VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

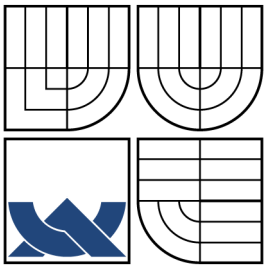
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

BEZPEČNOSTNÍ PROTOKOLY V PRAXI

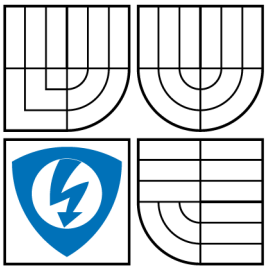
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JIŘÍ MILFAJT
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

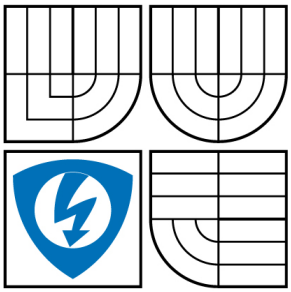
BEZPEČNOSTNÍ PROTOKOLY V PRAXI
SECURITY PROTOCOLS IN PRACTICE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ MILFAJT
AUTHOR

VEDOUCÍ PRÁCE — ING. TOMÁŠ PELKA
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Milfajt Jiří	***ID:***	83465
Ročník:	3	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

Bezpečnostní protokoly v praxi

POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta je v bakalářské práci popsat dnes používané AAA protokoly. Student se zaměří na jejich hlavní znaky, výhody, nevýhody a možné bezpečnostní slabiny. Vzájemně porovná jednotlivé implementace těchto protokolů.
V duhé fázi popíše podrobně konfiguraci volně dostupných autentizačních serverů RADIUS a Kerberos v síti simulované prostřednictvím dostupných virtualizačních nástrojů.

DOPORUČENÁ LITERATURA:

[1] DOSTÁLEK, Libor. Velký pruvodce protokoly TCP/IP : Bezpecnost. 2. aktualiz. vyd. Brno : Computer Press, 2003. 592 s. ISBN 80-7226-849-X.

[2] NAKHJIRI , Madjid, NAKHJIRI, Mahsa. AAA and Network Security for Mobile Access : Radius, Diameter, EAP, PKI and IP Mobility. 1st edition. [s.l.] : Wiley, 2005. 318 s. ISBN 978-0470011942.

Termín zadání:	11.2.2008	***Termín odevzdání:***	4.6.2008

Vedoucí práce: Ing. Tomáš Pelka

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Jiří Milfajt
Bytem: por. Vodičky 2473/2, 75002, Přerov - Přerov I-Město
Narozen/a (datum a místo): 13.6.1985, Přerov

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☐ diplomová práce
   - ☑ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Bezpečnostní protokoly v praxi
Vedoucí/školitel VŠKP: Ing. Tomáš Pelka
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

ABSTRAKT

Tato práce mapuje dnes nejvíce používané AAA protokoly. Patří mezi ně především protokoly RADIUS, TACACS+, DIAMETER a KERBEROS. První část se zaměřuje na jejich hlavní znaky. U všech je popsána základní komunikace mezi uživatelem a serverem. Dále jsou obsaženy srovnání jednotlivých protokolů sledující zejména jejich výhody, nevýhody a možné bezpečnostní slabiny. V druhé části je rozebrána konfigurace protokolů RADIUS a KERBEROS.

KLÍČOVÁ SLOVA

protokol, autentizace, autorizace, účtování, server, klient, uživatel, paket, lístek

ABSTRACT

This bachelor's thesis maps the most used AAA protocols in these days. These protocols include RADIUS, TACACS+, DIAMETER and KERBEROS. The first part considers its main signs. Then there is describes basic communication between user and server. The thesis also compares each protocol by its advantages, disadvantages and its possible security weaknesses. The second part is dedicated to configuration of RADIUS and KERBEROS protocol.

KEYWORDS

protocol, authentizacion, authorization, accounting, server, client, user, packet, ticket

MILFAJT J. *Bezpečnostní protokoly v praxi*. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2008. 56 s. Vedoucí bakalářské práce Ing. Tomáš Pelka.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Bezpečnostní protokoly v praxi“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu mé bakalářské práce Ing. Tomášovi Pelkovi za poskytnutí odborných připomínek při řešení této práce.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Internet je dnes již součástí života spousty z nás. Využíváme ho k práci, k zábavě, k nakupování, k posílání pošty a nebo třeba k online komunikaci skoro skýmkoliv na světě. Jeho možnosti využití se zdají být skoro neomezené a stěží by se hledali nějaké služby, které nedokáže poskytnout. Internet je však nezabezpečené území. Mnoho protokolů užívaných v internetu neposkytují žádné zabezpečení. Nástroje k zachycení hesel a citlivých údajů v síti jsou běžně používány zákeřnými hackery. Aplikace posílající sítí nezašifrovaná hesla jsou extrémně zranitelné. Ještě hůře jsou na tom aplikace důvěřující uživatelům v tom, že použijí pouze služby pro ně určené, aniž by byli nějakým způsobem omezováni ze strany serveru.

To je důvod, proč je nezbytné ověřit zda osoba nebo klient, se kterým komunikujete, je ve skutečnosti ten, kdo říká že je. Jakmile je jednou žadatel ověřen, stanoví se přesně jeho práva a zpřístupní se mu služby, na které má nárok. Neméně důležitá je také možnost sledování činnosti uživatelů a uchování těchto informací například pro vyúčtování platby za poskytnuté služby.

A zde nacházejí uplatnění tzv. AAA protokoly. Co vůbec tato zkratka znamená? AAA – autentizace, autorizace a účtování(*accounting*) – autentizace identifikuje uživatele, autorizace stanoví co uživatel může a účtování sleduje využívání danné služby. Mezi nejznámější AAA protokoly beze sporu patří RADIUS, vyvynutý v polovině devadesátých let firmou Livingston Enterprises. Ve svých počátcích nabízel pouze autentizaci a autorizaci, později byl rozšířen i o funkce účtovací. Za jeho nástupce je považován DIAMETER (*diameter* – z angličtiny znamená průměr, *radius* – poloměr), který, jak se dále dozvíme, vylepšuje především zabezpečení přenosu mezi jednotlivými subjekty. Důležitým hráčem na scéně je také protokol TACACS, původně vyvýjený Americkým ministerstvem obrany, časem rozšířený na XTACACS, dnes již vetšinou nahrazovaný protokolem TACACS+, jenž vyvinula firma Cisco. Poslední část práce je věnována autentizačnímu systému KERBEROS, jehož první verze byly vyvinuty v rámci projektu Athena v letech 1983-1991 v laboratořích MIT (*Massachusetts Institute of Technology*).

V jednotlivých kapitolách jsou popsány základní principy funkčnosti protokolů, výhody, nevýhody a srovnání mezi nimi.

1 AAA PROTOKOLY

1.1 Základní vlastnosti AAA

Jak již bylo řečeno v úvodu, v oblasti počítačové bezpečnosti AAA znamená authentication, authorization and accounting protocol – autentizační, autorizační a účtovací protokol.

1.1.1 Autentizace

Autentizace znamená potvrzení, že uživatel požadující služby je platným uživatelem poskytovaných síťových služeb. Autentizace je dosažena pomocí představení identity a jistého pověření nebo tajemství. Autentizace se dělí do třech základních tříd [3]:

- autentizace **znalostí** – svou identitu žadatel prokáže znalostí (např. znalostí správné kombinace uživatelského jména a hesla nebo PINu),
- autentizace **žadatelem** – žadatel dokazuje svou identitu pomocí svých vlastností, které lze prověřit (např. snímek oční zornice, otisk prstu, hlas),
- autentizace **předmětem** – identita je prokázána na základě předmětu, který žadatel vlastní (např. USB dongle, id průkaz, platební karta apod.).

1.1.2 Autorizace

Autorizace znamená udělení specifického typu služby uživateli, kterou požaduje, na základě jeho autentizace. Autorizace může být založena na omezeních, například omezení na určité hodiny v rámci dne, nebo omezení na fyzickou polohu, nebo omezení vícenásobného přihlášení jednoho uživatele. Autorizace určuje povahu služby, která je poskytnuta uživateli. Mezi typy služeb například patří: filtrování IP adres, přidělení adresy, přidělení cesty, šifrování, atd.

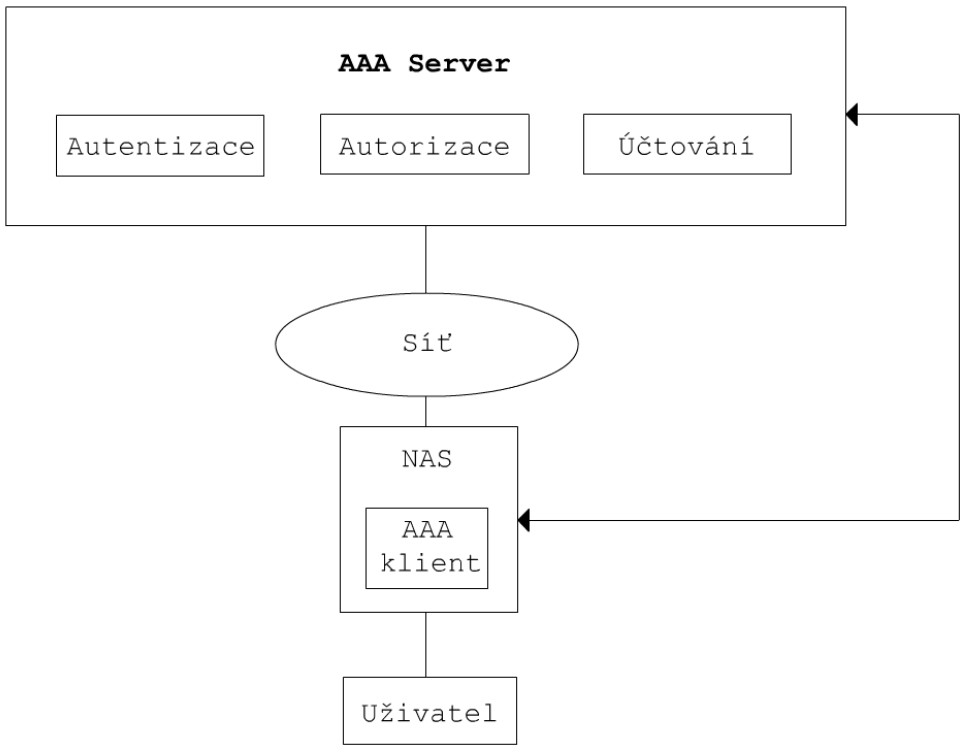
1.1.3 Účtování

Účtování znamená sledování využívání síťových služeb uživateli. Tyto informace mohou být použity pro správu, plánování, účtování, nebo další účely. Účtování v reálném čase je doručeno současně s využíváním zdrojů. Dávkové účtování ukládá informace o účtech dokud není později doručena. Běžně se sbírají informace o identitě uživatele, povaze dodaných služeb a časy počátků a konců dodaných služeb.

1.1.4 Obecný princip fungování AAA

Obrázek 1.1 ukazuje základ řešení AAA. AAA server je spojen se sítí a slouží jako hlavní místo pro uskladnění a distribuci AAA informací. Zařízení mající úlohu vstupu do sítě je typicky NAS – *Network Access Server* (fyzicky Access Point nebo router) a nebo další hostitel plnící funkci AAA klienta. Průběh AAA procesu lze shrnout do následujících kroků:

- Uživatel se připojí na vstupní zařízení (AAA klient) a požádá o přístup do sítě.
- AAA klient žádost přijme a pošle ji na AAA server.
- AAA server data zpracuje a vrátí zpět AAA klientovi kladnou či zápornou odpověď.
- AAA klient informuje uživatele, zda mu byl přístup udělen nebo zamítnut.



Obr. 1.1: AAA – základní princip

1.2 RADIUS

1.2.1 Protokol RADIUS

Protokol RADIUS (Remote Authentication Dial In User Service – uživatelská vytáčená služba pro vzdálenou autentizaci) [1] slouží pro přenos autentizačních, autorizačních, konfiguračních a evidenčních informací mezi přístupovým serverem (RADIUS klient) a společným autentizačním serverem (RADIUS server). Umožňuje centrální správu uživatelských účtů.

RADIUS server autentizuje a autorizuje vzdálené uživatele pro přístup do systému. RADIUS server také rozhoduje o zpřístupnění služby (například připojení do sítě). RADIUS klient je zodpovědný za odesílání uživatelských informací určenému RADIUS serveru a zpracování odpovědí, které RADIUS server vrátí. Oficiálně přidělené číslo portu pro RADIUS je UDP 1812.

Prostředníkem mezi uživatelem a RADIUS serverem je tedy RADIUS klient (NAS). RADIUS server zpracovává požadavek ve 2 krocích - autentizace a autorizace. Ověřením zkontroluje identitu uživatele, tedy porovná údaje ve své databázi se zaslanými údaji. Po úspěšné autentizaci dojde k autorizaci, která rozhoduje, jaké služby budou uživateli zpřístupněny.

Princip komunikace je založen na dvojici atribut-hodnota (AVP – *Attribute Value Pairs*), například jako pár „username“ a „Tomas“. Server vyzve uživatele k doplnění dvojice atributů, např. na dotaz „username“ očekává od uživatele odezvu ve formě „Tomas“.

1.2.2 Funkce RADIUSu

Pokud je klient nakonfigurován k použití RADIUS protokolu [2], každý z uživatelů musí klientovi předat své autentizační údaje.

Klient informace od uživatele obdrží jednou a provede autentizaci pomoci RADIUS protokolu. To udělá tak, že klient vytvoří `Access_Request` (požadavek o přístup) , obsahující atributy uživatelské jméno, uživatelské heslo a ID portu, přes který je uživatel připojen.

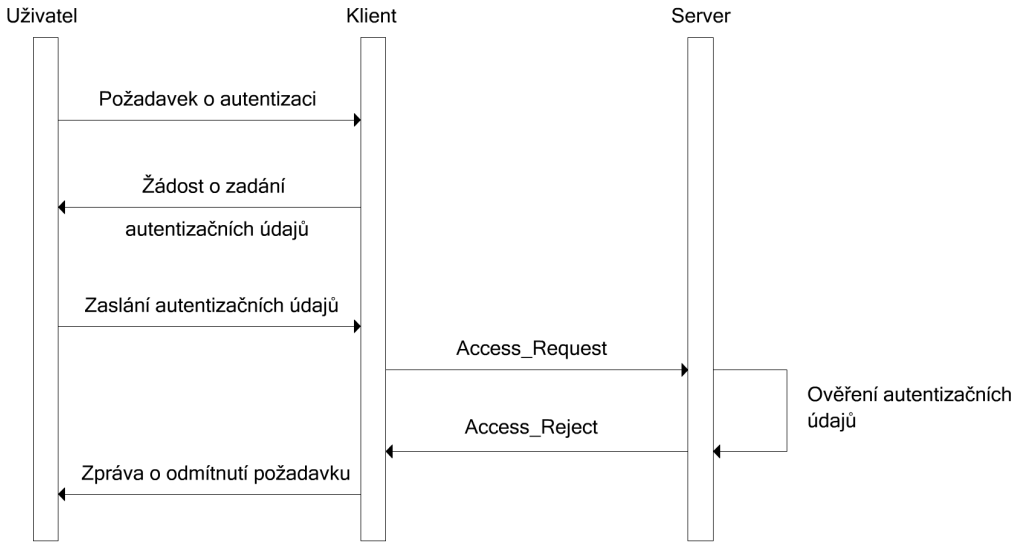
Požadavek `Access_Request` je odeslán RADIUS serveru přes síť. Jestliže se nevrátí od RADIUS serveru žádná odezva v určeném čase, požadavek je opakovaně odeslán znovu. Klient může také přeposlat požadavek alternativnímu serveru nebo serverům v případě, že primární server je vypnut nebo nedostupný. Alternativní server může být použit po určitém počtu pokusů, kdy primární server selhal.

RADIUS server příjme požadavek a ověří odesílajícího klienta. Požadavek od klienta, pro kterého RADIUS server nemá sdílené tajemství, by měl být tiše zahozen. Jestliže je totožnost klienta správná, RADIUS server se podívá do databáze

uživatelů a vyhledá jméno uživatele, jež je obsaženo v požadavku. Uživatelský záznam v databázi obsahuje seznam parametrů (například uživatelova IP-adresa nebo IP-adresa RADIUS klienta, přes který se uživatel snaží přistupovat) a které musí souhlasit s údaji pro umožnění přístupu uživateli. Pro umožnění přístupu uživateli se ověřuje heslo, které může také specifikovat RADIUS klienta nebo port přístupového serveru, přes který je uživateli umožněn přístup.

Jestliže některá z podmínek není splněna, RADIUS server odešle `Access_Reject` (zamítnutí přístupu), zprávu oznamující, že tento uživatelský požadavek je neplatný.

Průběh komunikace mezi uživatelem, klientem a serverem je znázorněn na obrázcích. Obr. 1.2 je případ, kdy nejsou splněny všechny podmínky nutné pro autentizaci a tudíž dochází k zamítnutí požadavku.

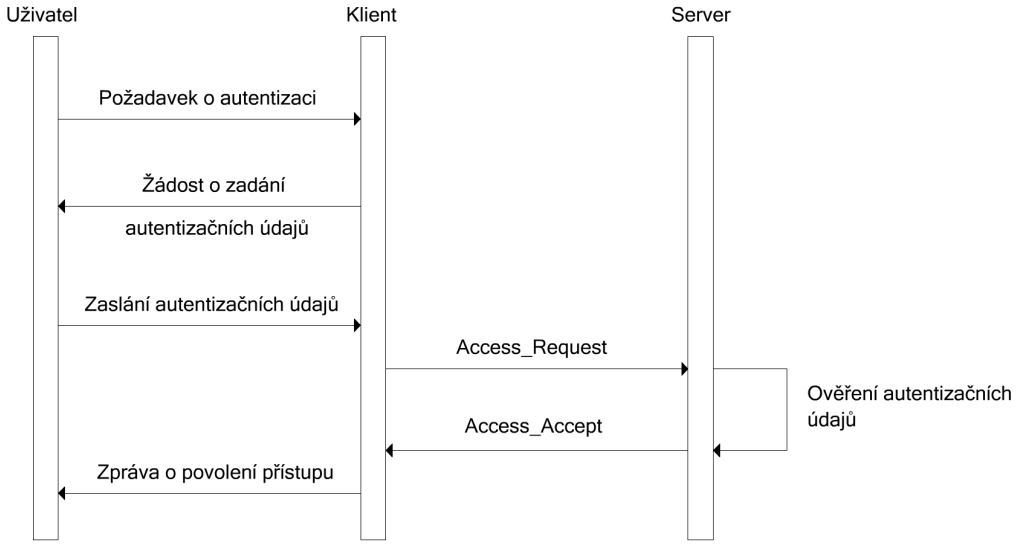


Obr. 1.2: RADIUS – zamítnutí požadavku

Pokud dojde ke splnění všech podmínek, seznam konfiguračních hodnot pro uživatele je umístěn do `Access_Accept` odpovědi. Tyto hodnoty obsahují typ služby například: IP adresu, masku sítě, login uživatele a všechny hodnoty, které je potřeba předat požadované službě. Tato zpráva tedy již obsahuje autorizační informace. Obr. 1.3 je případ, kdy byly splněny všechny nutné podmínky pro autentizaci a uživateli byl povolen přístup k poskytované službě.

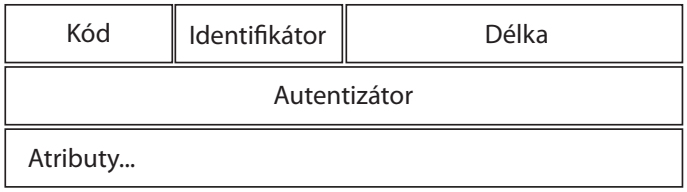
1.2.3 Formát paketu

Data mezi serverem a uživatelem jsou posílána prostřednictvím RADIUS paketů [5]. Paket je schématicky znázorněn na obr. 1.4 . Každý paket obsahuje následující informace:



Obr. 1.3: RADIUS – povolení přístupu

- **Kód**(8 bitů) – identifikuje typ RADIUS paketu. V případě kdy je hodnota neplatná, je paket zahozen. Může obsahovat hodnoty: Access-Request, Access-Accept, Access-Reject, Accounting-Request, Accounting-Response
- **Identifikátor**(8 bitů) – pomáhá správnému párování odpovídajících požadavků a odpovědí.
- **Délka**(16 bitů) – určuje velikost RADIUS paketu. V případě kdy je paket menší než je určeno v poli délka může být paket zahozen. Minimální délka je 20B, maximální délka je 4096B.
- **Autentizátor**(128 bitů) – jeho hodnota je použita při autentizaci odpovědi z RADIUS serveru a dále je použita při šifrování posílaného hesla.
- **Atributy**(proměnná délka) – nesou specifické autentizační, autorizační, informační a konfigurační detaily pro požadavky a odpovědi. Konec seznamu atributů je určen délkou RADIUS paketu.



Obr. 1.4: RADIUS – formát paketu

1.2.4 Spolehlivost přenosu a bezpečnost

V době specifikace RADIUSu byl transport přes UDP považován za více vhodný než TCP, kvůli faktu, že navázání relace přes TCP je časově náročné. Zajímavostí však je, že tvůrci RADIUSU vydali poznámku na přední stranu jeho specifikace, že trpí výkonovou degradací a ztrátou paketů v nepříznivých síťových podmínkách a ochrana proti zahlcení by měla být lépe specifikovaná v jeho nástupci [1].

Transakce mezi RADIUS klientem a RADIUS serverem je autentizována pomocí sdíleného tajemství, které není nikdy posíláno přes síť. Navíc všechna uživatelská jména jsou přes sít zasílána šifrované (šifrováné pomocí algoritmu MD5[1]), a tím je eliminována možnost vysledování nechráněného hesla na síti.

Použití sdíleného tajemství jako základu pro poskytování bezpečnostních funkcí však znamená značnou zranitelnost. Následující seznam uvádí některé bezpečnostní nedostatky [1]:

- *Statická ručně konfigurovaná sdílená tajemství*: Sdílená tajemství jsou obvykle ručně konfigurována v NAS. Kvůli velkému množství NAS serverů připojených do mnoha sítí, vznikly případy, kde technik nakonfiguroval více NAS se stejným sdíleným tajemstvím, za účelem snížení administračních nákladu. Tajemství mají zpravidla dlouhou životnost a specifikace RADIUSu nedefinují jejich obnovení.

- *Řetězení proxy serverů*: V případech kdy dojde k použití RADIUS proxy serverů mezi NAS a RADIUS serverem, NAS sdílí tajemství pouze s prvním AAA proxy, ale s koncovým RADIUS serverem již ne. To znamená, že důvěra mezi NAS a RADIUS serverem je založena na řetězu proxy serverů. Pokud je některý z proxy nabourán, může dojít k bezpečnostním problémům.

- *Ochrana přenosu*: K ochraně při přenosu využívá RADIUS skrývání atributů, což zajišťuje výbornou ochranu aplikační vrstvě. Neposkytuje však žádnou ochranu pro RADIUS zprávy nebo pro vrstvy (UDP, IP), po kterých jsou zprávy posílány. Může tedy např. dojít k tomu, že IP adresa lehce podlehne spoofingu[2].

[1] *Message–Digest algorithm 5* – rozšířená hašovací funkce s kontrolním součtem (hash) o velikosti 128 bitů

[2] *Spoofing* – způsob průniku na zakázanou stránku, principem je podvržení referenční stránky, ze které jakoby přicházíte. Využívá se toho, že mnoho stránek, v rámci jednoho hesla, umožňuje přístup na další stránky.

1.2.5 Protokol RADIUS Accounting

Tento protokol je rozšířením standardního RADIUS protokolu [2]. Jeho rozšíření spočívá v možnosti zasílání účtovacích informací z přístupového serveru RADIUS accounting serveru. Tato technologie je především určena pro společnosti, které poskytují internet koncovým uživatelům, protože RADIUS accounting poskytuje informace o využití poskytovaných služeb. Z těchto informací se pak snadno dají například vystavit faktury jednotlivým uživatelům.

K používání RADIUS Accounting protokolu musí být nakonfigurován Klient. Na začátku poskytování služby vygeneruje uživateli klient `Accounting_Start` paket, popisující typ poskytované služby a uživatele, kterému je služba poskytovaná. Tento paket odešle RADIUS Accounting serveru, který jako odpověď pošle potvrzení o přijetí paketu.

Při ukončení poskytovaní dané služby klient vygeneruje `Accounting_Stop` paket, popisující typ poskytované služby a volitelný typ statistik, jako doba trvání služby, přijaté a odeslané pakety nebo přijaté a odeslané bajty. Tento paket je odeslán RADIUS Accounting serveru, ten po přijetí paketu odešle zpět potvrzení o přijetí.

1.2.6 Použití RADIUSu

I když nebyl RADIUS původně vytvořen pro autentizační metody v bezdrátových sítích, je v dnešní době běžně používán jako autentizační protokol v bezpečnostním standardu 802.1x, který je znám především z oblasti Wi–Fi. Jedná se však o obecný bezpečnostní rámec pro jakoukoli LAN.

802.1x je založený na protokolu EAP (*Extensible Authentication Protocol*), jenž zajišťuje pouze transportní mechanismus pro ověřování. V procesu autentizace podle 802.1x hrají úlohu tři komponenty [1]:

- **žadatel** – uživatel nebo klient, který chce být ověřen;
- **autentizátor** – zařízení mezi žadatelem a autentizačním serverem; buď Access point nebo přepínač;
- **autentizační server** – nejčastěji se jedná o server RADIUS.

Protokol 802.1x specifikuje přístup k portu zařízení (AP, přepínač). EAP zajišťuje přenos autentizačních zpráv, přičemž vlastní ověření probíhá zvoleným typem autentizačního protokolu (např. EAP-MD5, LEAP, EAP-TLS). Mechanismus autentizace funguje tak, že autentizátor přeposílá žádost RADIUS protokolem autentizačnímu serveru, který poté oznámí přepínači (nebo AP), zda má žadateli povolit či odmítnout přístup.

1.3 TACACS

TACACS (*Terminal Access Controller Access–Control System*)umožňuje klientovi přijmout uživatelské jméno a heslo a poslat požadavek na TACACS autentikační server, někdy zvaný TACACS démon nebo jen TACACSD. Tento server rozhodne zda přijmout nebo zamítnout pořadavek a a pošle zpět odpověď. Takto je rozhodovací proces otevřený a algoritmy a informace k němu použité jsou zcela na tom, kdo provozuje TACACS démona. K přenosu informací používá UDP protokol.

TACACS lze používat ve spolupráci se systémem KERBEROS. Uživatel nejprve od serveru systému Kerberos získá podklady pro autentizaci, aniž by se heslo přenášelo v síti, a na jejich základě se pak autentizuje přístupovému serveru, kdykoli potřebuje vzdálený přístup.

Novější verze TACACS byly nazývány XTACACS nebo extended (rozšířený) TACACS. Obě verze již byly většinou nahrazeny novější implementací TACACS+ od firmy Cisco.

1.4 TACACS+

Rozšířený systém TACACS+ [6] již podporuje všechny tři složky architektury AAA prostřednictvím autentizace při přihlášení se do systému (dialog mezi uživatelem a systémem pro ověření jména a hesla), autorizace (např. automatickým navázáním spojení se serverem a protokolem, který je uživateli dovoleno využívat) a shromáždění údajů o využívání systému uživatelem (účtování).

TACACS+ užívá, oproti TACACS, pro přenos protokol TCP, port 49.

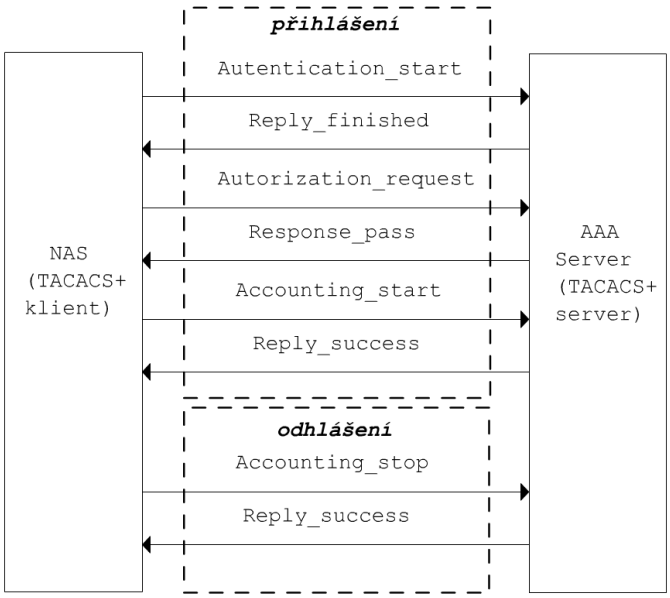
1.4.1 Průběh přihlášení a odhlášení pomocí TACACS+

Na Obr. 1.5 je znázorněn tok zpráv když se uživatel připojuje do sítě pomocí NAS a poté se odpojí [6].

1. Nejprve NAS získá uživatelské jméno a heslo od vzdáleného uživatele a pošle ho jako `Authentication_start` na TACACS+ server (autentizační fáze).

2. Pokud je kombinace jména a hesla v pořádku a server nepotřebuje žádné další informace, odpoví `Reply_finnished`

3. NAS si vyžádá některé autorizační informace od uživatele a odešle je jako `Authorization_request` (autorizační fáze).



Obr. 1.5: TACACS+ – sled zpráv

4. Odpověď serveru `Response_pass` obsahuje požadované autorizační informace (např. dobu po jaké bude uživatel odpojen, maximální objem dat které může přenést, apod.).

5. Nyní NAS odešle `Accounting_start` signalizující, že je uživatel přihlášen do sítě (fáze účtování).

6. TACACS+ server odpoví `Reply_success` s informací o úspěšném uložení účtovací zprávy.

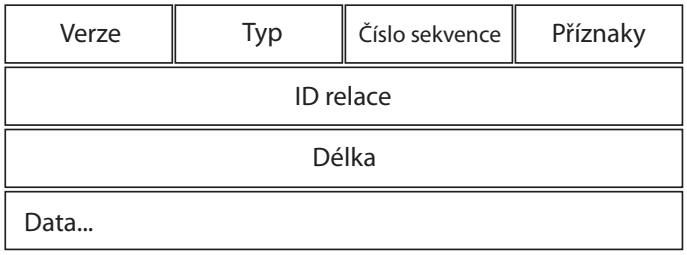
7. Ve chvíli kdy se uživatel odpojí, odešle NAS `Accounting_stop`, jenž obsahuje např. čas přihlášení a odpojení, údaje o přenesených datech, důvod proč se uživatel odpojil atd.

8. Server opět odpoví pomocí `Reply_success`, která indikuje uložení účtovací zprávy.

1.4.2 Formát paketu

Obecné schéma TACACS+ paketu [6] je znázorněno na obr. 1.6 . Každý paket obsahuje následující informace:

- **Verze**(8 bitů) – pole obsahující hlavní číslo verze a číslo vedlejší, z důvodu kompatibility mezi jednotlivými verzemi TACACS+ protokolu.
- **Typ**(8 bitů) – určuje o jaký typ zprávy se jedná (autentizační, autorizační nebo účtovací).
- **Číslo sekvence**(8 bitů) – udává číslo paketu v dané sekvenci. Může nabývat hodnot 1 až 255.
- **Příznaky**(8 bitů) – indikují zda jsou data po poli „Délka“ šifrována nebo ne.
- **ID relace**(32 bitů) – náhodně generované číslo klientem pro danou relaci.
- **Délka**(32 bitů) – celková délka TACACS+ paketu.
- **Data...**(proměnná délka) – informace v tomto poli se liší v závislosti na typu zprávy.



Obr. 1.6: TACACS+ – formát paketu

1.4.3 TACACS+ vs. RADIUS

Tabulka 1.1 popisuje hlavní rozdíly mezi systémy RADIUS a TACACS+.

Zásadním rozdílem je použití transportního protokolu. UDP na rozdíl od TCP nezaručuje, zda se přenášený paket neztratí, nezmění se pořadí paketů, nebo zda se některý paket nedoručí vícekrát. TCP poskytuje okamžité oznámení o tom, že je server mimo provoz, pomalý a nebo že třeba neexistuje.

Zašifrováním celého těla paketu zajišťuje TACACS+ bezpochyby více zabezpečení během přenosu.

Tab. 1.1: Srovnání RADIUS a TACACS+

RADIUS	**TACACS+**
Využívá k přenosu UDP	Využívá k přenosu TCP
Šifruje pouze heslo v `access_request`	Šifruje celé tělo paketu
Kombinuje autentizaci s autorizací	Odděluje autentizaci, autorizaci a účtování

Dalším nedostatkem RADIUSu je nemožnost jednoznačného rozlišení autentizace a autorizace, protože zpráva `access_accept` odesílaná RADIUS serverem do NAS obsahuje autorizační informace. TACACS+ oproti tomu jednoznačně rozlišuje jendotlivé procesy, což např. umožňuje použití systému Kerberos k autentizaci a autorizace je poté zprostředkována TACACS+ serverem [4].

1.5 DIAMETER

AAA protokoly jako TACACS a RADIUS byl zpočátku určeny převážně pro vytáčený PPP (*Point to Point protocol*) [1]. V průběhu času, s růstem Internetu a zavedení nových přístupových technologií, jako např. Wi–Fi, DSL a Ethernetu, došlo k růstu schopností routerů a síťvých přístupových serverů (NAS) a tím i rozšíření požadavků na AAA protokoly.

Lidé pracující v oblasti vývoje AAA protokolů specifikovali, na základě nedostatků RADIUSu, požadavky na AAA protokoly. Na základě těchto nových nároků byl navrhnut nový systém jménem Diameter.

Diameter je založen na základním protokolu diameteru (*Diameter Base Protocol*) a na aplikacích[3] diametru, které mohou rozšířit základní protokol přidáním nových příkazů nebo atributů.

Stejně jako u RADIUSu je komunikace založena na využívání AVP.

1.5.1 Struktura Diametru

Jak již bylo řečeno, Diameter je založen na základním protokolu diametru a sadě aplikací. Tento design dovoluje protokolu, aby se rozšířil do nových přístupových technologií.

Základní protokol diametru definuje hlavní mechanismy spolehlivého přenosu, formát zpráv a chybová oznámení.

[3] Aplikací zde není myšlen program, nýbrž protokol založený na Diameteru

K hlavním aplikacím Diametru patří CMS zabezpečení (*Cryptographic Message Syntax*), NASREQ (*Network Access Server REQuirements*) a Mobile IP. Každá aplikace se spoléhá na služby základního protokolu [1].

Obrázek 1.7 schematicky ukazuje architekturu Diametru. Základem je základní protokol Diametru, jenž je úzce spojen s CMS zabezpečením k tomu, aby poskytoval zabezpečení všem aplikacím. Ostatní aplikace, ačkoliv mají různé funkce, musí podporavat základní protokol.

NASREQ

Aplikace NASREQ poskytuje služby pro uživatele využívající vytáčený PPP a je zamýšlena jako nahrazení protokolu RADIUS.

NASREQ užívá, tak často jak to jen lze, existující atributy RADIUSu k přenosu dat. Důvodem je co nejjednoduší přechod z RADIUSu na Diameter. Navrhuje také základní pokyny pro případy použití na serveru vystupujícím jako RADIUS–Diametr protokol gateway, to znamená, že server přijíme RADIUS zprávu, jenž má být přeložena a přenesena jako zpráva Diametru a naopak [8].

Mobile IP

Aplikace Mobile IP umožňuje uživatelům využití komunikačního protokolu Mobile IP, který umožňuje uživatelům ponechat si stejnou IP adresu, zatím co cestují přes různé sítě a uchovat si tak jejich otevřená sezení [8].

CMS zabezpečení

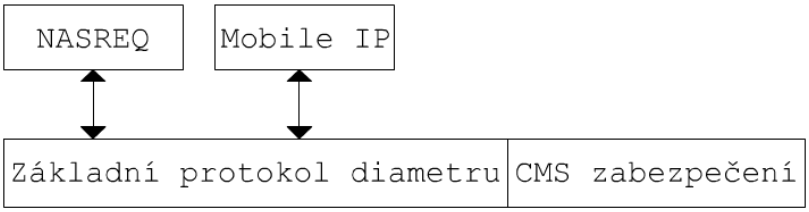
Zabezpečení protokolu Diameter je poskytováno protokolem IPsec[4] nebo TLS[5]. Tyto dvě metody se využívají pouze pro zabezpečení typu hop–by–hop, tj. mezi jednotlivými subjekty v přenosu (např. mezi uživatelem a proxy serverem, nebo mezi dvěma proxy apod.) [8].

CMS poskytuje zabezpečení typu end–to–end. Toto zabezpečení garantuje, že veškeré změny v zprávách během přenosu jsou zaznamenány.

CMS využívá k zabezpečení dvou hlavních technik. Jedná se o digitální podpisy a šifrování jednotlivých AVP.

[4] *IP security* – balík protokolů zabezpečující IP komunikaci autentizací a šifrováním každého IP paketu během přenosu

[5] *Transport Layer Security* – pomocí kryptografie autentizuje TLS koncový server – uživatel si je jist s kým komunikuje

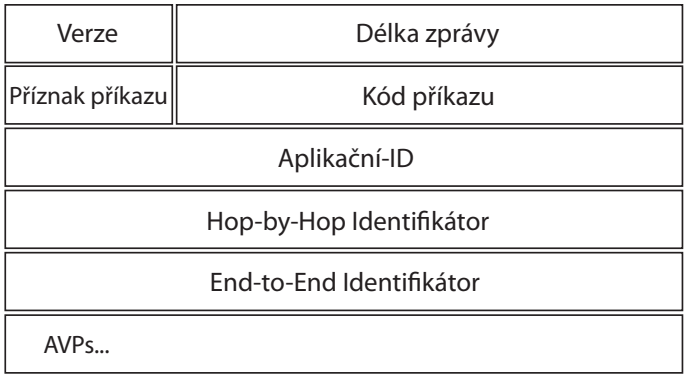


Obr. 1.7: DIAMETER – základní struktura

1.5.2 Formát paketu

Formát základního Diameter-paketu [7] je schématicky znázorněn na obrázku 1.8. Každý paket obsahuje následující informace:

- **Verze**(8 bitů) – toto pole indikuje verzi Diametru.
- **Délka zprávy**(24 bitů) – určuje celkovou délku Diameter paketu.
- **Příznak příkazu**(8 bitů) – určuje typ zprávy. Nastavením bitů lze určit zda je zpráva požadavkem či odpovědí, jestli může být přesměrována nebo pouze lokálně zpracována nebo zda je zpráva chybová.



Obr. 1.8: Diameter – formát paketu

- **Kód příkazu**(24 bitů) – obsahuje kód příkazu, pomocí kterého lze jednotlivé příkazy identifikovat. Tyto kódy jsou definovány v Základním protokolu diametru. Hodnoty 0–255 jsou rezervovány pro zpětnou kompatibilitu s kódy protokolu RADIUS.
- **Aplikační–ID**(32 bitů) – identifikuje, které aplikaci zpráva přísluší.
- **Hop–by–Hop Identifikátor**(32 bitů) – číselný identifikátor zabezpečení typu Hop–by–Hop, který pomáhá při sjednocování požadavků a odpovědí. Je nahrazen při každém skoku ze serveru na server.

- **End–to–End Identifikátor**(32 bitů) – číselný identifikátor zabezpečení typu End–to–End. Spolu s ověřením odesílatelovi identity, slouží také k detekci duplicitních požadavků. Je během celého přenosu zprávy neměnný.
- **AVPs...**(proměnná délka) – stejně jako u protokolu RADIUS nesou specifické autentizační, autorizační, informační a konfigurační detaily pro požadavky a odpovědi.

1.5.3 Diameter vs. RADIUS

Diameter je tedy navržen jako nástupce RADIUSu, který má řešit jeho dnes známé problémy. Ty nejdůležitější nedostatky RADIUSu, které řeší Diameter ukazuje následující seznam [1]:

- **Transportní protokol:** Jak bylo řečeno, RADIUS využívá UDP protokol a nedefinuje chování zpětného přenosu, následkem toho kolísá spolehlivost mezi implementacemi. Toto je hlavní problém účtování, kde se ztráta paketu může přímo projevit do ztráty příjmu. K tomu, aby DIAMETER poskytoval řádně definované chování přenosu, používá spolehlivé mechanizmy transportu (TCP, SCTP).
- **SC vs. P-to-P:** RADIUS je protokol typu server–client, kde žádosti vydává vždy klient, zatímco odpovědi jsou tvořeny serverem. Na rozdíl od toho je Diameter peer–to–peer protokol(volně přeloženo „rovný s rovným“), což znamená, že jak klient, tak server mohou tvořit buť žádosti nebo odpovědi.
- **Neúčinný algoritmus opakovaného přenosu:** RADIUS má rezervován pouze jeden byte jako pro identifikaci opětovného přenosu, což velmi limituje počet žádostí čekajících na vyřízení (max. 255). Za tímto účelem má Diametr přichystány 4 byty (max. 2^{32}).
- **Fail–over**[6] **podpora:** RADIUS neobsahuje žádný způsob upozornění na to, zda server běží či nikoliv. Diameter podporuje Keep–alive zprávy, které umožňují zjistit přerušení spojení nebo selhání serveru.
- **Zabezpečení hop–by–hop:** RADIUS podporuje pouze zabezpečení typu hop–by–hop, což dovoluje, v případě použití proxy serverů, modifikaci zprávy při každém skoku ze serveru na server aniž by si toho klient či server všimnuli. Naproti tomu Diameter podporuje ještě navíc zabezpečení end–to–end, které garantuje, že informace nebudou změněny bez povšimnutí.

[6] *Fail–over* – schopnost automatického přepnutí na záložní server v případě selhání serveru původního

1.6 Kerberos

1.6.1 Protokol Kerberos

Kerberos je systém pro autentizaci a autorizaci uživatelů v rámci sítě. Koncepce vychází z předpokladu, že prostředí sítě není důvěryhodné a vždy je zde možnost zachycení přenášených informací cizí osobou, předstírání cizí identity, neoprávněné využití služby nebo odposlouchávání přenosu.

Název byl převzat z antické mytologie, ve které trojhlavý pes Kerberos hlídal vchod do podsvětí.

Hlavní částí systému Kerberos je KDC (Key Distribution Center), které je tzv. důvěryhodnou třetí stranou umožňující bezpečnou autentizaci a autorizaci přístupu ke zdrojům a která bezpečně uchovává data o uživatelích a službách v síti. KDC se skládá ze dvou služeb:

- AS – Autentizační server, je odpovědný za autentizaci a autorizaci uživatelů a služeb
- TGS – Ticket Granting Server, je odpovědný za přidělování oprávnění - lístků - k použití zdroje (služby)

Kerberos je tedy založen na použití tzv. lístků vydávaných centrálním autentizačním serverem, který spravuje databázi všech uživatelů.

Vedle podpory autentizace poskytuje mechanismus kerberovských lístků také podporu tzv. principu SSO – Single Sign-On, který umožňuje uživatelům pohodlné použití prostředků aniž by byli zbytežně zatíženi složitými bezpečnostními procedurami. Uživatel se totiž autentizuje vůči serveru Kerbera pouze jednou a získá tak základní lístek TGT (Ticket Granting Ticket), který se dále použije k získání dalších lístků pro přístup k službám v systému, které vyžadují autentizaci. Při vzdáleném přihlašování na jiné stroje, se přenese také TGT lístek a uživatel tak má stále možnost, jak se autentizovat. Veškeré operace, kromě prvotní autentizace však již probíhají transparentně bez zásahu uživatele, který tak není zdržován ve své činnosti [2].

1.6.2 Průběh autentizace

Nyní si trochu detajněji přiblížíme průběh autentizace. Je důležité si uvědomit, že server na kterém běží požadovaná služba nikdy nekomunikuje přímo s KDC, veškerý datový pohyb prochází přes uživatele, který chce požadovanou službu zpřístupnit.

Sled jednotlivých zpráv [9] (pro lepší představuje je celý proces zobrazen na obr. 1.9):

1. `AS_REQ` – požadavek uživatele o autentizaci, který je směrován na Autentizační server.

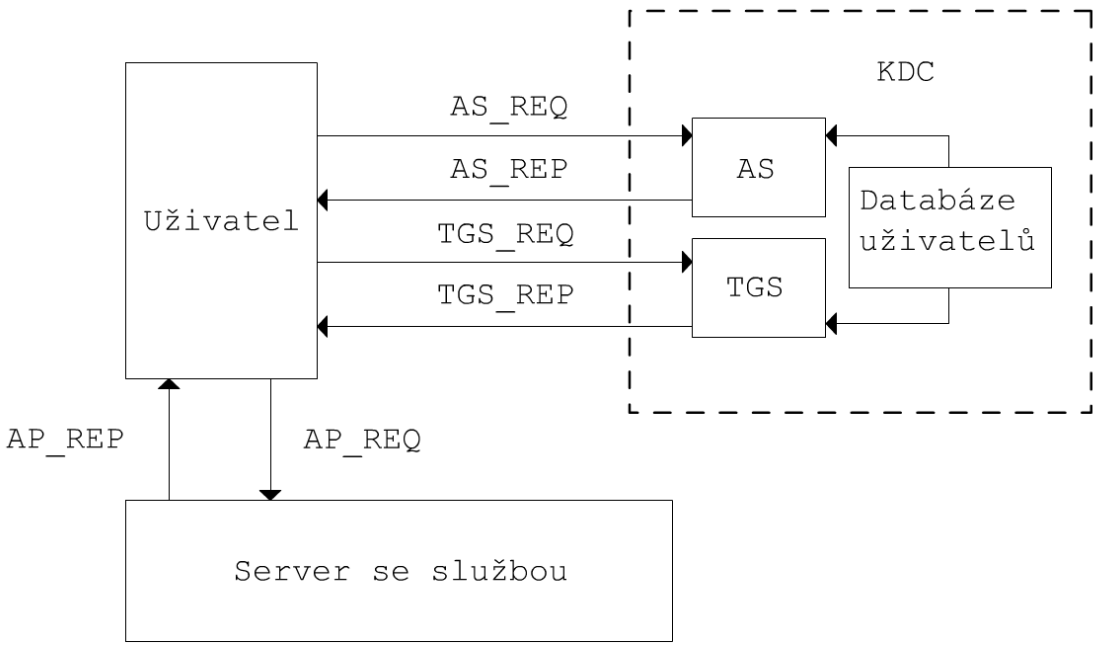
2. `AS_REP` – odpověď AS na předchozí žádost. V podstatě obsahuje TGT (zašifrovaný pomocí TGS tajného klíče) a `session_key` (zašifrovaný pomocí tajného klíče od uživatele).

3. `TGS_REQ` – žádost od uživatele určená pro TGS o lístek ke zpřístupnění služby. Tento balíček obsahuje TGT z předchozí zprávy a autentifikátor generovaný uživatelem a zašifrovaný pomocí `session_key`.

4. `TGS_REP` – odpověď TGS na předešlou žádost, která obsahuje požadovaný lístek ke službě (šifrován tajným klíčem služby) a `session_key` služby, který generuje TGS, šifrovaný předchozím `session_key`, který vygeneroval AS.

5. `AP_REQ` – požadavek uživatele poslaný na server se službou o přístup ke službě. Součástí žádosti jsou lístek ke službě obdržený z TGS v předchozí zprávě a autentifikátor opět generovaný uživatelem, tentokrát však šifrovaný pomocí `session_key` služby (generovaný v TGS).

6. `AP_REP` – odpověď serveru se službou uživateli s prokázáním že se opravdu jedná o server který klient očekával. Tato zpráva není vždy požadována, pouze v případě kdy je nezbytná oboustranná autentizace.



Obr. 1.9: KERBEROS – autentizace

1.6.3 Kerberos v5

Dnes se nejvíce používá Kerberos ve verzi 5. První tři verze Kerbera nebyly veřejně vůbec použity, naproti tomu verze 4 se poměrně dost rozšířila. Verze 4 však obsahovala některá bezpečnostní omezení, na jejichž základě byla uvedena verze 5.

Nejvetším problémem Verze 4 bylo, že nezajišťovala ochranu proti útoku opakováním (*replay attack*). Princip útoku spočívá v odposlouchávání komunikace mezi dvěma autentizujícími se stranami a následném použití odchycených dat k autentizaci útočníka. V případě Kerbera hacker zkopíruje lístek, prolomí šifrování a pokusí se vydávat za klienta s použitím jeho lístku.

Tento nedostatek je řešen posláním dodatečné informace ověřující původ zprávy. Tato informace (zvaná **autentizátor**) je šifrována tajným klíčem relace a uživatel navíc přidá časovou známku (**timestamp**). Časovou známkou je prokázáno, že zprávu obsahující lístek vytvořil uživatel nedávno a nejedná se tedy o útok opakováním. Zašifrování autentizátoru lístkem i klíčem relace potvrzuje, že zprávu generovala strana, která vlastní klíč relace. Klíč relace zaručuje totožnost uživatele, protože nikdo kromě žadatele a serveru klíč relace nezná [9].

1.6.4 Bezpečnostní slabiny Kerbera v5

Kerberos sám o sobě žádným způsobem neřeší útoky typu DoS[7] a je také náchylný na offline útoky hrubou silou (tj. i slovníkové útoky) vedoucí k získání hesla. Nejsouli použity žádné preautentizační metody, může útočník na požádání od AS získat data zašifrovaná uživatelovým heslem. Je-li heslo „slabé“ (je krátké, je to běžně používané slovo atd.) lze jej snadno hrubou silou získat (a počáteční autentizace pomocí hesla se tak stává nejslabším místem celého systému). Protože tímto způsobem útočník mohl kompromitovat spoustu slabých hesel, bylo zavedeno používání preautentizačních dat, jejichž pomocí si AS před zasláním odpovědi ověří s kým komunikuje. Výše popsanému naivnímu útoku je tak sice zamezeno, ale protože první zpráva (tj. samotná preautentizační data) opět obsahuje data zašifrovaná uživatelovým heslem, může poněkud silnější útočník odposlouchávající síťový provoz opět provést útok na heslo hrubou silou. Je tedy doporučeno volit dostatečně silná hesla, nebo raději rovnou přístupové fráze (což ovšem obecně platí všude) [9].

[7] *Denial of Service* (odmítnutí služby) – technika útoku na internetové služby nebo stránky, při níž dochází k přehlcení požadavky a pádu nebo minimálně nefunkčnosti a nedostupnosti pro ostatní uživatele.

2 KONFIGURACE AUTENTIZAČNÍCH SERVERŮ RADIUS A KERBEROS

Tato část práce je zaměřena na praktickou ukázku konfigurace dvou autentizačních serverů. Aby bylo možné, kromě samotné konfigurace serverů, ověřit také funkční autentizaci, je zapotřebí vytvořit virtuální síť, ve které se bude nacházet autentizační server a uživatel, který má být autentizován. V tomto ohledu je výborným pomocníkem virtualizační nástroj VMware server 1.05, díky kterému lze na jednom počítači vytvořit více virtuálních strojů propojené simulovanou sítí.

2.1 RADIUS

Pomocí VMware serveru bylo potřeba vytvořit jeden virtuální stroj, sloužící jako RADIUS server. Ten pracoval na operačním systému KUBUNTU. Druhé PC, ze kterého probíhalo testování funkčnosti serveru, pracovalo na Windows Vista. Testování bylo realizováno speciální utilitou vytvořenou přímo k tomuto účelu. Jedná se o NTRadPing 1.5 od společnosti MasterSoft, Inc. Jako RADIUS server byl použit FreeRADIUS.

2.1.1 Nastavení FreeRADIUSu

Nejdříve bylo nutné doinstalovat do KUBUNTU potřebný balíček – *freeradius*.

Pro konfiguraci serveru jsou důležité tyto soubory

- `radiusd.conf` – obsahuje základní konfiguraci programu,
- `users` – obsahuje přístupová hesla pro uživatele, lze využít i pro autentizaci založené na MAC adresách[1],
- `clients.conf` – seznam klientů oprávněných klást dotazy.

Ke správnému fungování serveru je tedy potřeba několik úprav techto souborů:

1. Úprava souboru `radiusd.conf`:

   Tento soubor je základním konfiguračním souborem serveru FreeRADIUS. Je velmi rozsáhlý, avšak většinu informací zabírají komentáře a navíc lze u drtivé většiny parametrů ponechat výchozí hodnoty. Důležité je nastavit IP adresu stroje na kterém běží FreeRADIUS.

   ```
   bind_address = 192.168.126.133
   ```

[1] *MAC adresa* – jedinečný identifikátor přiřazený síťové kartě bezprostředně po její výrobě.

V tomto souboru lze také nastavit výchozí port pro naslouchání, v praxi se však lze setkat jak s portem 1812 tak s portem 1645. Proto je lepší port nastavit až při spuštění serveru (viz níže).

`radiusd.conf` obsahuje např. také nastavení pro autentizační protokoly PAP a CHAP či nastavení LDAP protokolu, který může obsahovat databázi uživatelů jenž se mohou autentizovat do systému. Lze také např. definovat připojení k MySQL databázi. Jak bylo řečeno, celý soubor obsahuje velké množství komentářů, které jsou velmi užitečné při konfiguraci jednotlivých funkcí a pro názornost většinou obsahují i ukázkové příklady.

2. Úprava souboru `users`:

   Zde jsou uloženy informace o jednotlivých uživatelích.

   ```
   Jirka    Auth-Type := local, User-Password== "heslo"
   ```

   `Jirka` – reprezentuje jméno uživatele.

   `Auth-Type` – určuje způsob autentizace uživatele, konkrétně typ `local` říká, že pro uživatele `Jirka` má být pro autentizaci použito heslo zadané v parametru `User-Password`.

   V případě použití autentizace založené na MAC adrese by záznam vypadal např. následovně:

   ```
   01024304950c  Auth-Type = Local, User-Password == "01024304950c"
   ```

   MAC adresa zde vystupuje 2x, jednou jako jméno uživatele a podruhé jako heslo. Zapisuje se bez oddělovačů.

   Uživatelům lze nastavit také více parametrů, např. `Framed-IP-Address` což je jeho IP adresa nebo `Client-IP-Address` což je adresa klienta, přes kterého se uživatel snaží připojit, apod.

3. Úprava souboru `clients.conf`:

   V tomto souboru jsou popány pokyny pro konfiguraci klienta (NAS).

   ```
   client 192.168.126.101 {
   secret = radius123
   shortname = TestovaciNAS
   }
   ```

Je potřeba tedy zadat IP adresu NAS, `secret` – sdílené tajemství mezi klientem (NAS) a serverem a `shortname` což je název NAS.

V praxi je také důležitým konfiguračním souborem `eap.conf` především při využití FreeRADIUSu pro autentizaci v bezdrátových sítích. Zde se konfigurují konkrétní ověřovací protokoly které lze pro přenos informací použít. Jedná se především o EAP–MD5, EAP–TLS, PEAP.

Timto je nutná konfigurace RADIUS serveru u konce. Pomocí příkazu `radiusd` se server spustí. Přidán je ještě parametr `-p 1645`. Ten určuje port, na kterém server naslouchá. I když podle oficiální specifikace [5] RADIUSu je přidělený port číslo 1812, bývá převážně z historických důvodů požíván port 1645.

```
root@server ~: radiusd -p 1645
radiusd: Starting - reading configuration files ...
root@server ~:
```

2.1.2 Test funkčnosti FreeRADIUS serveru

Nejjednoduší ověření správné funkčnosti serveru je pomocí utility NTRadPing 1.5. Tento program běží na operačních systémech Windows a je přímo určen pro testování RADIUS serverů. Obrázek 2.1 ukazuje okno programu.

Ke spuštění testu je potřeba zadat požadované údaje:

1. Nejprve IP adresu stroje, na kterém FreeRADIUS běží a číslo portu pro naslouchání. Konkrétně 192.168.126.133 a port 1645.
2. Do pole *RADIUS secret key* patří sdílené tajemství nastavené v `clients.conf` – radius123.
3. *User–Name* a *Password* slouží pro identifikaci uživatele, jež je nastaven v `users`. User–Name: Jirka, Password: heslo.
4. Menu *Request type:* určuje typ požadavku. Zde zůstane *Authentication request.*
5. Na závěr tlačítko *send.*

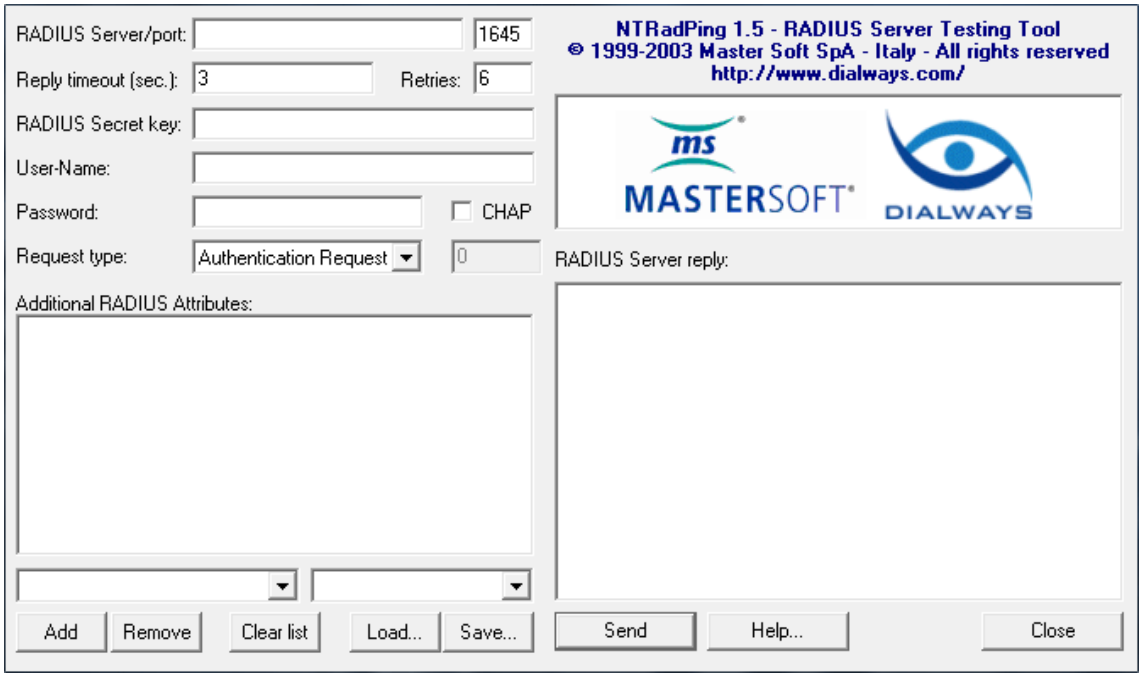
Odpověď serveru má následují podobu:

```
Sending authentication request to server 192.168.126.133:1645
Transmitting packet, code=1 id=1 length=47
Received response from the server in 15 milliseconds
Reply packet code=2 id=1 length=20
Response: Access-Accept
------------------attribute dump----------------------
```

Důležitý je poslední řádek odpovědi – `Response: Access-Accept`, který dává najevo, že byl přístup povolen.

V případě kdy je zadáno chybné heslo pro uživatele obsahuje odpověď serveru informaci o zamítnutí přístupu – `Access-Reject`.

```
Sending authentication request to server 92.168.126.133:1645
Transmitting packet, code=1 id=3 length=47
No response from server (timed out), new attempt (#1)
Received response from the server in 3516 milliseconds
Reply packet code=3 id=3 length=20
Response: Access-Reject
------------------attribute dump----------------------
```



Obr. 2.1: NTRadPing – okno programu

Jde vidět, že pro ověření autentizační funkce serveru je NTRadPing vhodným nástrojem a podle formátu obdržených zpráv lze soudit, že server pracuje správně.

2.2 Kerberos

Za pomoci VMware serveru bylo potřeba vytvořit dva virtuální stroje, kdy jeden sloužil jako KDC (pojmenovaný jako *server*) a druhý zastával úlohu klientského PC (pojemnován jako *client*). Oba stroje pracovali na operačním systému KUBUNTU. Jako kerberos server byla zvolena implementace MIT kerberos.

2.2.1 Nastavení knihoven a KDC

Prvním krokem bylo nainstalování příslušného balíčku *krb5-admin-server* a sním spojené *krb5-kdc*, *krb5-user* a *krb5-config*.

K samotné konfiguraci jsou důležité především dva soubory. Jedním je `krb5.conf`, který ovlivňuje chování serverových i klientských programů Kerbera a druhým je `kdc.conf`, který slouží k nastavení samotného KDC. Nastavení kerbera je rozděleno do několika kroků:

1. Nastavení konfiguračního souboru `krb5.conf`:

```
[logging]
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[realms]
KERBER.TEST = {
kdc = server.kerber.test:88
admin_server = server.kerber.test:749
default_domain = kerber.test
}

[domain_realm]
.kerber.test = KERBER.TEST
kerber.test = KERBER.TEST

[libdefaults]
default_realm = KERBER.TEST
dns_lookup_realm = false
dns_lookup_kdc = false
forwardable = yes
[kdc]
profile = /etc/krb5kdc/kdc.conf
```

V sekci `[realms]` je definováno, kde běží naše KDC a také služba admin serveru pro vzdálenou správu.

`[domain_realm]` je seznamem pro převod domény na název Kerberos realmu[2].

Sekce `[libdefaults]` nastavuje výchozí realm pro klientské programy, dále např. životnost lístku apod.

`[kdc]` určuje, kde se nachází konfigurační soubor KDC.

2. Konfigurace `kdc.conf`:

```
[kdcdefaults]
    kdc_ports = 88

[realms]
    KERBER.TEST = {
     profile = /etc/krb5.conf
     database_name = /etc/krb5kdc/principal
     admin_database_name = /etc/krb5kdc/kadm5_adb
     admin_database_lockfile = /etc/krb5kdc/kadm5_adb.lock
     admin_keytab = FILE:/etc/krb5kdc/kadm5.keytab
     acl_file = /etc/krb5kdc/kadm5.acl
     key_stash_file = /etc/krb5kdc/.k5stash
     kdc_ports = 88
     kadmind_port = 749
     max_life = 10h 0m 0s
     max_renewable_life = 7d 0h 0m 0s
     master_key_type = des-cbc-crc
     supported_enctypes = des-cbc-crc:normal des:v4
     }
```

`[kdcdefaults]` – výchozí port pro KDC.

`[realms]` obsahuje nastavení pro jednotlivé realmy kerbera. Kromě cest k důležitým souborům, je zde také nastaven port pro KDC a pro admin server a maximální doba platnosti lístku. Položka `supported_enctypes` obsahuje seznam typů šifrování, které bude KDC podporovat.

3. Zavedení databáze:

   Databáze se vytvoří příkazem `kdb5_util`. Je potřeba zadat master key.

[2] *Realm* – v podstatě se jedná o doménu, ve které se nacházejí účty. Jméno realmu odpovídá názvu DNS domény převedené na velká písmena.

```
root@server ~: kdb5_util create -s

Loading random data
Initializing database '/etc/krb5kdc/principal' for realm 'KERBER.TEST',
master key name 'K/M@KERBER.TEST'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
```

4. Vytvoření Administrátora:

   Příkazem `kadmin.local` lze vytvořit administrátora pro správu databáze.

```
root@server ~: kadmin.local -q "addprinc kerber/admin"

Authenticating as principal root/admin@KERBER.TEST with password.
WARNING: no policy specified for kerber/admin@KERBER.TEST;
defaulting to no policy
Enter password for principal "kerber/admin@KERBER.TEST":
Re-enter password for principal "kerber/admin@KERBER.TEST":
Principal "kerber/admin@KERBER.TEST" created.
```

5. Spuštění serveru:

   Nejprve spuštění admin serveru a kdc

```
root@server ~: service kadmin start
root@server ~: service krb5kdc start
```

   a poté vstup do samotné administrace

```
root@server ~: kadmin -p kerber/admin
```

6. Tvorba uživatele:

   Pomocí příkazu `addprinc` je potřeba vytvořit uživatele *user*

```
kadmin: addprinc user
WARNING: no policy specified for user@KERBER.TEST;
defaulting to no policy
Enter password for principal "user@KERBER.TEST":
Re-enter password for principal "user@KERBER.TEST":
Principal "user@KERBER.TEST" created.
```

2.2.2 Nastavení klientského PC

Opět bylo potřeba doinstalovat balíček *krb5-clients*. Stejným způsobem jako u serveru došlo k úpravě konfiguračního souboru `krb5.conf`.

Pomocí příkazu `kinit` z klientského stroje lze získat TGT pro uživatele *user*. Příkazem `klist` je pak možné obdržený lístek zobrazit.

```
root@client ~: kinit user
Password for user@KERBER.TEST:

root@client ~: klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: user@KERBER.TEST

Valid starting     Expires            Service principal
05/16/08 11:15:05  05/16/08 21:15:05  krbtgt/KERBER.TEST@KERBER.TEST
       renew until 06/16/08 11:14:52
```

Takto byl klient úspěšně autentizován kerberovským serverem. Z výpisu jde vyčíst životnost lístku, která je 10 hodin.

3 ZÁVĚR

Autentizace, autorizace a účtování jsou dnes tři základní pojmy k zajištění ověřeného a bezpečného přístupu do sítě (ať už vytáčeného, pevného či bezdrátového). Pomocí různých mechanismů je možné přístup povolit či zamítnout, omezit, nebo sledovat chování uživatelů v síti a zaznamenat ho.

RADIUS, i přes řadu svých omezení a nedostatků, patří k nejrozšířenějším AAA protokolům dneška. Velkou zásluhu na tomto má jeho časté spojení se standardem 802.1x, který je v základu implementován v nejpoužívanějším operačním systému dneška – Windows XP a i v nastupujicích Windows Vista. Nemalý podíl na oblíbensti přináší take skutečnost, že RADIUS je otevřený protokol a jeho nasazení a použití není řízeno jedním distributorem.

Protokol TACACS+ vycházející s protokolu TACACS, řeší mimo jiné základní nedostatek RADIUSu, kterým je používání nevhodného transportního protokolu UDP. Výhody TCP jsou jasné – řídí správně přenos packetů a sleduje stav serveru s poskytovanou službou. Podstatné je také jednoznačné oddělení autentizace od autorizace, umožňující nasazení autentizačního systému třetí strany. TACACS+ je ovšem oproti RADIUSu vyvinut a používan firmou Cisco. Není proto tak široce rozšířen, navíc Cisco ve spoustě svých produktů používá také RADIUS.

Jako ve všech technologiích, i mezi AAA protokoly dochází k vývoji. Protokol Diameter byl stvořen za účelem nahrazení RADIUSu a celý je definován převážně na jeho chybách. Používá TCP transportní protokol a vylepšuje zabezpečení. Díky své architektuře, založené na základním protokolu a aplikacích, je rozšiřitelný pro více služeb.

V práci je také uveden protokol Kerberos i přesto, že není typickým AAA protokolem. Nezajišťuje totiž žádným způsobem účtování, a také oproti předchozím, využívá pro autentizaci systém tzv. lístků. Díky tomu bývá nasazován především v nedůvěryhodných sítích kde se přímo předpokládá možnost zachycení informací či odposlouchávání přenosu. I přes svá slabá místa je Kerberos, v kombinaci s některými službami operačního systému, velmi silným autentizačním protokolem.

LITERATURA

[1] NAKHJIRI, Madjid, NAKHJIRI, Mahsa. *AAA and Network Security for Mobile Access : Radius, Diameter, EAP, PKI and IP Mobility. 1st edition.* [s.l.] : Wiley, 2005. 318 s. ISBN 978-0470011942.

[2] DOSTÁLEK, Libor. *Velký pruvodce protokoly TCP/IP : Bezpecnost. 2. aktualiz. vyd.* Brno: Computer Press, 2003. 592 s. ISBN 80-7226-849-X.

[3] BURDA, Karel. *Bezpečnost informačních systémů.* Skripta VUT Brno: 2005. 104 s.

[4] Cisco Systems, Inc. *Cisco AAA Case Study Overview* [online]. URL: `http://www.cisco.com/univercd/home/` [cit. 7. 12. 2007]

[5] RIGNEY, C., et al. *Remote Authentication Dial In User Service (RADIUS)* [online] URL: `http://tools.ietf.org/html/rfc2865` [cit. 20. 4. 2008]

[6] CARREL, D. *The TACACS+ Protocol version 1.78* [online] URL: `http://tools.ietf.org/html/draft-grant-tacacs-02` [cit. 6. 12. 2007]

[7] Calhoun, P., et al. *Diameter Base Protocol* [online] URL: `http://tools.ietf.org/html/rfc3588` [cit. 20. 4. 2008]

[8] Hewlett-Packard Company. *Introduction to Diameter: White Paper* [online] URL: `http://www.docs.hp.com` [cit. 7. 12. 2007]

[9] RICCIARDI, Fulvio. *The Kerberos protocol and its implementations* [online]. URL: `http://www.zeroshell.net/eng/kerberos/` [cit. 7. 12. 2007]

SEZNAM ZKRATEK

AAA authentication, authorization and accounting protocol – autentizační, autorizační a účtovací protokol

RADIUS Remote Authentication Dial In User Service – uživatelská vytáčená služba pro vzdálenou autentizaci

TACACS Terminal Access Controller Access Control System

AVP Attribute Value Pairs

NAS Network Access Server – přístupový server

EAP Extensible Authentication Protocol

CMS Cryptographic Message Syntax

NASREQ Network Access Server REQuirements

KDC Key Distribution Center

AS Autentizační server

MD5 Message–Digest algorithm 5

IPsec IP security

TLS Transport Layer Security

TGS Ticket Granting Server

TGT Ticket Granting Ticket

SSO Single Sign-On

PPP Point to Point protocol

DoS Denial of Service

PAP Password Authentication Protocol

CHAP Challenge Handshake Authentication Protocol

LDAP Lightweight Directory Access Protocol

SEZNAM PŘÍLOH

A PŘÍLOHY

A.1 RADIUS – konfigurační soubory

A.1.1 radiusd.conf

```
##
## radiusd.conf -- FreeRADIUS server configuration file.

prefix = /usr/local/radius
exec_prefix = ${prefix}
sysconfdir = ${prefix}/etc
localstatedir = ${prefix}/var
sbindir = ${exec_prefix}/sbin
logdir = ${localstatedir}/log/radius
raddbdir = ${sysconfdir}/raddb
radacctdir = ${logdir}/radacct

#  Location of config and logfiles.
confdir = ${raddbdir}
run_dir = ${localstatedir}/run/radiusd

#  The logging messages for the server are appended to the
#  tail of this file.
#
log_file = ${logdir}/radius.log
log_destination = files
libdir = ${exec_prefix}/lib
pidfile = ${run_dir}/radiusd.pid

max_request_time = 30

delete_blocked_requests = no

cleanup_delay = 5

max_requests = 1024

bind_address = 192.168.126.133
```

```
#  port: Allows you to bind FreeRADIUS to a specific port.
#
#  The default port that most NAS boxes use is 1645, which is historical.
#  RFC 2138 defines 1812 to be the new port.  Many new servers and
#  NAS boxes use 1812, which can create interoperability problems.
#
port = 0

hostname_lookups = no

allow_core_dumps = no

regular_expressions = yes
extended_expressions = yes

log {
syslog_facility = daemon
}

log_stripped_names = no

log_auth = no

log_auth_badpass = no
log_auth_goodpass = no

usercollide = no

lower_user = no
lower_pass = no

nospace_user = no
nospace_pass = no

# SECURITY CONFIGURATION
#
#  There may be multiple methods of attacking on the server.  This
#  section holds the configuration items which minimize the impact
```

```
#  of those attacks
#
security {
max_attributes = 200
reject_delay = 1
status_server = no
}

# PROXY CONFIGURATION
#
#  proxy_requests: Turns proxying of RADIUS requests on or off.
#
#  The server has proxying turned on by default.  If your system is NOT
#  set up to proxy requests to another server, then you can turn proxying
#  off here.  This will save a small amount of resources on the server.
#
proxy_requests  = yes
$INCLUDE  ${confdir}/proxy.conf

# CLIENTS CONFIGURATION
#
#  Client configuration is defined in "clients.conf".
#

#  The 'clients.conf' file contains all of the information from the old
#  'clients' and 'naslist' configuration files.  We recommend that you
#  do NOT use 'client's or 'naslist', although they are still
#  supported.
#
#  Anything listed in 'clients.conf' will take precedence over the
#  information from the old-style configuration files.
#
$INCLUDE  ${confdir}/clients.conf

# MODULE CONFIGURATION
#
modules {
# PAP module to authenticate users based on their stored password
#
```

```
#  Supports multiple encryption schemes
#  clear: Clear text
#  crypt: Unix crypt
#    md5: MD5 ecnryption
#   sha1: SHA1 encryption.
#     nt: NT-Password encryption
#     lm: LM-Password encryption
#  DEFAULT: crypt
pap {
encryption_scheme = crypt
}

# CHAP module
#
#  To authenticate requests containing a CHAP-Password attribute.
#
chap {
authtype = CHAP
}

# Pluggable Authentication Modules
#
pam {
pam_auth = radiusd
}

# Unix /etc/passwd style authentication
#
unix {
cache = no
cache_reload = 600
radwtmp = ${logdir}/radwtmp
}

#  Extensible Authentication Protocol
#
#  For all EAP related authentications.
#  Now in another file, because it is very large.
#
```

```
$INCLUDE ${confdir}/eap.conf

# Microsoft CHAP authentication
#
#  This module supports MS-CHAP and MS-CHAPv2 authentication.
#  It also enforces the SMB-Account-Ctrl attribute.
#
mschap {
authtype = MS-CHAP
}

# Lightweight Directory Access Protocol (LDAP)
#
#  This module definition allows you to use LDAP for
#  authorization and authentication (Auth-Type := LDAP)
#
ldap {
server = "ldap.your.domain"
basedn = "o=My Org,c=UA"
filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"

start_tls = no

dictionary_mapping = ${raddbdir}/ldap.attrmap

ldap_connections_number = 5
}

#  'realm/username'
#
#  Using this entry, IPASS users have their realm set to "IPASS".
realm IPASS {
format = prefix
delimiter = "/"
ignore_default = no
ignore_null = no
}

#  'username@realm'
```

```
#
realm suffix {
format = suffix
delimiter = "@"
ignore_default = no
ignore_null = no
}

#  'username%realm'
#
realm realmpercent {
format = suffix
delimiter = "%"
ignore_default = no
ignore_null = no
}

#  'domain\user'
#
realm ntdomain {
format = prefix
delimiter = "\\"
ignore_default = no
ignore_null = no
}

#  A simple value checking module
#
checkval {
item-name = Calling-Station-Id
check-name = Calling-Station-Id
data-type = string
}

# Preprocess the incoming RADIUS request, before handing it off
# to other modules.
#
#  This module processes the 'huntgroups' and 'hints' files.
#  In addition, it re-writes some weird attributes created
```

```
#  by some NASes, and converts the attributes into a form which
#  is a little more standard.
#
preprocess {
huntgroups = ${confdir}/huntgroups
hints = ${confdir}/hints

with_ascend_hack = no
ascend_channels_per_line = 23
with_ntdomain_hack = no
with_specialix_jetstream_hack = no
with_cisco_vsa_hack = no
}

# Livingston-style 'users' file
#
files {
usersfile = ${confdir}/users
acctusersfile = ${confdir}/acct_users
compat = no
}

# Write a detailed log of all accounting records received.
#
detail {
detailfile = ${radacctdir}/%{Client-IP-Address}/detail-%Y%m%d
detailperm = 0600
}

#  Write a 'utmp' style file, of which users are currently
#  logged in, and where they've logged in from.
#
radutmp {
filename = ${logdir}/radutmp
username = %{User-Name}
case_sensitive = yes
check_with_nas = yes
perm = 0600
callerid = "yes"
```

```
}

radutmp sradutmp {
filename = ${logdir}/sradutmp
perm = 0644
callerid = "no"
}

# attr_filter - filters the attributes received in replies from
# proxied servers, to make sure we send back to our RADIUS client
# only allowed attributes.
attr_filter {
attrsfile = ${confdir}/attrs
}

counter daily {
filename = ${raddbdir}/db.daily
key = User-Name
count-attribute = Acct-Session-Time
reset = daily
counter-name = Daily-Session-Time
check-name = Max-Daily-Session
allowed-servicetype = Framed-User
#return-attribute = Session-Timeout
cache-size = 5000
}

# The "always" module is here for debugging purposes. Each
# instance simply returns the same result, always, without
# doing anything.
always fail {
rcode = fail
}
always reject {
rcode = reject
}
always ok {
rcode = ok
simulcount = 0
```

```
mpp = no
}

expiration {
reply-message = "Password Has Expired\r\n"
}

logintime {
reply-message = "You are calling outside your allowed timespan\r\n"
minimum-timeout = 60
}
#  Execute external programs
#
exec {
wait = yes
input_pairs = request
shell_escape = yes
}

exec echo {
wait = yes
program = "/bin/echo %{User-Name}"
input_pairs = request
output_pairs = reply
shell_escape = yes
}

#  Authorization. First preprocess (hints and huntgroups files),
#  then realms, and finally look in the "users" file.
#
#  The order of the realm modules will determine the order that
#  we try to find a matching realm.
#
#  Make *sure* that 'preprocess' comes before any realm if you
#  need to setup hints for the remote radius server
authorize {
preprocess
chap
mschap
```

```
suffix
eap
expiration
logintime
}

#  Authentication.
#
authenticate {
Auth-Type PAP {
pap
}
Auth-Type CHAP {
chap
}
Auth-Type MS-CHAP {
mschap
}

#  Pre-accounting.  Decide which accounting type to use.
#
preacct {
preprocess
acct_unique
suffix
files
}

#
#  Accounting.  Log the accounting data.
#
accounting {
detail
unix
radutmp
}
```

A.1.2 clients.conf

```
#
# clients.conf - client configuration directives
#
#########################################################################

#########################################################################
#
#  Definition of a RADIUS client (usually a NAS).
#
#  The information given here over rides anything given in the
#  'clients' file, or in the 'naslist' file.  The configuration here
#  contains all of the information from those two files, and allows
#  for more configuration items.
#
#  The "shortname" is be used for logging.  The "nastype", "login" and
#  "password" fields are mainly used for checkrad and are optional.
#

#
#  Defines a RADIUS client.  The format is 'client [hostname|ip-address]'
#
#  '127.0.0.1' is another name for 'localhost'.  It is enabled by default,
#  to allow testing of the server after an initial installation.  If you
#  are not going to be permitting RADIUS queries from localhost, we suggest
#  that you delete, or comment out, this entry.
#
client 192.168.126.101 {
secret = radius123
shortname = TestovaciNAS
}
```

A.1.3 users.txt

```
#
# This file contains authentication security and configuration
# information for each user.
#
# You don't need to specify a password if you set Auth-Type += System
# on the list of authentication requirements. The RADIUS server
# will then check the system password file.
#

Jirka    Auth-Type := local, User-Password== "heslo"

# The rest of this file contains the several DEFAULT entries.
# DEFAULT entries match with all login names.
#
# First setup all accounts to be checked against the UNIX /etc/passwd.
# (Unless a password was already given earlier in this file).
#
DEFAULT Auth-Type = System
Fall-Through = 1

#
# Defaults for all framed connections.
#
DEFAULT Service-Type == Framed-User
Framed-IP-Address = 255.255.255.254,
Framed-MTU = 576,
Service-Type = Framed-User,
Fall-Through = Yes

DEFAULT Framed-Protocol == PPP
Framed-Protocol = PPP,
Framed-Compression = Van-Jacobson-TCP-IP

DEFAULT Hint == "CSLIP"
Framed-Protocol = SLIP,
Framed-Compression = Van-Jacobson-TCP-IP
```

A.2 Keberos – konfigurační soubory

A.2.1 krb5.conf

```
[logging]
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[realms]
KERBER.TEST = {
kdc = server.kerber.test:88
admin_server = server.kerber.test:749
default_domain = kerber.test
}

[domain_realm]
.kerber.test = KERBER.TEST
kerber.test = KERBER.TEST

[libdefaults]
default_realm = KERBER.TEST
dns_lookup_realm = false
dns_lookup_kdc = false
forwardable = yes

[kdc]
profile = /etc/krb5kdc/kdc.conf
```

A.2.2 kdc.conf

```
[kdcdefaults]
    kdc_ports = 88

[realms]
    KERBER.TEST = {
     profile = /etc/krb5.conf
     database_name = /etc/krb5kdc/principal
     admin_database_name = /etc/krb5kdc/kadm5_adb
     admin_database_lockfile = /etc/krb5kdc/kadm5_adb.lock
     admin_keytab = FILE:/etc/krb5kdc/kadm5.keytab
     acl_file = /etc/krb5kdc/kadm5.acl
     key_stash_file = /etc/krb5kdc/.k5stash
     kdc_ports = 88
     kadmind_port = 749
     max_life = 10h 0m 0s
     max_renewable_life = 7d 0h 0m 0s
     master_key_type = des-cbc-crc
     supported_enctypes = des-cbc-crc:normal des:v4
     }
```