů do správného pořadí a až následné odevzdání cílové aplikaci a mechanismy pro přizpůsobení rychlosti přenosu aktuálnímu stavu sítě s cílem dosáhnout co nejrychlejších přenosů, zamezit ztrátě dat a zahlcení sítě. Podobně jako jiné protokoly, i protokol TCP chrání přenášená data kontrolním součtem. Komunikující body si vytvářejí na začátku komunikace virtuální spojení, dohodnou si podporované vlastnosti implementace TCP a udržují toto spojení až do jeho skončení. Pro přenosy dat tímto protokolem je typické stavové zpracování — spojení přechází od svého vytvoření až do ukončení několika stavy s vymezeným chováním.

Přenos dat je v protokolu TCP řízen tzv. *posuvným oknem*, jehož velikost je příjemce dat ohlašuje během celého spojení. Protokol TCP rozlišuje dva typy datových přenosů — prvním je přenos malých bloků dat (interaktivní přenosy, např. pro vzdálenou konzoli) a druhým je přenos velkých bloků dat, kde se využívají mechanismy pro efektivní využití dostupné šířky pásma a rychlý přenos dat. Nicméně, vždy platí, že odesílatel nesmí odeslat více dat, než dovoluje velikost okna (t.j. velikost bufferu) ohlášena příjemcem.

Okno je posuvné z toho důvodu, že neustále dochází k posuvu levé a pravé hranice okna. Levá hranice určuje odeslaná a potvrzená data, oblast mezi levou a pravou hranicí odeslaná a ještě nepotvrzená data a data, které je možno odeslat. Oblast za pravou hranicí určuje data, která není možno odeslat, protože to nedovoluje velikost příjemcova posuvného okna. Posuv hranic okna je zajištěn příjmem tzv. potvrzovacích segmentů (dříve zmíněné pozitivní potvrzování), která znamenají přijetí a zpracování dat daných potvrzeným sekvenčním číslem. Velikost okna se může v průběhu přenosu měnit (dokonce i zmenšit na nulu).

Při přenosu dat se uplatňuje několik technik: na straně příjemce se uplatňuje *zpožděné doručování potvrzování*, kdy příjemce nezasílá potvrzení o přijatých datech hned, ale čeká určitý čas (typicky 200 ms), aby spolu s potvrzením mohla být zaslána nějaká data. V případě příjmu více datových segmentů od odesílatele se toto čekání neuplatňuje, po přijetí *druhého* datového segmentu se pošle potvrzení hned (nezávisle na připravenosti dat k odeslání) a zruší se časovač na zpožděné doručení potvrzení. V případě odesílatele se uplatňuje jednoduchý *Naglův algoritmus*, který u odesílatele nedovoluje odeslat segmenty nových dat, pokud má odesílatel samotný ještě nepotvrzená data. Pokud však má dostatek

dat (tím se myslí maximální velikost odesílaného segmentu), tento požadavek se neuplatňuje a další data mohou být odeslána, ale s ohledem na velikost okna u příjemce[2].

Při výměně segmentů nestejné velikosti může dojít k *syndromu hloupého okna* způsobeného odesílatelem, či příjemcem. V případě odesílatele tento problém nastává tehdy, když posílá malé segmenty dat, i když je možno počkat a následně odeslat větší objem dat v jednom segmentu. U příjemce syndrom hloupého okna nastává v případech, když v potvrzovacích segmentech ohlašuje malé velikosti svého dostupného okna, i když by bylo možno počkat a ohlásit větší velikost. Samotné zpožděné doručování potvrzování a Naglův algoritmus tomuto problému přímo nezabrání, ale specifikace protokolu TCP obsahuje posloupnost kroků, jak řešit odesílaní dat a potvrzení tak, aby problému s „hloupým oknem“ nedocházelo. Zasílání krátkých segmentů dat vede k velmi neefektivnímu využití přenosového pásma, kde se navíc velká část spotřebuje pouze na režii samotného protokolu.

TCP používá několik časovačů, např. pro již zmíněné zpožděné doručování potvrzení nebo po zmenšení velikosti okna u příjemce na nulu (pak se odesílatel opakovaně dotazuje na změnu velikosti). Další instance časovačů se používají pro vytváření spojení a čekání na odpověď a také během procesu ukončování spojení. Nejčastěji používaný je časovač RTO (angl. *Retransmission TimeOut*) používáný pro čekání na potvrzení odeslaných segmentů.

V případě přenosu dat může dojít ke ztrátě jak dat samotných, tak potvrzení o přijetí. Proto odesílatel ve svém odesílacím okně udržuje data odeslaná, ale ještě nepotvrzená. Znovu zaslání těchto dat je podmíněno vypršením časovače RTO (nebo také strukturou příjmu potvrzení, viz níže). Nastavení časovače RTO je v čase proměnlivé a odesílatel si jeho hodnotu neustále přizpůsobuje chování sítě. Přesný způsob výpočtu a nastavení počátečních hodnot jsou součástí specifikace protokolu TCP[3]. V případě znovu odeslání dat může nastat další problém související s nastaveními parametrů pro správný výpočet časovače RTO. Problém spočívá ve správném určení toho, k jakému datovému segmentu se přijaté potvrzení váže – zda je to původní segment nebo znovu odeslaný. Tento problém se řeší *Karnovým algoritmem* skládajícím se ze dvou částí. První část řeší uvedenou nejednoznačnost tak, že parametry pro aktualizaci hodnoty časovače RTO se nezmění v případě příjmu potvrzení pro znovu zaslaný segment. Druhá část řeší exponenciální nárůst hodnoty časovače. Hodnota časovače se zdvojnásobuje do doby, než bude přijato potvrzení pro první segment dat, který nebyl znovu odeslán[4]. Při vypršení časovače se také aktivují další mechanismy pro snížení rychlosti přenosu dat. Tyto mechanismy si přiblížíme v následující části.

[2] Tento algoritmus se však v některých případech neuplatňuje, např. v případě přenosů vzdálené plochy, kde je vyžadována co největší interaktivita i při každém pohybu myší.

[3] Žádná z rozšířených implementací však uvedené předpisy pro RTO nedodržuje [39].

[4] Problém s jednoznačností a nastavením správných hodnot je možné řešit i tzv. *selektivním potvrzováním* a časovými značkami. Vyžaduje to však podporu na obou stranách komunikace a pro naši implementaci tyto vlastnosti prozatím neuvažujeme.

Protože nejdůležitější částí v přenosech zachycených dat v LIS je přenos již filtrovaných dat, v síti ISP se však může nacházet zachytávacích sond několik a současně mohou agregovat vstup z několika svých vstupních rozhraní. To klade jisté požadavky i na přenosový protokol, aby nedošlo k zahlcení sítě samotného poskytovatele a zároveň aby nebyl přerušen tok potřebných dat do zařízení MF. Protokol TCP obsahuje ve své specifikaci mechanismy, pomocí kterých dokáže řešit přenos velkých bloků dat a neustále sledovat vlastnosti sítě k přizpůsobení rychlosti zasílání dat. Velikost bloku dat, který je možno přenést v jednom segmentu TCP je dána jak ohlášením ze strany příjemce během zahájení spojení (bez explicitního určení se nastaví pouze 536 bajtů), tak pomocí MTU (maximální velikosti přenášených dat na lokální lince) a PMTU (maximální velikosti přenášených dat na cestě mezi komunikujícími stranami).

Na řízení přenosu dat se v TCP podílejí 4 algoritmy: *slow start, congestion avoidance, fast retransmit a fast recovery*, v jednom okamžiku je aktivní právě jeden z nich. Zavádí se další proměnná, tzv. okno zahlcení, které určuje, kolik dat je možné ještě odeslat (velikost okna zahlcení se neustále během spojení mění; snahou je, aby odesílatel nezahltil příjemce od začátku množstvím datových segmentů TCP pokrývajících jeho buffer). Na začátku přenosu se tak uplatňují dvě fáze: slow start a congestion avoidance (v tomto pořadí, přechod do druhé fáze je dán překročením hranice nastavené velikosti okna zahlcení).

Rozdíl je ve způsobu zvětšování velikosti okna zahlcení. Ve fázi slow start okno roste o jeden MSS (*Maximum Segment Size* – Maximální velikost segmentu) s každým přijatým potvrzením. Ve fázi congestion avoidance k takovému rychlému nárůstu nedochází. Počítají se potvrzené bajty (nebo segmenty) a pokud se potvrdí všechny až do současné velikosti okna zahlcení, toto se zvětší o jeden MSS. V případě vypršení časovače RTO se pokračuje znovu od fáze slow start.

V průběhu přenosu odesílatel sleduje počty opakovaných potvrzení zaslaných příjemcem. Důležitým okamžikem je přijetí *třetího duplicitního* potvrzení pro stejný datový segment[5]. Odesílatel zašle požadovaný chybějící segment, nastaví novou hranici velikosti okna zahlcení pro přechod mezi slow start a congestion avoidance (fáze fast recovery) a vysílaní dat bude pokračovat fází fast retransmit. Po přijetí potvrzení pro znovu zaslaný segment a současně potvrzení pro případná mezitím odeslaná data fáze končí a pokračuje se fází slow start.

3.5 Shrnutí vlastností přenosových protokolů

V předchozích částech byly přiblíženy vlastnosti protokolů zmiňovaných v normách ETSI jako použitelné pro spolehlivý přenos odposlouchávaných dat. Jak ale vyplývá z uvedených vlastností ani jeden z doporučených protokolů či jeho variant neřeší zásadní problémy spojené s transportem dat v síti. Protokol UDP umožňuje pouze doručování dat cílové aplikaci naslouchající na daném portu, protokol UDP Lite je pouze upravenou variantou standardního UDP protokolu, přičemž úpravy nejsou použitelné pro citlivá odposlouchávaná data v LIS, protokol Reliable UDP je pouze experimentální a podpora by vyžadovala zásah

[5] Vlastnost duplicity popisuje specifikace TCP protokolu.

do jádra operačního systému, Tsunami UDP spolu s Realtime Tsunami UDP jsou zase aplikační nadstavby a použitelná by byla pouze jeho druhá varianta — nicméně, stále se jedná o ten samý protokol UDP rozšířený eventuálně o jednoduchou detekci chybějícího paketu a znovu zaslání.

Protokol RTP již patří do skupiny velmi propracovaných (aplikačních) protokolů, jeho použitelnost se však omezuje na multimediální aplikace. I navzdory své propracovanosti neobsahuje mechanismy potřebné pro data v LIS. Pro multimediální data je možné pomocí RTP detekovat ztrátu (obvykle ale bez znovu zasílání, protože to není v oblasti multimediálních dat potřebné) či jednoduché zabezpečení dat paritou či jinými mechanismy. Ani tyto vlastnosti však nestačí k tomu, aby byl tento protokol využit pro spolehlivé přenosy dat v LIS a jeho použitelnost je diskutabilní. Výjimkou pro inspiraci mohou být snad mechanismy zabezpečení pomocí šifrování, i pro ty však možné zvolit některý z dostupných protokolů, který otázku zabezpečení řeší komplexně.

Protože z pohledu normy ETSI se jedná pouze o doporučení a norma nechává volnost pro konečné řešení spolehlivosti přenosu dat, náš zájem se také soustředil na analýzu vlastností protokolu TCP a možností jeho implementace (nebo nutné části) v prostředí vestavěné sondy. Protokol TCP totiž na rozdíl od výše popsaných protokolů abstrahuje problém přenosu dat a vlastností přenosové sítě od aplikace samotné. Aplikace tak pouze vkládá data do TCP nebo je z ní odebírá a proces přenosu je pro ní plně transparentní. Výhodou je také jeho rozšířenost, podpora ze strany všech operačních systémů a flexibilita ve vzájemné komunikaci dvou různých implementací.

4 Bezpečnost přenášených dat

Pro potřeby přenosu dat souvisejících se zákonnými odposlechy vyvstává problém s jejich zabezpečením. Je potřeba si uvědomit, že data sledovaných entit jsou velmi citlivá a je potřeba zabránit jejich odhalení, jak pracovníky u ISP, v jejichž síti je systém LIS nasazen nebo mimo sítě ISP jinými útočníky.

Tato kapitola stručně popisuje vybrané dnes používané protokoly zajišťující bezpečnost dat přenášených počítačovými sítěmi. Na síťové vrstvě ISO/OSI modelu je to sada protokolů IPsec, na vyšších vrstvách jsou to protokoly SSL/TLS a SSH.

4.1 Sada protokolů IPsec

IPsec je skupina protokolů [26,28,24] určená pro zajištění autentizace, důvěrnosti a integrity dat při komunikaci nejen mezi dvěma uzly, ale i mezi uzly patřící do jedné multicastové skupiny. Nachází se na síťové vrstvě ISO/OSI modelu a umožňuje zabezpečení protokolu IP a vyšších protokolů. Na rozdíl od bezpečnostních protokolů vyšších vrstev (TLS/SSL, SSH) nevyžaduje součinnost s aplikací, jejíž data mají být zabezpečena.

IPsec se skládá ze dvou základních částí. První částí jsou protokoly *IP Authentication Header (AH)* a *IP Encapsulating Security Payload (ESP)*, které definují rozšiřující hlavičky umožňující přenos zabezpečených dat. Tyto protokoly

jsou navržené tak, aby nebyly závislé na použitých kryptografických algoritmech. Oba dva protokoly umožňují použití IPsec ve dvou režimech - transportním a tunelovacím. Druhá část standardu je tvořena protokolem *Internet Key Exchange (IKE)*, který zajišťuje autentizaci, ustanovení a správu klíčů relace.

Standard je součástí protokolu IPv6, je však možné ho používat i v sítích založených na protokolu IPv4.

Spojení - Security Association

Mezi dvěma komunikujícími uzly je sestaveno kryptograficky zabezpečené logické spojení zvané Security Association (SA). Toto logické spojení definuje identifikaci obou stran, a obsahuje dohodnuté kryptografické klíče, sekvenční čísla a další informace. Pro duplexní spojení existují vždy dvě logické SA - pro každý směr komunikace jedna. SA je identifikována tzv. Security Parameter Index (SPI), které v odpovídající hlavičce datagramu IPsec určuje, ke které SA daný datagram patří. Tímto je možné odlišit několik různých spojení mezi týmiž uzly. Každý komunikující uzel si musí udržovat databázi těchto SPI, která se nazývá Security Association Database (SAD).

Základní protokoly

Standard IPsec definuje základní protokoly AH [25], ESP [27] a IKE [20,23,16]. Protokol AH zajišťuje pouze integritu a autentizaci dat, použitými mechanismy umožňuje zabezpečit komunikaci také proti replay útokům. Zajištění důvěrnosti dat poskytuje protokol ESP, který volitelně může zajistit také integritu a autentizaci dat.

Oba dva protokoly mohou být použity samostatně nebo mohou být vzájemně kombinovány. Ačkoli protokol ESP je schopen zajistit také integritu, kvůli povinnému šifrování dat znemožní firewallům a směrovačům nahlédnutí do hlaviček na vyšší transportní vrstvě, což znesnadňuje filtrování provozu. S tímto problémem se AH protokol nepotýká a je díky němu možné zajistit integritu dat tam, kde není vyžadováno šifrování, resp. tam, kde je šifrování hlaviček protokolů transportní vrstvy nežádoucí. Nicméně aby nemohlo dojít k narušení důvěrnosti dat přenášených protokolem AH, je následně nutné využívat jiný bezpečnostní mechanismus poskytovaný například protokoly TLS/SSL, SSH nebo další.

Protokol IKE definuje způsob, jakým je mezi dvěma vzdálenými uzly ustanoveno spojení SA, dále jakým způsobem jsou dohodnuty a obnovovány klíče relace. Protokol vznikl kombinací bezpečnostních protokolů ISAKMP [31] a Oakley [32]. Definuje dvě fáze ustavení spojení. V první fázi je provedena vzájemná autentizace uzlů a ustavení klíčů relace. S využitím takto sestaveného spojení je ve druhé fázi možno vytvořit několik dalších SA používaných pro protokoly AH a ESP.

První fáze protokolu IKE může probíhat ve dvou režimech - agresivním a hlavním. Agresivní režim zajistí vzájemnou autentizaci a ustavení klíče výměnou třech zpráv. Hlavní režim rozšiřuje agresivní režim o další možnosti. Oba dva režimy využívají pro ustavení klíče relace mechanismus Diffie-Hellman [38].

V rámci této fáze je také provedena volba kryptografických algoritmů zabezpečujících důvěrnost a integritu dat.

Druhá fáze pracuje v tzv. rychlém režimu. Slouží pro rychlé vytvoření další IPsec SA, v rámci které se budou vyměňovat zprávy protokolu ESP nebo AH. Přitom je dohodnuto, pro který konkrétní datový tok bude ustavená SA využita.

Režimy

Standard IPsec definuje dva režimy přenosu, které jsou využitelné jak protokolem AH, tak ESP. Transportní režim vkládá hlavičku IPsec mezi hlavičky protokolů síťové a transportní vrstvy a původní hlavička protokolu IP je zachována v nešifrované podobě. Naproti tomu tunelovací režim celý původní datagram protokolu IP zapouzdří hlavičkou IPsec a následně je přidána nová hlavička protokolu IP, která velice často obsahuje jiné než původní IP adresy.

Transportní režim se používá především ve spojení mezi dvěma koncovými uzly. Tunelovací režim se často používá obecně mezi směrovači, případně mezi směrovačem a koncovým uzlem - v takovém případě jsou data často zabezpečena pouze na části cesty mezi koncovými uzly.

Díky oběma režimům je možné mít v rámci jednoho datového rámce i více vrstev IPsec a oba dva režimy se dají kombinovat.

Podporovaná kryptografie

Standard IPsec je z hlediska výběru algoritmů pro zajištění důvěrnosti a integrity dat velice robustní. V současně platném standardu [28] definuje možnosti využití algoritmů *hmac-md5-96, kpdk-md5, hmac-sha1-96, des-mac* a *aes-xcbc-96* pro zajištění integrity, pro zajištění důvěrnosti pak algoritmy *des-iv64, des, 3des, rc5, idea, cast, blowfish, 3idea, des-iv32, rc4, aes-cbc, aes-ctr*. Asi nejupřednostňovanějším šifrovacím algoritmem je v současné době AES, pro informace o zavedení jeho různých forem v IPsec je možné nahlédnout do [15,21,22]. Obecně není díky architektuře IPsec obtížné dodefinovat použití dalších kryptografických algoritmů, protože se předpokládá, že některé časem nebudou či již nyní nejsou některé z nich považovány za dostatečně bezpečné.

Výhody a nevýhody IPsec

Největší výhodou mechanismu IPsec je vysoká robustnost a to, že aplikace není nucena k součinnosti s tímto mechanismem. IPsec umožňuje v rámci jednoho spojení mezi dvěma stanicemi či sítěmi zabezpečit i více datových toků zároveň. Vzhledem ke svému umístění na síťové vrstvě se také chová transparentně k síťovým prvkům pracujícím na stejné vrstvě.

Mezi nevýhody mechanismu IPsec patří nutnost podpory v jádře OS a nesnadná implementace hardwarové podpory standardu. Problémem může být také šifrování informací transportní vrstvy, které mohou být využívány firewally, v sítích založených na protokolu IPv4 pak může nastat problém s jeho využitím v kombinaci s mechanismem NAT. Kvůli obtížné konfiguraci není IPsec také využíván tak široce, jak bylo původně plánováno.

4.2 Protokol TLS/SSL

Protokol TLS [10,24] (a jeho předchůdce SSL [17]) je protokol poskytující zabezpečenou a spolehlivou komunikaci mezi dvěmi komunikujícími aplikacemi. Nachází se na šesté vrstvě ISO/OSI modelu. Na rozdíl od protokolu IPsec neprovádí zašifrování informací transportní vrstvy a nižších. Obecně lze protokol TLS rozdělit do dvou vrstev - vrstvy záznamů (record layer), kterou zajišťuje *Record protocol*, a vrstvy vyšších protokolů. Ty jsou pro TLS definovány čtyři - *ChangeCipherSpec*, *Alert*, *Handshake* a za čtvrtý je považován libovolný aplikační protokol na odpovídající úrovni ISO/OSI modelu, přičemž TLS/SSL zajišťuje pro PDU tohoto protokolu důvěrnost a integritu.

Protokol Handshake

Protokol Handshake slouží k ustanovení relace mezi dvěma vzdálenými aplikacemi typu klient-server. Výměna zpráv mezi klientem, který iniciuje spojení, a serverem probíhá dle následujícího (zjednodušeného) schématu:

1. Klient A žádá o ustavení relace mezi ním a cílovým serverem B. Serveru odešle svůj požadavek spolu s náhodně vygenerovaným číslem R_A a seznamem podporovaných kryptografických protokolů.
2. Server B odpoví klientovi na jeho požadavek, zašle mu náhodně vygenerované číslo R_B, svůj certifikát, díky kterému si klient může ověřit identitu serveru, a zvolí nejsilnější z nabídnutých kryptografických protokolů, který se bude v rámci relace používat.
3. Klient po ověření identity serveru následně odesílá zprávu obsahující náhodné *PreMaster Secret* S zašifrované veřejným klíčem serveru.
4. Klient i server následně vygenerují s využitím S, R_A a R_B *Master Secret*, ze kterého jsou pro ustanovenou relaci odvozeny klíče a inicializační vektory pro šifrovací algoritmy, a klíče pro HMAC funkce zajišťující integritu.

Z uvedeného schématu vyplývá, že je požadováno ověření identity serveru, nikoli však klienta. Protokol však poskytuje volitelnou část autentizace klienta, kterou si může server vyžádat. Vhodný okamžik na uskutečnění této autentizace se v daném schématu nachází mezi body *2.* a *3.*.

Protokol ChangeCipherSpec

Tento protokol obsahuje jedinou zprávu, která se odesílá bezprostředně po ustanovení relace. Zpráva je vzájemně zaslána oběma stranami a slouží k určení okamžiku, od kterého bude všechna následující komunikace zabezpečena dohodnutým způsobem.

Protokol Alert

Zprávy protokolu Alert jsou vzájemně vyměňovány komunikujícími stranami v případě, že je zapotřebí na něco upozornit. V závislosti na závažnosti se může jednat buď o upozornění nebo o chybu. Speciální Alert zprávou je zpráva *closure*, kterou jedna ze stran oznamuje, že již nemá další data k odeslání.

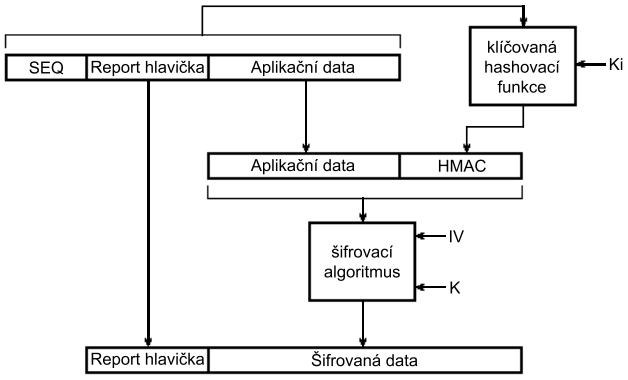
Protokol Record

Protokol Record provádí pouze zapouzdření protokolů vyšších vrstev, definuje hlavičku záznamu. Ta obsahuje především typ zapouzdřené zprávy - zda se jedná o zprávu protokolů Handshake, Alert, ChangeCipherSpec nebo šifrovaná aplikační data.

Protokol také definuje postup zabezpečení aplikačních dat. Ten je následující:

1. K aplikačním datům je přidána hlavička záznamu a sekvenční číslo v rámci aktuální relace.
2. Z celého takto sestaveného rámce se vytvoří HMAC pro zabezpečení integrity dat.
3. K původním datům aplikace se připojí vypočtený HMAC. Takto vzniklá datová struktura je pak vstupem šifrovacího algoritmu, který byl dohodnut při ustavení spojení.
4. K zašifrovaným datům se připojí nešifrovaná hlavička záznamu, která byla vstupem HMAC. Tento výsledek se odesílá příjemci.

Schéma zabezpečení je možné vidět názorně na obrázku 2.



Obrázek 2. Schéma zabezpečení aplikačních dat protokolem TLS/SSL [24]

Podporovaná kryptografie

Nejnovější verze TLS (1.2, RFC [10]) pro účely autentizace serveru standardně používá asymetrický šifrovací algoritmus RSA, případně různé varianty mechanismu Diffie-Hellman. Pro účely šifrování jsou vybrány symetrické šifry *rc4-128, 3des-cbc, aes128-cbc, aes256-cbc*, pro zajištění integrity dat jsou pak využívány algoritmy *MD5, SHA-1 a SHA-256*.

Slabiny TLS/SSL

Vzhledem k tomu, že standard umožňuje použití nejnovějších kryptografických algoritmů, které jsou považovány za bezpečné, lze i o aktuální verzi TLS říci, že zajišťuje velmi bezpečné spojení typu klient-server. Největší slabinou tohoto protokolu je však velmi nejasně či vůbec nedefinovaná forma obnovování klíčů v průběhu ustanovené relace. Při příliš dlouhé relaci toto může kryptoanalytikovi, který bude komunikaci zaznamenávat, poskytnout dostatečné množství dat pro statistický útok na šifrovací algoritmus a následné prolomení zabezpečení. Z tohoto důvodu se zpravidla TLS/SSL využívá pouze pro krátké relace, například aplikačního protokolu HTTP, a pokud není zajištěna výměna klíčů v průběhu relace, nejedná se o vhodný protokol pro zabezpečení déle trvajících spojení.

4.3 Protokol SSH

Protokol Secure Shell (SSH) [30,46,44,47,45,41,9] je určen pro zabezpečení síťové komunikace. Tento protokol umožňuje v rámci jednoho TCP spojení tunelovat více zabezpečených kanálů libovolných dat (např. příkazový řádek na vzdáleném počítači, přesměrování portů jiné TCP aplikace, přenos souborů atd.). SSH zajišťuje autenticitu (ověření identity uživatele nebo zařízení), autorizaci (přidělení oprávnění k přístupu), šifrování přenášených dat (ochranu dat převedením do nečitelné podoby) a jejich integritu (ochranu před jejich modifikací během přenosu). Pracuje na aplikační úrovni a je pro něj dostupných několik implementací (např. knihovna OpenSSH). Pro protokol jsou definovány jak serverová tak klientská část.

Ověření identity uživatele může být založeno na použití hesla přenášeného v zašifrované podobě, nebo veřejného klíče nebo jiných mechanismech založených na systému Kerberos [48], RSA SEcurID Tokenu [5], S/Key jednorazových heslech [18], systému PAM [40] nebo jiných. Způsob autentizace si dojednávají klient a server na začátku spojení na základě své konfigurace. Přidělení práv nastává po ověření identity a je obvykle založeno na nastaveních SSH serveru nebo nastaveních daných pro účet přihlašovaného uživatele. Tunelování je vlastnost SSH protokolu, která umožňuje bezpečné zapouzdření komunikace libovolné aplikace běžící nad TCP[6]. Tunelování je vůči aplikaci úplně transparentní a nezávislé.

Existují dvě navzájem nekompatibilní verze protokolu: SSH-1 a SSH-2. Protože druhá verze nahradila první a původní je v mnoha ohledech zastaralá a překonaná i v bezpečnostních mechanismech, další popis se bude týkat výlučně protokolu SSH-2.

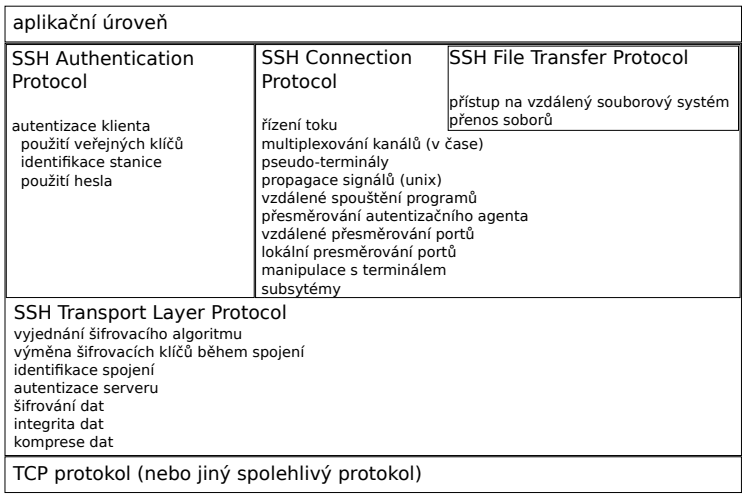
Protokol SSH se skládá ze čtyř hlavních částí, každá z nich je popsána jako samostatný protokol v dokumentech RFC:

– SSH Transport Layer Protocol

[6] Ačkoliv je použitelné pro většinu aplikací, v některých případech je to velmi obtížné a řeší se jinými způsoby, např. pro protokol FTP, kde je vhodné použít speciálního *sftp* klienta.

– SSH Authentication Protocol
– SSH Connection Protocol
– SSH File Transfer Protocol

Existují i další rozšíření, ale výše uvedené jsou klíčové. Protokoly jsou uspořádány jako moduly ve vrstvách (obr. 3) a poskytují příslušné služby, jako by se jednalo o jeden protokol.



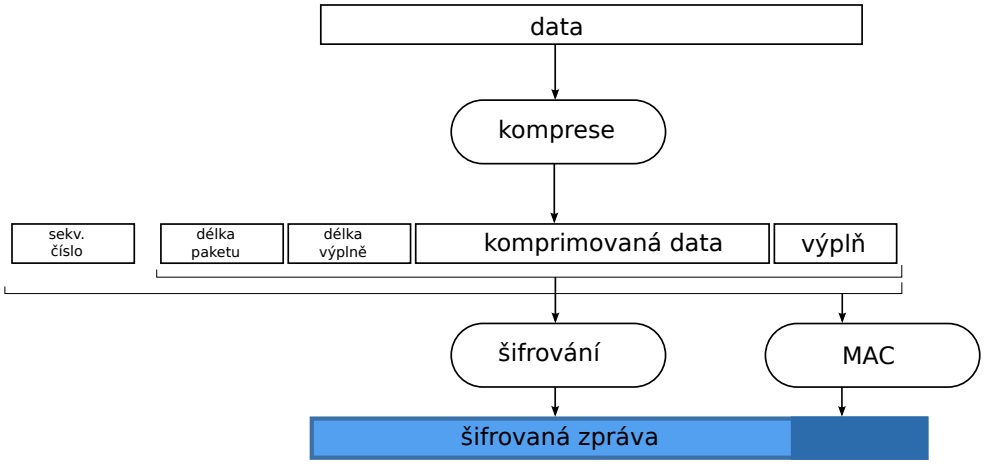
Obrázek 3. Vrstvy protokolu SSH [7].

Modularita umožňuje jednoduché rozšiřování o další vlastnosti. Téměř všechny parametry spojení si jednotlivé strany dojednávají, čímž se zajišťuje interoperabilita různých implementací. Navíc požadavek ze specifikace na povinnost implementace nějaké funkcionality (např. autentizace veřejným klíčem) ještě neznamená povinnost jejího použití. Některé vlastnosti tedy mohou být podle potřeby konfigurovány a celé prostředí přizpůsobeno. Jednotlivé stavební bloky protokolu SSH jsou provázané a navzájem si poskytují služby.

Navázání spojení se uskutečňuje prostřednictvím protokolu SSH Transport Layer Protocol. Dochází k autentizaci na serveru a základnímu šifrování pro vytvoření plně duplexního bezpečného kanálu s klientem. V této fázi server pošle svůj klíč (výměna klíčů se realizuje pomocí mechanismu Diffie-Hellman), kterým prokazuje svoji identitu. Klient musí samozřejmě mít informace o klíčech serverů, ke kterým se připojuje. K tomu jsou využívány dva mechanismy: 1) Klient si udržuje lokální databázi názvu serverů asociovaných s příslušným klíčem serveru. Tento mechanismus nevyžaduje centrální správu ani třetí stranu pro spolupráci. 2) Provázání serveru s daným klíčem je dáno certifikátem od certifikační autority. Klientovi postačuje znát kořenový klíč autority pro ověření identity serverů,

ke kterým se připojuje. Klient pomocí protokolu SSH Authentication Protocol provádí svou autentizaci. Tento protokol specifikuje mechanismy, za pomoci kterých je možné využít několik různých druhů autentizačních mechanismů, ale pro implementaci specifikace vyžaduje pouze jednu — použití veřejného klíče s algoritmem DSS.

Po autentizaci se komunikující strany dohodnou na parametrech přenosu. Obě si vymění seznam podporovaných algoritmů zahrnujících algoritmy pro výměnu klíčů, šifrování, MAC hashovacích funkcí a kompresi. Pro šifrování je možno použít *3des-cbc, blowfish-cbc, twofish256-cbc, twofish192-cbc, twofish128-cbc, aes256-cbc, aes192-cbc, aes128-cbc, Serpent256-cbc, Serpent192-cbc, Serpent-128cbc, arcfour, cast128-cbc*. Pro hashovací funkce je možno využít *hmac-sha1, hmac-sha1-96, hmac-md5, hmac-md5-96*. Pokud je komprese vyžadována, použije se *zlib*, jinak žádná. Klíče pro šifrování, hashování a inicializační vektor jsou vygenerovány ze sdíleného tajného klíče. Možnosti odhalení šifrovacích klíčů dostatečně dlouhým odposlechem šifrovaných dat pro jejich následné odvození se zabraňuje pravidelným obměňováním šifrovacích klíčů (může se změnit i na požádání a změna se může týkat i použitého šifrovacího algoritmu). Šifruje se celý datový paket, včetně políčka MAC, které umožňuje ověření integrity přenášeného paketu. Formát paketů vyměňovaných mezi komunikujícími stranami je na obr. 4.



Obrázek 4. Schéma zabezpečení aplikačních dat protokolem SSH [4].

Po úspěšné autentizaci klient použije protokol SSH Connection Protocol, který umožňuje použít více datových kanálů v jednom zašifrovaném kanálu a navzájem je multiplexovat: přenášet protokol X Windows nebo tunelovat TCP spojení nějaké jiné aplikace, provádět kompresi dat, vzdáleně spouštět programy, propagovat (unixové) signály na druhou strany spojení aj.

Pomocí protokolu SSH File Transfer Protocol může aplikace přistupovat ke vzdálenému souborovému systému a manipulovat se soubory.

Výhody a slabiny SSH

Protokol SSH chrání šifrováním vytvořený kanál před odposlechem (není možné jej jednoduše dešifrovat). Pomocí ověřování identity serveru a klienta kontrolou klíčů zajišťuje odolnost vůči útokům na doménové názvy či zcizení IP adresy a případné přesměrování na útočníkův server. Nepřímo je protokol SSH odolný i vůči útokům Man-in-the-Middle a zcizení spojení (to je však slabina protokolu TCP). Protokol SSH však nepředstavuje všeobecné řešení bezpečnosti přenosů. Typickým příkladem jsou útoky na hesla (ale to je spíš v rovině uživatelského problému, bezpečnost se výrazně zvýší použitím veřejného klíče). Dalším problémem mohou být útoky na transportní vrstvě na úrovni protokolu TCP (např. reset spojení, generování potřebných ICMP zpráv pro zrušení spojení nebo útoky na převzetí běžícího spojení). Nedochází však k odhalení citlivých dat, ani prozrazení přístupových informací, pro tyto případy dojde ze strany SSH k okamžitému ukončení spojení. Takové typy útoků mohou být detekovány zaznamenáváním a přijímáním potřebných bezpečnostních protiopatření. Navzdory tomu v současnosti protokol SSH zajišťuje velmi vysokou úroveň ochrany přenášených dat.

4.4 Shrnutí vlastností bezpečnostních protokolů

V předchozích částech byly přiblíženy vlastnosti v současnosti dostupných a používaných protokolů, které zajišťují ochranu přenášených dat v počítačových sítích. Mnohé vlastnosti těchto protokolů se překrývají (např. používané šifrovací algoritmy) a odlišnosti se týkají způsobu definice komunikace v příslušném protokolu, použití protokolů a jejich nasazení.

Sada protokolů IPsec zajišťuje ochranu dat již na síťové úrovni a tudíž pracuje transparentně vůči všem síťovým aplikacím. Poskytuje velmi robustní a propracované mechanismy pro zabezpečení, vyžaduje však podporu v jádru operačního systému a také velmi netriviální konfiguraci pro vytvoření a správu zabezpečených spojení. Přestože je IPsec součástí definice protokolu IPv6 pro vzájemnou komunikaci, jeho použití v prostředí sítí IPv6 je raritou — právě z důvodu složitosti jeho nasazení a používání.

Proto jsme se zabývali i protokoly TLS/SSL a SSH, které pracují na vyšších vrstvách ISO/OSI modelu a nevyžadují podporu na úrovni jádra operačního systému. Protokol TLS/SSL zajišťuje bezpečnost na aplikační úrovni, je dostupný ve formě knihoven umožňujících vytváření síťových aplikací, jejichž datový tok v síti bude zabezpečen šifrováním. Použití tohoto protokolu je výrazně jednodušší oproti použití IPsec. Velkou nevýhodou je chybějící povinnost implementace mechanismu obnovy šifrovacích klíčů během existujícího spojení. Samotný algoritmus se také typicky používá pro zabezpečení přístupu k webovým serverům a realizaci citlivých operací. Pro tato spojení je typické jejich krátké trvání, kde obnova klíčů či výměna šifrovacích algoritmů nehraje významnou roli.

Protokol SSH také pracuje nad transportní vrstvou, je dostupný ve formě aplikací umožňujících vytváření zabezpečených kanálů pro jiné aplikace. Tudíž

původní aplikace není potřeba modifikovat a doplňovat jejich kód o bezpečnostní mechanismy využitím jiných knihoven. Protokol SSH chrání data podobnými šifrovacími algoritmy a mechanismy jako předchozí protokoly. Má také jasně definován mechanismus obnovy klíčů během spojení, což je důležitá vlastnost pro dlouhotrvající spojení, ve kterých se přenáší velké množství dat. To je důležité i v případě LIS, kde je možné očekávat vytvoření spojení mezi sondami a MF, ve kterých se budou zabezpečeným kanálem po celou dobu fungování sond a platnosti požadavků na odposlech nepřetržitě přenášet data.

Pro zajištění ochrany dat v LIS se proto vzhledem k výše uvedenému zaměříme na využití protokolu SSH, detailnímu nastudování specifikace, navržení a implementace hardwarové architektury pro podporu přenosů protokolem SSH pro prostředí mikro-sondy.

5 Plánovaný výsledek

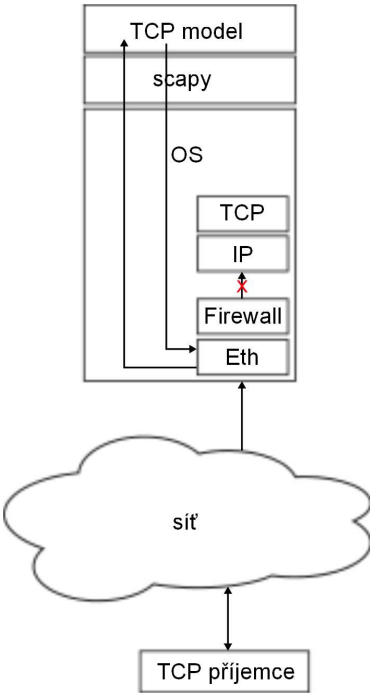
V rámci analýzy vlastností protokolů pro spolehlivý a bezpečný přenos dat se v dalším období zaměříme na využití protokolu TCP a jeho vlastností pro zajištění spolehlivých a rychlých přenosů zachycených dat v systému LIS. Pro implementaci vlastností protokolu TCP vycházíme z popisované analýzy v sekci 3. Pro zajištění bezpečnosti jsme se po analýze popisované v sekci 4 rozhodli použít protokol SSH. V této sekci bude popsán vytvářený model komunikace pomocí protokolu TCP a nástin využití kombinace protokolu TCP a SSH pro výslednou implementaci zajišťující spolehlivý a bezpečný přenos dat v systému LIS.

5.1 Model komunikace pomocí protokolu TCP

Pro potřeby vývoje hardwarové implementace protokolu TCP pro prostředí mikro-sondy jsme vytvořili model protokolu TCP v jazyce Python. Protože implementaci spolehlivého protokolu budeme muset provádět vlastními silami, je cílem modelu ověřit funkčnost vybraných mechanismů a zároveň vlastností definovaných ve specifikaci protokolu. Komunikační model je vytvořen nad knihovou Scapy [6] pro zjednodušení práce s vytvářením potřebných síťových hlaviček a odesíláním síťových rámců skrze síťové rozhraní. Schéma modelu TCP je znázorněna na obr. 5.

Model TCP je vytvořen v aplikaci běžící v uživatelském prostoru a z pohledu operačního systému se jedná o standardní aplikaci. Model se týká hlavně strany iniciátora spojení, na straně serveru běží aplikace vytvořená nad síťovým soketem využívajícím služby operačního systému pro příjem dat a zasílání potvrzení. Úlohou příjemce je přijímat data, ukládat je pro pozdější porovnání a generovat odpovědi odesílateli.

Operační systém má standardně přehled o vytvořených síťových spojeních a ví, které aplikaci přísluší používaný port. Protože je model TCP implementován nezávisle na jádře operačního systému, tak je nutné zajistit, aby operační systém nepřerušoval komunikaci programu zasíláním žádosti o reset spojení. Úpravou



Obrázek 5. Komunikační model TCP vrstvy

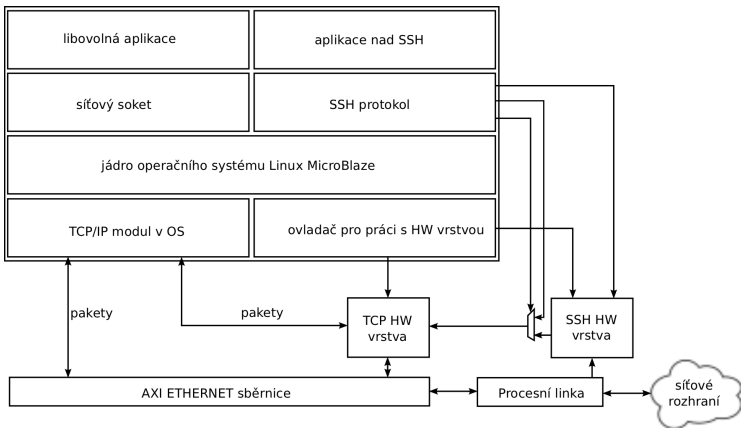
pravidel firewallu je možné docílit toho, že operační systém sice bude přijímané odpovědi vyhodnocovat jako nežádoucí, ale jeho reakce se již skrze firewall nedostanou na síťové rozhraní a k příjemci. Příjem dat a zpracování příjmu paketů je realizováno funkcemi knihovny scapy. Opět se obchází zpracování operačním systémem a pomocí zachytávacích funkcí scapy se přímo ze síťového rozhraní zachytávají síťové rámce a ty se zpracovávají v aplikaci.

Model byl vyvíjen v několika fázích, ve kterých se ověřovaly možnosti navázání spojení, otestování a ukončení spojení, dále jednoduchý přenos dat s jejich čtením ze vstupní datové fronty a také s využitím segmentů maximální velikosti. Model byl také testován na některé nestandardní situace.

5.2 Hardwarová implementace

Souběžně probíhá návrh a vývoj hardwarové implementace protokolu TCP pro prostředí mikro-sondy v jazyce VHDL.

Navrhli jsme také architekturu využívající blok protokolu TCP pro spolehlivé přenosy dat a bloků pro bezpečný přenos dat. Architektura pokrývá softwarové rozhraní nad operačním systémem Linux běžícím na procesoru MicroBlaze v mikro-sondě umožňující komunikaci s hardwarovými bloky. Výsledná architektura je znázorněna na obrázku 6.



Obrázek 6. Výsledná komunikační architektura

Výsledné schéma umožňuje využití stávajících modulů v operačním systému pro komunikace protokolem TCP a SSH. Doplnění hardwarových bloků pro urychlení jak šifrování, tak přenos dat protokolem TCP bude vyžadovat úpravy na úrovni operačního systému. Tyto úpravy zajistí možnost konfigurovatelnosti jednotlivých bloků a také volitelné vynechání těchto bloků pro realizaci síťových přenosů.

Předpokládá se částečné využití zpracování na softwarové úrovni — např. zpracování úvodní komunikace pro ustavení bezpečného kanálu protokolem SSH a také příchozích přenosů. Urychlení zpracování se očekává hlavně v odchozím směru sondy, kdežto v příchozím směru bude dat výrazně méně. Příchozí data jsou totiž tvořena jen počáteční výměnou informací pro ustavení spojení nebo konfiguraci.

6 Závěr

V předchozím textu byla přiblížena analýza protokolů pro zajištění bezpečného a spolehlivého přenosu dat v systému LIS. Analýza vychází z doporučení norem pro stavbu systémů LIS. Protože normy neuvádějí přesně požadované protokoly pro spolehlivý a bezpečný přenos dat, provedli jsme analýzu protokolů zmiňovaných v normách či jejich příbuzných variant.

První analýza se týkala vlastností existujících protokolů pre spolehlivý přenos dat. Vzhledem k nedostatkům, které analyzované protokoly obsahují, jsme se rozhodli pro další nasazení využít protokol TCP. Protokol TCP je v současnosti velmi propracovaný protokol, je dostupný v podstatě v každé implementaci dnešních operačních systémů, čímž je zajištěna interoperabilita a navíc obsahuje i velmi pokročilé mechanismy pro zajištění řízení toku dat.

Další část analýzy tvořilo nastudování vlastností protokolů zajišťujících bezpečnost přenášených dat. Z této skupiny protokolů jsme pro další implementaci vybrali protokol SSH, který obsahuje všechny dnes požadované vlastnosti pro zajištění vysoké úrovně zabezpečení a podporu existujících implementací.

V rámci práce na implementaci podpory těchto protokolů v našem LIS jsme vyvinuli model komunikace pomocí protokolu TCP, na kterém ověřujeme analyzované vlastnosti vůči dostupným implementacím. Implementace bloků protokolu SSH bude provedena v dalším období, kde je naší snahou jednotlivé bloky spolu propojit, aby tvořili navzájem spolupracující celek realizující spolehlivý a bezpečný přenos dat.

Reference

1. Enhanced Interior Gateway Routing Protocol (EIGRP). [[online]], citováno 2012-10-24.
   URL
   http://www.cisco.com/en/US/tech/tk365/tk207/tsd_technology_support_sub-protocol_home.html
2. How The Real-Time Tsunami Works. [[online]], citováno 2012-10-24.
   URL http://tsunami-udp.cvs.sourceforge.net/viewvc/tsunami-udp/docs/howTsunamiWorks.txt
3. Installing and Using Tsunami UDP. [[online]], citováno 2012-10-24.
   URL
   http://http://projects.arcs.org.au/trac/systems/wiki/DataServices/TsunamiUdp
4. The Internet Protocol Journal, Volume 12, No.4, Protocol Basics: Secure Shell Protocol. [[online]], citováno 2012-10-24.
   URL http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_12-4/124_ssh.html
5. RSA SecurID. [[online]], citováno 2012-10-24.
   URL http://www.emc.com/security/rsa-securid.htm
6. Scapy. [[online]], citováno 2012-10-24.
   URL http://www.secdev.org/projects/scapy/
7. Barret, D. J.; Silverman, R. E.; Byrnes, R. G.: *SSH, The Secure Shell: The Definitive Guide*. Sebastopol: O'Reilly Media, Inc., druhé vydání, 2005, ISBN 0-13-046019-2.
8. Baugher, M.; McGrew, D.; Naslund, M.; aj.: *The Secure Real-time Transport Protocol (SRTP)*. březen 2004.
   URL http://tools.ietf.org/html/rfc3711
9. Cusack, F.; Forssen, M.: *Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)*. leden 2006.
   URL http://tools.ietf.org/html/rfc4256
10. Dierks, T.; Rescorla, E.: *The Transport Layer Security (TLS) Protocol Version 1.2*. srpen 2008.
    URL http://www.ietf.org/rfc/rfc5246.txt

11. European Telecommunications Standards Institute: *ETSI TR 101 943: Telecommunications security; Lawful Interception (LI); Concepts of Interception in a generic Network Architecture*. červenec 2001, version 1.1.1.
12. European Telecommunications Standards Institute: *ETSI TR 102 528: Lawful Interception (LI); Interception domain Architecture for IP networks*. říjen 2006, version 1.1.1.
13. European Telecommunications Standards Institute: *ETSI TR 102 661: Lawful Interception (LI); Security Framework in Lawful Interception and Retained Data Environment*. listopad 2009, version 1.2.1.
14. European Telecommunications Standards Institute: *ETSI TR 102 232-1: Lawful Interception (LI); Handover Interface and Service-Specific Details (SSD) for IP delivery; Part 1: Handover specification for IP delivery*. srpen 2010, version 2.5.1.
15. Frankel, S.; Glenn, R.; Kelly, S.: *The AES-CBC Cipher Algoritm and Its Use with IPsec*. září 2003.
URL http://tools.ietf.org/html/rfc3602
16. Frankel, S.; Krishnan, S.: *RFC 6071 IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap*. únor 2011.
URL http://tools.ietf.org/html/rfc6071
17. Freier, A.; Karlton, P.; Kocher, P.: *The Secure Sockets Layer (SSL) Protocol Version 3.0*. srpen 2011.
URL http://tools.ietf.org/html/rfc6101
18. Haller, N.: *The S/KEY One-Time Password System*. únor 1995.
URL http://tools.ietf.org/html/rfc1995
19. Handley, M.; Jacobson, V.; Perkins, C.: *SDP: Session Description Protocol*. červenec 2006.
URL http://tools.ietf.org/html/rfc4566
20. Harkins, D.; Carrel, D.: *RFC 2409 The Internet Key Exchange (IKE)*. listopad 1998.
URL http://tools.ietf.org/html/rfc2409
21. Housley, R.: *Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)*. leden 2004.
URL http://tools.ietf.org/html/rfc3686
22. Housley, R.: *Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)*. prosinec 2005.
URL http://tools.ietf.org/html/rfc4309
23. Kaufman, C.; Hoffman, P.; Nir, Y.; aj.: *RFC 5996*. září 2010.
URL http://tools.ietf.org/html/rfc5996
24. Kaufman, C.; Perlman, R.; Speciner, M.: *Network Security: Private Communication in a Public World*. New Jersey: Prentice Hall PTR, druhé vydání, 2002, ISBN 0-13-046019-2.
25. Kent, S.; Atkinson, R.: *IP Authentication Header*. listopad 1998.
URL http://tools.ietf.org/html/rfc2402
26. Kent, S.; Atkinson, R.: *RFC 2401 Security Architecture for the Internet Protocol*. listopad 1998.
URL http://tools.ietf.org/html/rfc2401

27. Kent, S.; Atkinson, R.: *RFC 2406 IP Encapsulating Security Payload*. listopad 1998.
URL http://tools.ietf.org/html/rfc2406
28. Kent, S.; Seo, K.: *RFC 4301 Security Architecture for the Internet Protocol*. prosinec 2005.
URL http://tools.ietf.org/html/rfc4301
29. L-A.Larzon; Pink, S.; Jonsson, L.-E.; aj.: *The Lightweight User Datagram Protocol (UDP-Lite)*. červenec 2004.
URL http://tools.ietf.org/html/rfc3828
30. Lehtinen, S.; Lonvick, C.: *The Secure Shell (SSH) Protocol Assigned Numbers*. leden 2006.
URL http://tools.ietf.org/html/rfc4250
31. Maughan, D.; Schertler, M.; Schneider, M.; aj.: *RFC 2408 Internet Security Association and Key Management Protocol (ISAKMP)*. listopad 1998.
URL http://tools.ietf.org/html/rfc2408
32. Orman, H.: *RFC 2412 The OAKLEY Key Determination Protocol*. listopad 1998.
URL http://tools.ietf.org/html/rfc2412
33. Ott, J.; Wenger, S.; Sato, N.; aj.: *RFC 4585 Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)*. červenec 2006.
URL http://tools.ietf.org/html/rfc4585
34. Partridge, C.; Hinden, R.: *Version 2 of the Reliable Data Protocol*. duben 1990.
URL http://tools.ietf.org/html/rfc1151
35. Polčák, L.; Hranický, R.: Útoky na systémy pro zákonné odposlechy. Technical Report FIT-TR-2012-008, Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic, 2012.
36. Postel, J.: *Transmission Control Protocol*. leden 1980.
URL http://tools.ietf.org/html/rfc761
37. Postel, J.: *User Datagram Protocol*. srpen 1980.
URL http://tools.ietf.org/html/rfc768
38. Rescorla, E.: *Diffie-Hellman Key Agreement Method*. červen 1999.
URL http://tools.ietf.org/html/rfc2631
39. Rewaskar, S.; Kaur, J.; Smith, F.: A Performance Study of Loss Detection/Recovery in Real-world TCP Implementations. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, říjen 2007, s. 256 –265.
40. Samar, V.; Schemers, R.: *Unified Login with Pluggable Authentication Modules (PAM)*. říjen 1995.
URL http://www.kernel.org/pub/linux/libs/pam/pre/doc/rfc86.0.txt.gz
41. Schlyter, J.; Griffin, W.: *Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints*. leden 2006.
URL http://tools.ietf.org/html/rfc4255
42. Schulzrinne, H.; Casner, S.; Frederick, R.; aj.: *RTP: A Transport Protocol for Real-Time Applications*. červenec 2003.
URL http://tools.ietf.org/html/rfc3550
43. Velten, D.; Hinden, R.; Sax, J.: *Reliable Data Protocol*. červenec 1984.
URL http://tools.ietf.org/html/rfc908

44. Ylonen, T.; Lonvick, C.: *The Secure Shell (SSH) Authentication Protocol*. leden 2006.
URL http://tools.ietf.org/html/rfc4252
45. Ylonen, T.; Lonvick, C.: *The Secure Shell (SSH) Connection Protocol*. leden 2006.
URL http://tools.ietf.org/html/rfc4254
46. Ylonen, T.; Lonvick, C.: *The Secure Shell (SSH) Protocol Architecture*. leden 2006.
URL http://tools.ietf.org/html/rfc4251
47. Ylonen, T.; Lonvick, C.: *The Secure Shell (SSH) Transport Layer Protocol*. leden 2006.
URL http://tools.ietf.org/html/rfc4253
48. Zhu, L.; Jaganathan, K.; Hartman, S.: *The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2*. červenec 2005.
URL http://tools.ietf.org/html/rfc4121

A Seznam zkratek

- 3DES – *Triple Data Encryption Standard* – Symetrický šifrovací algoritmus
- AES – *Advanced Encryption Standard* – Symetrický šifrovací algoritmus
- AF – *Administration Function* – Administrační funkce
- CC – *Content of Communication* – Obsah komunikace
- CC-IIF – *Content of Communication - Internal Interception Function* – *Funkce odposlechu obsahu komunikace*
- CCTF – *Content of Communication Trigger Function* – Trigerovací funkce
- HI – *Handover Interface* – Předávací rozhraní
- HMAC – *Keyed-Hash Message Authentication Code*
- INI – *Internal Network Interface* – Interní síťové rozhraní
- IP – *Internet Protocol*
- IRI-IIF – *Intercept Related Information - Internal Interception Function* – Funkce dynamické identity
- ISP *Internet Service Provider* Poskytovatel připojení k Internetu (ISP)
- LEA *Law Enforcement Agency* – Orgány činné v trestním řízení
- LIS – *Lawful Interception System* – Systém pro sběr dat pro zákonné odposlechy
- MAC – *Message Authentication Code*
- MF – *Mediation Function* – Mediační funkce
- MSS – *Maximum Segment Size* – Maximální velikost segmentu
- MTU – *Maximal Transmission Unit* – Maximální velikost přenášeného paketu
- PDU – *Protocol Data Unit*
- PMTU – *Path Maximal Transmision Unit* – Maximální velikost přenášeného na cestě
- RTO – *Retransmission Timeout*
- RSA – *Rivest, Shamir, Adleman* – Asymetrický šifrovací algoritmus

– SHA – *Secure Hash Standard* – Rozšířená hashovací funkce
– SSH – *Secure Shell*
– SSL – *Secure Sockets Layer*
– TCP – *Transmission Control Protocol*
– TLS – *Transport Layer Security*
– UDP – *User Datagram Protocol*