České vysoké učení technické v Praze
Fakulta elektrotechnická

Virtualizovaná univerzální AAA platforma

Diplomová práce

Autor: Richard Kuchár
Vedoucí práce: Ing. Tomáš Vaněk, Ph.D.

Studijní program: Komunikace, multimédia a elektronika
Obor: Sítě elektronických komunikací
Prosinec 2013

Čestné prohlášení

Prohlašuji, že jsem zadanou diplomovou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé diplomové práce nebo její části se souhlasem katedry.

V Praze dne 19.12.2013 .....

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra telekomunikační techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Kuchár Richard**

Studijní program:
Obor: Sítě elektronických komunikací

Název tématu: **Virtualizovaná univerzální AAA platforma**

Pokyny pro vypracování:

Seznamte se s problematikou AAA (autentizce, autorizace, accounting) v datových sítích a s protokoly Radius, Diameter, Kerberos, TACACS+. Ve vhodném virtualizačním prostředí zprovozněte a nakonfigurujte vybranou linuxovou distribuci obsahující serverové části těchto autentizačních protokolů. Vytvořte webové rohraní integrující ovládání a konfiguraci těchto služeb. Funkčnost systému ověřte vůči síťovým prvkům Cisco, HP/3Com, Mikrotik, Huawei a Juniper.

Seznam odborné literatury:

[1] Smith, R.E.:Authentication - From Passwords to Public Keys, Addison-Wesley, 2005, ISBN:0-201-61599-1
[2] Nakhjiri M.,Nakhjiri D.: AAA and Network Security for Mobile Access, John Wiley & Sons, 2006, ISBN:978-0-470-01194-2
[3] Hassel J.: Radius, O'Reilly, 2002, ISBN: 0-596-00322-6
[4] Garman, J.: Kerberos, O'Reilly, 2003, ISBN: 0-596-00403-6
[5] RFC4120, The Kerberos Network Authentication, http://tools.ietf.org/html/rfc4120 Service (V5),
[6] RFC3558, Diameter Base Protocol, http://tools.ietf.org/html/rfc3588

Vedoucí: Ing. Tomáš Vaněk, Ph.D.

Platnost zadání: do konce zimního semestru 2013/2014

L.S.

prof. Ing. Boris Šimák, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 5. 11. 2012

Anotace

Tato diplomová práce se zabývaá základním popisem a implementací AAA protokolů RADIUS, Diameter, TACACS+ a Kerberos ve virtualizovaném prostředí. V rámci práce je připravené webové rozhraní pro účely výuky a testování jednotlivých protokolů. Výsledná platforma je testována vůči zařízením výrobců Cisco Systems, Inc., Juniper Networks, Inc. a MikroTik, přičemž práce také dodává základní konfiguraci těchto zařízení a srovnání podpory jednotlivých protokolů.

Summary

This diploma thesis deals with basic description and implementation of AAA protocols RADIUS, Diameter, TACACS+ and Keberos in virtual environment. Web interface is set up for education purposes and testing of the protocols. The final platform is tested with devices from vendors Cisco Systems, Inc., Juniper Networks, Inc. and MikroTik. The basic configuration for tested devices and protocols is provided as a part of this thesis. In the end we compare the level of support for these protocols by the selected vendors.

Klíčová slova

AAA, RADIUS, Diameter, TACACS+, Kerberos, autentifikace, autorizace, účtování, virtuálni prostředí

Keywords

AAA, RADIUS, Diameter, TACACS+, Kerberos, autentification, authorization, accounting, virtual environment

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Rozvoj sítí a osobních počítačů sebou přinesl také nové bezpečnostní problémy. Uživatelská zařízení se stávala univerzálnější a umožnila uživatelům personalizaci. Uživatel si tak mohl přizpůsobit svá zařízení svým požadavkům. Mohl ale také upravit svá zařízení, aby se v sítích tvářila jako zařízení jiných uživatelů, a získat tak přístup ke službám, které mu nebyly dostupné a nebo za ně platil někdo jiný. Pro správce služeb a sítí tak vznikl nový problém. Jak jednoznačně identifikovat uživatele nebo zařízení v síti a zabránit tak neoprávněným přístupům[4].

V této práci se budeme zabývat standardně používanými protokoly v telekomunikacích, které vznikly jako odpověď na tuto problematiku. Představíme si generický protokol AAA a následně si ukážeme protokoly z něj vycházející (RADIUS, Diameter, Kerberos, TACACS+). Komplexní popis každého protokolu je na samostatnou práci. Pro účely naší práce se budeme soustředit jenom na jejich základní vlastnosti, princip funkčnosti a rozdíly mezi nimi.

Po obeznámení se s teorii si vyzkoušíme jejich praktickou implementaci v širokém spektru zařízení od různých výrobců. Konfiguraci zařízení a jejich funkčnost si vyzkoušíme vůči virtuálnímu serveru, který připravíme tak, aby dokázal komunikovat každým protokolem a byl univerzální pro správu uživatelů a správu přístupu ke službám.

2 AAA

AAA je protokol určen pro identifikaci uživatelů nebo zařízení a k řízení jejich přístupů ke službám nebo funkcím sítě. Je definovaný komisí Internet Engineering Task Force (dále jen IETF) jako RFC 2903[6]. Název AAA je zkratkou pro: „Authentification, Authorization and Accounting" neboli: „Autentizace, Autorizace a Účtování", což jsou tři části definující funkci protokolu. V nasledujícím textu si, podle knihy [7], uděláme stručný přehled AAA protokolu a vysvětlíme si jeho základní principy. Zájemce o podrobnější informace odkážeme na už zmiňované zdroje [7] a [6].

2.1 Terminologie

Na obr. 1 vidíme základní entity protokolu a jejich vzájemnou interakci. **Uživatel**[1], někdy také označován jako suplikant[2], se snaží získat přístup k síti a jejím službám. Uživatel je do sítě připojen prostřednictvím přístupového zařízení, které tvoří hranici mezi uživatelem a sítí poskytovatele. Je proto logické, že toto zařízení bude uživatele identifikovat a následně mu umožní nebo zamítne přístup do sítě. Bude se tedy chovat jako **AAA klient**[3]. AAA klient sám o sobě nemá žádnou autoritu pro udělení povolení přístupu, ale působí jako strážce, kerý se při pokusu uživatele o přístup zeptá autority, jestli ho pustit nebo ne a na základě tohoto rozhodnutí umožní uživateli využívat služby sítě. AAA klient také dohlíží, aby uživatel využíval jenom ty služby, pro které je autorizován.

AAA server je autorita, které se AAA klient dotazuje. AAA server rozhoduje na základě porovnání informací poskytnutých uživatelem AAA klienta a informací v databázi[4], jestli má být uživatel do sítě vpuštěn nebo ne a případně s jakými restrikcemi. AAA server a AAA klient spolu komunikují prostřednictvím **AAA protokolu**, jako např. RADIUS. Komunikace může probíhat přes další zařízení poskytovatele, proto je potřeba komunikaci zabezpečit.

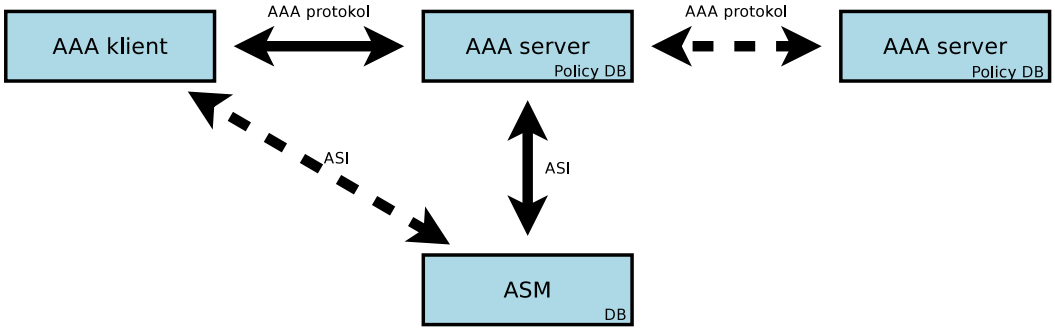
Služby nebo funkce sítě, které uživatel využívá, vyžadují také informace o uživateli, zejména jestli k nim má uživatel povolen přístup. Spolupracují proto s AAA serverem, který jim potřebná data poskytne. Specifikace proto zavádí Application-Specific Module (dále jen **ASM**), který je abstrakcí komunikace mezi těmito službami, AAA aplikacemi a AAA serverem. ASM a AAA server komunikují prostřednictvím Application-Specific Information (dále jen **ASI**) abstrakčnímu API. AAA server vyhodnotí požadavek od uživatele o pří-

[1] V celé práci budeme striktně rozlišovat uživatele od klienta. Uživatel je fyzická osoba využívající prostředky sítě. Klient je zkrácený název pro AAA klient

[2] Pod pojmem suplikant se myslí software nebo zařízení, prostřednictvím něhož se uživatel identifikuje v síti. V případě počítačových sítí je to ovladač v operačním systému, v telefonních sítích je to modem.

[3] Jelikož AAA klient umožní uživateli přístup k síti, nazývá se někdy také **NAS** (Network Acess Server, neboli Server síťového přístupu).

[4] AAA server může mít víc databází zaměřených na různé informace. Jedna databáze tak může obsahovat informace o uživatelských přístupech a jiná databáze zase informace o povolených službách a restrikcích pro daného uživatele. Druhou zmiňovanou budeme nazývat Policy DB.



Obrázek 1: Interakce základních komponent generického AAA serveru

stup ke specifické službě pomocí své databáze a informaci předá prostřednictvím ASI na příslušný ASM.

AAA server může také dostat požadavek na identifikaci od uživatele kterého nemá ve své databázi. Požadavek tak může odmítnout nebo ho může přeposlat na jiný AAA server, o kterém ví, že by mohl znát odpověď. Takovéto předávání požadavku může pokračovat dál, dokud některý z AAA serverů, kterému je požadavek předán, nerozhodne o jeho akceptaci nebo zamítnutí. AAA server, který požadavek předal dál se nazývá **AAA proxy**.

Poslední pojem, který si zavedeme je **realm**. Realmem označujeme zařízení patřící do stejné sítě. Jako název realm-u se často používá název domény přidělené dané síti. Realm by měl být pro každou síť jedinečný, aby bylo možné odlišit uživatele z různých realmů. **Roamingem** budeme nazývat proces, kdy se uživatel bude připojovat do své sítě prostřednictvým jiného realm-u.

2.2 Autentifikace (Autentification)

Autentifikace je proces prokázaní identity autentifikovaného. Potkáváme se s ním denně v bance, na poště, letišti, apod. Je proto zřejmé, že se musí provádět i při využívání elektronických služeb. Bez autentifikace nemůže poskytovatel služby určit, jestli je daný uživatel oprávněn službu využívat nebo nikoli. Proto tvoří autentifikace první písmeno „A" protokolu AAA.

Rozeznáváme tři typy autentifikace:

- **Client autentification** (autentifikace klienta)
  Uživatel poskytne údaje, na jejichž základě ho služba ověří.

- **Message autentification** (autentifikace zprávy)
  Zpávy, které uživatel posílá, mohou být v síti ovlivněny útočníkem nebo také chybou. Uživatel proto ke zprávě přiloží údaje, pomocí kterých je možné ověřit autenticitu zprávy po jejím přijetí.

- **Mutual autentification** (vzájemná autentifikace)
  V případě drátových (pevných) sítí se dá předpokládat, že se uživatel připojuje do správné sítě, kterou chce využívat. Jenomže veřejné sítě, jako je například wifi v kavárnách, mohou byt lehce napadeny útočníkem a komunikace uživatele tak může procházet přes naslouchací zařízení útočníka. Proto síť poskytne jedinečný identifikátor, kterým si ji uživatel ověří, a pokud je ta správná, odešle svou identifikaci potřebnou k autentifikaci. Vzájemně se tedy identifikuje síť uživateli a uživatel síti.

Autentifikace může probíhat dvěma postupy:

- **Two-party** (se dvěma účastníky)
  Ověřovatel přímo ověří ověřovaného (např. při výdeji zásilky na poště).
- **Three-party** (se třemi účastníky)
  Ověřovatel se zeptá nadřízené osoby, která má autoritu rozhodnout, jestli je ověření platné a až pak ověřovaného ověří (např. při vstupu do klubu se ochranka zeptá vedoucího podniku, jestli může danou osobu vpustit). Tento model je použit i v protokolu AAA, jak sme si popsali výše.

2.3 Autorizace (Authorization)

Autorizace tvoří druhé písmeno „A" protokolu AAA. Nové trendy a služby si žádají vyšší pozornost této funkci navzdory tomu, že se jí původní specifikace příliš nevěnuje. Co to vlastně znamená a v čem se liší od autentifikace, si uvedeme na příkladu studenta na univerzitě. Student prochází při vstupu přes turniket, který ho autentifikuje a vpustí do budovy nebo zamítne jeho vstup (např. pokud se jedná o studenta, který své studium již ukončil). Student s povoleným přístupem se pak může volně pohybovat po univerzitě. Když však ale bude chtít vstoupit do některé laboratoře nebo posluchárny, tak se musí autorizovat, jestli do ní má nebo nemá přístup. Ne každý student může vstoupit do všech laboratoří a poslucháren, tento přístup získává na základě potřeb svého studia.

Uživatel tedy komunikuje s AAA serverem aby se autorizoval a získal přístup k požadované službě. Rozpoznáváme tři scénaře komunikace:

- **Agent sequence** (pomocí prostředníka)
  Uživatel pošle požadavek na službu AAA serveru, který jej ověří. Pokud je požadavek akceptován, informuje AAA server požadovanou službu a po jejím nastavení informuje uživatele, že jeho požadavek byl akceptován. AAA server se tedy chová jako agent.
- **Pull sequence** (na vyžádaní)
  Uživatel pošle požadavek přímo službě. Ta ho ale potřebuje autorizovat, proto si vyžádá od AAA serveru autorizaci požadavku. Podle odpovědi od AAA serveru se rozhodne a toto rozhodnutí sdělí uživateli.

- **Push sequence** (prokázáním)
  Uživatel zašle tiket nebo certifikát, který získal autorizací, službě a prokáže tak, že na ni má oprávnění.

2.4 Účtování

Poslední „A" z názvu AAA protokolu patří účtování (accounting). Jedná se o proces sběru dat a informací o využívaných prostředcích pro všechny nebo specifické částí sítě. Sesbíraná data mohou být použita např. pro účely auditu, alokace cen a prostředků a pro analýzu vývoje trendů.

Data sbírá zařízení poskytující službu a předává je AAA serveru. Různé aplikace mají různé požadavky na typ nebo podrobnost dat, proto se AAA protokol snaží poskytnout nástroje pro všechny tyto potřeby.

Sběr dat ze zařízení může probíhat dvěma způsoby:

- **Polling** (na dotaz)
  AAA server se dotazuje zařízení poskytujících službu v pravidlených intervalech. Tento způsob může mít výkonnostní problém v případě, že server potřebuje data pro určitého uživatele, protože musí posbírat data ze všech zařízení současně, a to může značně zatížit síť.
- **Event-Driven**(Řízený událostí)
  Zařízení oznámí serveru, když jsou data dostupná, a ten si je následně převezme. Neodeslaná data se ukládají v paměti zařízení, avšak jenom po dobu stanoveného intervalu. Po tomto intervalu se paměť pročistí a uvolní pro nová data.

Data získaná tímto procesem se často používají jako podklady pro fakturaci využívaných služeb, proto je potřeba zabezpečit:

- že data mohou číst, případně je modifikovat, jenom autorizované osoby,
- autentifikaci zdroje dat,
- ochranu dat při přenosu od zařízení poskytujícího službu k AAA serveru,
- ověřitelnost korektnosti dat.

Při rozhodování nad použitím konkrétního protokolu pro účtování pak hraje vysokou roli spolehlivost přenosu dat od jejich zdroje k AAA serveru. Docílit toho můžeme třemi mechanismy:

- **Interim (přírůstek)**
  AAA klient periodicky posílá změny na server. Při výpadku spojení se tak může obnovit spojení v bodě, ve kterém byla poslána poslední změna.

- **Transport protocol** (transportní protokol)
  Data poškozená při přenosu nebo nedoručená potřebujeme získat zpět. To nám může zabezpečit vhodná volba transportního protokolu, nebo musí AAA protokol tuto funkci zabezpečit sám.

- **Fail-Over mechanism** (opravný mechanismus)
  Při selhání primárního serveru se data automaticky začnou ukládat na server záložní.

2.5 Požadavky AAA protokolu na NAS zařízení

Existuje mnoho různych NAS zařízení od různych výrobců a jsou na ně administrátory sítě kladeny různe nároky na podporu AAA. Proto vznikla specifikace RFC 3169[1], která definuje požadavky na AAA protokoly a NAS. Některé z těchto požadavků si uvedeme:

- AAA protokol musí být přenositelný na třetí (síťové) a čtvrté (transportní) vrstvě OSI modelu jako protokol sedmé (aplikační) vrstvy.

- AAA protokol musí být schopný detekovat chybu přenosu do definovatelné časové periody.

- AAA protokol musí umožnit změnu spojení mezi NAS a AAA serverem z primárního, který selhal, na sekundární. Při změně musí být zachována relace původního spojení.

- AAA protokol musí umožňovat vzájemnou autentifikaci mezi NAS a AAA serverem.

- AAA protokol musí umožňovat selektivní enkripci určitých atributů.

- NAS a AAA protokol musí umožňovat přenos přihlašovacích údajů mezi uživatelem a AAA serverem. Přenos musí být skryt před entitami, které nepatří do AAA, jako je třeba směrovač (router) mezi NAS a AAA serverem.

- NAS a AAA protokol musí umožňovat více fázové přihlášení.

- AAA protokol musí poskytovat jedinečný kód lokace NAS a časové známky požadavků pro potřeby autentifikace a auditu požadavků.

- AAA protokol musí poskytovat prostředky na přidělení IP adresy a pro aplikaci IP filtrů.

- NAS a AAA protokol musí podporovat doručení účtovacích informací po určitých událostech, jako je počátek/konec relace, reutorizace, expirace predefinovaných intervalů.

3 AAA protokoly

V kapitole 2 jsme si popsali generický AAA protokol. Nyní se podíváme na konkrétní implementace autentifikačních protokolů které se řadí, byť ne všechny tam patří, do skupiny AAA protokolů[5]. Budeme se věnovat jejich základním principům tak, abychom je byli později schopni implementovat.

3.1 RADIUS

Ačkoliv je RADIUS (Remote Authentication Dial In User Service) označován jako AAA protokol, jeho vznik a formalizace v podobě RFC dokumentu se datuje několik let před vznikem AAA. RADIUS vznikl na popud společnosti Merit Network, která čelila na své síti NSFnet problémům s různými typy proprietárních metod autentifikace pro různá zařízení. Spojila se proto se společností Livingston, výrobcem síťových zařízení, a po několika sezeních vznikl RADIUS. Ten byl později zformalizován v podobě RFC 2058 a přijat dalšími společnostmi jako standard.

Navzdory novějším protokolům, napsaným dle AAA, je RADIUS stále primárně nasazovaným protokolem, a to především díky své široké implementaci mezi výrobci a možnostem jeho rozšiřitelnosti.

Specifikace protokolu RADIUS je obsáhlá a není tématem této práce ji podrobně rozebrat. Detailnější popis najdeme v knize [5] nebo v [9].

3.1.1 Terminologie

Uvedeme si základní termíny používané ve výkladu RADIUS protokolu

RADIUS klient je, podobně jako AAA klient, zařízení nebo software, který komunikuje s RADIUS serverem. Nesmíme si ho plést s uživatelem, který se prostřednictvím RADIUS klienta autentifikuje. V následujícím textu budeme používat zkrácené označení klient.

RADIUS server dále jen server, je centrální bod RADIUS protokolu, který přijímá požadavky od klienta na autentifikaci a rozhoduje o jejich přijetí nebo zamítnutí.

HOP-by-HOP je model předávání si požadavků mezi servery. Jeho princip spočívá v postupném předání požadavku na další server (HOP) se všemi daty v požadavku obsaženými. Po dosáhnutí cíle se odpověď vrací stejným způsobem ke klientovi.

AVP Attribute-Value Pair je položka obsažená v těle paketu informující příjemce paketu o vlastnostech nebo charakteru požadavku. Je složen z

[5] V následujícím textu budeme používat jenom zkrácený zápis AAA.

- číslo - identifikace atributu
- délka - délka celého AVP
- hodnota - vlastnost nebo charakteristika daného atributu

VSA Vendor-Specific Attributes je speciálni typ atributů specifických pro konkrétního výrobce. Jejich struktura je shodná s AVP, jenom je doplněna o položku Vendor ID určující konkrétního výrobce. VSA atributy jsou vloženy jako hodnota AVP určeného pro tyto účely.

realm identifikátor domácí sítě. Nejčastěji se používá doména sítě

secret náhodně vygenerovaná hodnota známá oběma stranám komunikace (serveru i klientovi) a slouží k zabezpečení paketů proti podvržení útočníkem.

3.1.2 Vlastnosti protokolu

RADIUS protokol neudržuje informace o relacích, je tedy state-less (bezstavový). Komplikuje to správu zdrojů a relací, protože každý doplňující požadavek musí být odeslán jako nový, tedy musí obsahovat údaje k ověření uživatele, informaci ke kterým službám požaduje přístup atd.

Nemá podporu pro znovu získání a dealokaci zdrojů po autorizaci. Někteří výrobci proto implementují mechanismus odpojení uživatele po definovaném časovém intervalu, ten se pak musí znovu autorizovat.

Pro přenos zpráv po síti se používá UDP transportní protokol místo protokolu TCP. Nevýhody použití UDP protokolu řeší nad transportní vrstvou. Každý požadavek je zpětně potvrzen do stanoveného časového limitu a pokud se tak nestane, odešle klient uloženou kopii požadavku na sekundární server. Naopak RADIUS těží z UDP, a to především rychlejší odezvou na požadavek (a to i v případě ztráty paketů) a multithredovostí, kdy klient nemusí řadit požadavky pro každou relaci sekvenčne, ale odeslat je hned jak přijdou.

Staví na modelu HOP-by-HOP, což snižuje jeho zabezpečení, jelikož všechna data jsou viditelná na každé proxy, kterou požadavek prochází, a to včetně uživatelského jména, hesla nebo certifikátu.

Podporuje autentifikaci pomocí metod PAP, CHAP a v novějších verzích také EAP.

Obsahuje více než 50 atributů (AVP) s možným rozšířením o další, pro výrobce specifické, atributy (VSA).

RADIUS byl navržen s ohledem na potřeby roamingu. Při identifikaci proto používá řetězec složený z uživatelského jména a realm-u, nejčastěji od sebe oddělenými zavináčem.

3.1.3 Struktura paketů

Už jsme zmínili, že RADIUS používa UDP protokol, konkrétně porty 1812 pro autentifikaci a 1813 pro účtování (v původní specifikaci byly použity porty 1645 a 1646). Struktura paketu (obr. 2) se skládá z položek:

code (kód)
Kód určující typ zpávy, pro kterou je paket sestaven. Poznáme tyto typy paketů:

- Access-Request - žádost o autentifikaci
- Access-Challenge - používá se pri ověřovací metodě CHAP
- Access-Accept - uživatel je ověřen
- Access-Reject - uživatel nebyl ověřen
- Accounting-Request - statistická data pro funkci účtování
- Accounting-Response - potvrzení příjmu statistických dat

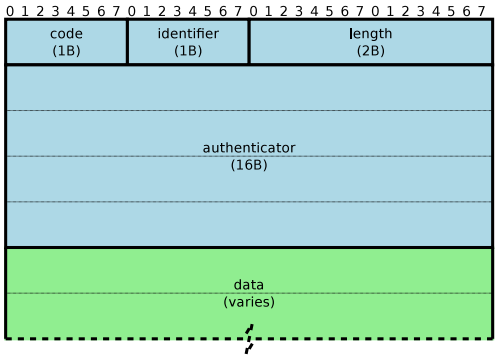
identifier (identifikátor)
Spolu s informacemi z UDP hlavičky, konkrétně zdrojové IP adresy a zdrojového portu, slouží identifikátor ke spárování odpovědi ze serveru s původním požadavkem.

length (délka)
Uvádí celkovou délku paketu (přípustné hodnoty jsou mezi 20 - 4096), podle které si příjemce ověří integritu dat. Pokud je přijatý paket delší, tak data navíc ignoruje. Paket kratší než udávaná délka se tiše ignoruje.

authenticator (ověřovací řetězec)
Slouží k ověření integrity a korektnosti datové části. Rozeznáváme dva typy ověřovacího řetězce:

- RequestAuth - Používá se pro zprávy Access-Request a Accounting-Request. Je vyplněný náhodně vygenerovaným 16B řetězcem nebo v případě Accounting-Request paketů vypočten jako MD5 hash dle vzorce:
  $RequestAuth{=}MD5(Kód{+}Identifikátor{+}Délka{+}zeros(16B){+}Data{+}Secret)$
- ResponseAuth - Používá se pro ostatní typy zpráv. Je vypočten jako MD5 hash dle vzorce:
  $ResponseAuth{=}MD5(Kód{+}Identifikátor{+}Délka{+}RequestAuth{+}Data{+}Secret)$

data (tělo paketu)
Obsahuje žádný a více AVP (Attribute-Value Pair) popisujících vlastnosti a charakteristiky celého AAA procesu. Který AVP je použit pro který paket a další restrikce, je přesně popsáno v RFC.



Obrázek 2: Struktura paketu RADIUS protokolu

3.1.4 Autentifikace a Autorizace

Autentifikace a autorizace je v podání RADIUS protokolu jedno a to samé. Na základě žádosti uživatele o přístup ke službě odešle klient Acess-Request paket s AVP s informacemi o identitě uživatele, ověřovací metodě, kterou má použít, a službách, ke kterým požaduje uživatel přístup. Server ověří identitu uživatele podle specifikované metody. V případě ověřovací metody CHAP odešle Access-Challenge paket. Po úspěšném ověření odešle paket Access-Accept paket, který může obsahovat AVP autorizující uživatele ke všem službám jemu povoleným. Pokud neobsahuje žádny AVP, je uživateli povolena jenom služba, o kterou žádal. Neúspěšné ověření je doprovázeno paketem Access-Reject.

3.1.5 Účtování

Víme, že sběr dat o využívání služby uživatelem je důležité zabezpečit proti ztrátě dat nebo nežádoucí minipulaci s daty. RADIUS tuhle problematiku řeší šifrovaním dat pomocí sdíleného Secret řetězce, potvrzením o doručení a v případě problému odesláním kopie na záložní účtovací server. Ale ani tyto vlastnosti nezaručují absolutní spolehlivost. Kopie dat jsou uchovávány jenom určitou předem stanovou dobu a pokud sa nepodaří získat potvrzení o přijetí požadavku od některého účtovacího serveru, tak je požadavek ztracen. Nová specifikace řeší i tenhle problém zavedením specifických AVP implementujících Interim.

Po úspěšné autorizaci oznámí RADIUS klient začátek sběru dat odesláním Accounting-Request paketu s AccountingStart AVP serveru, který potvrdí přijetí požadavku pomocí Accounting-Response. Po dokončení sběru dat se informace odešlou opět prostřednictvím Accounting-Request paketu, který obsahuje AccounintgStop AVP a další atributy pro jednotlivá statistická data. Zabezpečení dat probíhá pomocí sdíleného Secret řetězce, kterým jsou data skryta před neautorizovaným pozorovatelem.

RADIUS účtovací server může a nemusí byt totožný s autentifikačním/autorizačním serverem, což přináši vyšší škálovatelnost. Dokonce je možné mít víc účtovacích serverů pro různé požadavky (např jeden pro ISDN uživatele, jeden pro LAN a další pro WLAN uživatele). Účtovací server může fungovat také jako proxy a přeposílat kopii požadavku na cílový

server a až ten následně odešle Accounting-Response odpověď. Je nutno poznamenat, že kopie dat je také uložená na proxy serveru, tedy data se duplikují.

3.2 Diameter

RADIUS protokol byl po poslední úpravě v roce 2000 považován za dokončený. Ve stejném roce započala skupina zodpovědná za specifikaci AAA hledání jeho nástupce. Připravila kompletní požadavky na AAA protokol, na základě kterých nechala vyhodnotit skupinou expertů, který z kandidátů nejlépe vyhovuje požadavkům. Příklad těchto požadavků najdeme v [5].

Hlavními kandidáty byly SNMP, RADIUS++[6] , COPS (Common Open Policy Service Poll) a Diameter. Výsledky studií těchto protokolů, jejich porovnáni a doporučení najdeme v RFC 3127. Pro pořádek uvedeme výherce, tedy protokol Diameter definovaný normou RFC 6733. Oproti druhému v pořadí, protokolu COPS, měl jenom malý náskok díky lépe připravené specifikaci a lepšímu rozlišení firewallem[7].

3.2.1 Vlastnosti protokolu

Diameter zavádí koncept aplikací (služby, protokoly a procedury) využívajících vlastnosti Diameter serveru, proxy a protokolu samotného. Příkladem aplikace je proces autentifikace prostřednictvím NAS. Každá aplikace má své specifické zprávy.

Oproti RADIUS protokolu podporuje také End-to-End model komunikace a využíva TCP nebo SCTP transportní protokol.

3.2.2 Terminologie

Níže si rozvedeme termíny spojené s Diameter protokolem, které budeme používat při popisu funkčnosti protokolu.

End-to-End je model předávání si požadavků mezi servery. Jeho princip spočívá v postupném přesměrování klienta na cílový Diameter server, který pak vyřídí všechnu komunikaci s klientem přímo. Výhodou je, ze jednotlivé HOP-y nevidí důvěrné informace od uživatele, jenom přesměrují (navedou) klienta k cíli.

Peer-to-Peer zkráceně označován P2P, je model komunikace mezi dvěma síťovými zařízeními. Oproti standardnímu modelu klient-server, kde klient se dotazuje a server odpovídá, P2P zrovnoprávňuje obě strany komunikace. Každý peer tedy odesílá jak požadavek tak i odpověď bez ohledu na stav komunikace.

[6] někdy také označován jako RADIUSv2, je vylepšená a propracovaná verze RADIUS protokolu. Není zpětně kompatibilní s původním protokolem a existuje jenom ve formě draftu.

[7] COPS je přimárně určen pro QoS, ve firewallu se proto musí zkoumat pakety uvnitř dat, jestli se jedná o pakety pro AAA nebo QoS zprávy.

Za účelem bezpečnosti, spolehlivosti a směrování jsou jednotlivé funkce Diameter protokolu velmi dobře specifikovány a popsány. Níže si uvedeme entity infrastruktury protokolu Diameter, které mají svůj specifický úkol při výměně zpráv.

Diameter Nod dále jen nod, je program implementující Diameter protokol

Diameter Peer dále jen peer, je nod, který má přímé spojení s dalším nod-em

Diameter Client zařízení na hranici sítě, které provádí kontrolu přístupu do sítě. Příkladem jsou NAS zařízení. Opět platí, že uživatel není Diameter klient, protože se přímo neúčastní na Diameter signalizaci. V textu budeme používat zkrácený název klient.

Diameter Server zařízení zpracovávající AAA požadavky pro konkrétní realm. Diameter server, dále jen server, musí podporovat všechny aplikace používané pro daný realm.

Diameter Agent dále jen agent je nod, který poskytuje služby předávání (relay), zastupování (proxy), přesměrování (redirect) nebo překladu (translation).

- Relay agent přeposílá zprávu na základě informací obsažených ve směrovacích atributech. Relay nikdy neinicializuje komunikaci s dalšími entitami Diameter infrastruktury.
- Proxy agent můžeme se na ně dívat jako na Relay agenta, který také může vykonat rozhodnutí na základě dříve stanovené politiky a na jejím základě také může požadavek zamítnout
- Redirect agent odkazuje klienta na server na základě své konfigurace, která může také omezovat tuto funkčnost jenom na určitý typy požadavků
- Translation agent vykonává překlad Diameter protokolu na jiný AAA protokol, jako např. RADIUS

3.2.3 Struktura paketů

Na rozdíl od RAIDUS-u je Diameter P2P a také nerozeznává typ paketů jako to dělá RADIUS, ale zavádí koncept příkazů, které specifikují funkčnost a akci, která se má provést.

Na obr. 3 vidíme strukturu Diameter paketu (příkazu), která se skládá z:

version (verze)
Indikuje verzi protokolu Diameter, aktuálně se používá pouze číslice jedna.

length (délka)
Délka celého paketu včetne těla paketu.

command flags (příznaky)

Definují vlastnosti přikazu čtyřmi příznaky:

- R - určuje, jestli se jedná o požadavek nebo odpověď,
- P - určuje, jestli může být příkaz přesměrován některým agentem, nebo musí být zpracován lokálně,
- E - určuje, jestli příkaz obsahuje chyby,
- T - používá se k odstranění duplicitních příkazů nebo může určovat, jestli se jedná o příkaz přeposlaný po poruše na lince.

command code (kód příkazu)

Určuje, jakou akci má příjemce vykonat pro danou zprávu, např. zruš požadavek.

application ID (ID aplikace)

Typ aplikace, pro kterou je příkaz určen.

HOP-by-HOP identifier (HOP-by-HOP identifikátor)

Obsahuje identifikátor, který páruje požadavek s odpovědí. Odesílatel požadavku se musí ujistit, že identifikátor je jedinečný, a odesílatel odpovědi se musí ujistit, že identifikátor je stejný jako identifikátor v požadavku, na který odpovídá.

End-to-End identifier (End-to-End identifikátor)

Je určen pro identifikaci duplicitní zprávy. Identifikátor odpovědi musí být totožný s identifikátorem požadavku a musí být jedinečný minimálně čtyři minuty. Duplicitní požadavek může vyvolat duplicitní odpověď, ale tato odpověď nesmí ovlivnit žádný stav vyvolaný původním požadavkem.

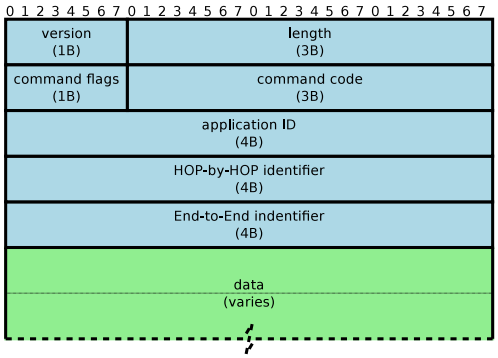
data (tělo paketu)

Obsahuje dodatečné informace zprávy obsažené v AVP. Struktura AVP je složená z:

- AVP code (AVP kód) - identifikuje atribut, kódy jsou jednotlivým atributům přiděleny společností IANA,
- flags (příznaky) - určuje vlastnosti AVP pomocí bitů. V označuje atributy specifické pro výrobce (nedefinované standardem), M označuje povinný atribut[8], P označuje požadavek na zabezpečení end-to-end spojení, pokud prochází přes Diameter agenta[9], R indikuje zatím nepoužité příznaky, tudíž rezervované pro další použití
- AVP length (délka) - délka celého AVP

[8] Je nastaven, pokud Diameter Nod vyžaduje, aby jeho protistrana (příjemce) podporovala daný atribut. Pokud Diameter Nod příjme takový příkaz a vyžadovaný atribut nerozezná, musí zahodit celý paket.

[9] požadavek nesmí být odeslán, pokud nebude spojení zabezpečeno mezi původcem požadavku a jeho příjemcem.



Obrázek 3: Struktura paketu Diameter protokolu

- Vendor ID (ID výrobce) - nastavuje se pro atributy specifické pro konkrétního výrobce (pokud je nastaven príznak V na jeho ID
- attribute data - hodnota atributu

3.2.4 Aplikace

Ne každý Diameter Nod musí podporovat všechny aplikace, proto diameter obsahuje ověřování podpory aplikací mezi nod-y. Aby to bylo možné, má každá aplikace unikátní identifikátor přiřazený a spravovaný spoločností IANA. Níže si uvedeme příklady vybraných aplikací:

Diameter Účtování

Diameter NAS application detailněji popisuje interakci Diameter serveru s NAS pro autentifikaci a další procedury. Podporuje různé autentifikace - PPP, CHAP, EAP

Diameter EAP definuje procedury přenosu EAP zpráv po Diameter zprávách

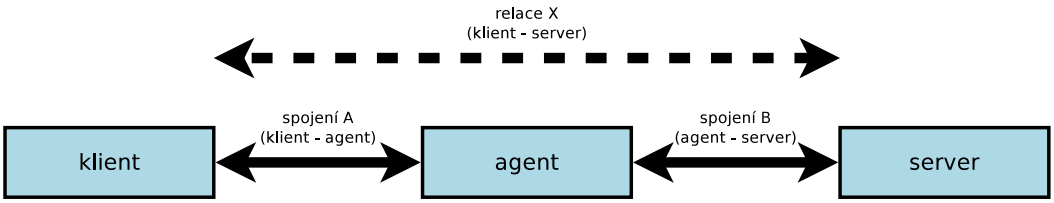
3.2.5 Principy přenosu a směrování

Diamater specifikace rozlišuje mezi relací a spojením. Spojení je navázáno mezi jednotlivými nod-y a může být po nich přenášeno více různých relací. Spojení jsou navázána a kontrolována transportním protokolem (TCP nebo SCTP). Oproti tomu relace je logická a vzniká mezi klientem a serverem pro posílání jednotlivých požadavků. Rozdíly lépe ilustruje obr. 4. Relaci identifikuje SessionID AVP.

Směřování se řídí dle Peer Table a realm-based Routing Table. Každý nod udržuje Peer Table obsahující informace o svých peer-ech[10]. realm-based Routing Table si vytváří agenti dynamicky mezi sebou a používají ji ke směrování požadavků do cíle nebo k dalšímu agentovi po nejvhodnější cestě[11].

[10]Peer Table si můžeme představit jako ARP tabulku. Obsahuje informace o identitě peer-a, jestli je peer statický nebo dynamicky objeven a čas vypršení záznamu.

[11]realm-based Routing Table obsahuje položky realm, klíč pro určení cest, seznam aplikací pro danou



Obrázek 4: Spojení a relace Diameter protokolu

3.3 TACACS+

TACACS+ poskytuje kontrolu přístupu k síťovým zařízením prostřednictvím jednoho nebo více centralizovaných serverů. TACAS+ je poslední generací TACACS protokolu původně vyvinutého pro armádu Spojených Států Amerických. Později byl potokol TACACS adoptován společnosti CISCO, která ho několikrát vylepšila a rozšířila, proto je její posledni verze protokolu označována jako XTACACS.

3.3.1 Vlastnosti

TACACS+ vylepšuje oba zmiňované protokoly oddělením autentifikace, autorizace a účtováni (AAA) do samostatných navzájem nezávislých částí. Zavádí také šifrování všech zpráv, které si TACACS+ klient (NAS) vymění s TACACS+ serverem. Protokol používá TCP protokol transportní vrstvy (port 49) a svou strukturou paketů umožňuje použití různých metod autentifikace a podporu pro další. Proces autorizace je dynamický, neposkytuje tedy jenom odpověď ano/ne, ale může také upravit nastavení služby pro daného uživatele[12]. Statistická data posbíraná v procesu účtování obsahují data z procesu autorizace doplněná např. o začátek a konec, využíváni služby apod.

3.3.2 Terminologie

relace je v protokole TACACS+ krátká událost platná jenom po dobu jedné autentifikační, autorizační nebo účetní výměny až do jejího konce. Každá relace má svůj identifikátor, který je použit pro šifrování.

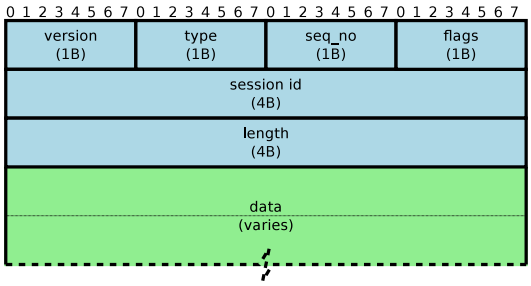
TACACS+ server zařízení na kterém běží démon protokolu TACACS+. Budeme používat jenom zkrácenou verzi názvu, server.

NAS (Network Access Server) je přístupové zařízení, klient TACACS+ serveru.

AVP (Attribute-Value Pair)

cestu, akci, která se má vykonat, seznam peerů pro danou cestu a jako poslední informaci, jestli je záznam dynamický nebo statický.

[12]Příkladem může být restrikce doby, po kterou může být uživatel ke službě připojen.



Obrázek 5: Struktura paketu TACACS+ protokolu

3.3.3 Struktura paketů

Pakety TACACS+ protokolu (obr. 5) jsou složeny z hlavičky a těla (datové části). Hlavička je stejná pro všechny pakety protokolu a skládá se z:

version (verze)
: Verze protokolu je složená z majoritní a minoritní verze protokolu a používá se pro ověření kompatibility mezi NAS a serverem. Pokud server dostane zprávu s verzí protokolu, kterou nezná, vygeneruje chybu ve které oznámí nejvyšší podporovanou verzi.

type (typ)
: Typ informace obsažené v těle paketu, používá se pro rozlišení autentifikačního, autorizačního a účtovacího paketu.

seq_no (pořadové číslo)
: Pořadové číslo paketu pro konkrétní relaci. První paket pro novou relaci musí mít pořadové číslo jedna, každý další paket pro danou relaci musí mít číslo o jedna větší. Pokud se pořadové číslo rovná maximální hodnotě 2^{8-1}, musí se relace ukončit a vytvořit nová.

flags (příznaky)
: Příznaky specifikující vlastnosti paketu, např. jestli je žádán přenos více relací po jednom spojení TCP protokolu, nebo jestli se pro každou relaci naváže nové spojení TCP protokolu.

session_id (id relace)
: Číslo identifikující relaci. Je generováno náhodně, v daném okamžiku musí být jedinečné a nesmí se po dobu trvání relace změnit.

length (délka)
: Délka celého paketu (tedy hlavičky i těla) v Bytech.

data (tělo)
: Tělo paketu je vždy šifrováno[13] a dělíme ho dle typu a směru kterým je posílán:

[13] Mechanismus šifrování najdeme v [2]. Šifrování je možné také vypnout pro účely ladění.

1. Autentifikační
   (a) START - Paket posílán od NAS na server jako oznámení o počátku procesu autentifikace (první požadavek). Tělo obsahuje AVP specifikující vlastnosti a stav NAS a také informace pro přihlášení závislé na typu přihlášení (CHAP, PPP, EAP, ...).
   (b) REPLY - Odpověď serveru na požadavky v procesu autentifikace. Tělo obsahuje zprávu která může oznámit schválení nebo zamítnuti autentifikace anebo může obsahovat zprávu vyžadující dodatečné informace.
   (c) CONTINUE - Odpověď NAS na REPLY paket požadující dodatečné informace. Tělo obsahuje zprávu, která je odpovědí na vyžádané informace. Může také obsahovat zprávu oznamující ukončení relace.
2. Autorizační
   (a) REQUEST - Paket posílaný z NAS na server obsahující v těle sadu AVP popisující autenticitu uživatele nebo procesu a informace o službách a nastaveních, o které uživatel žádá.
   (b) RESPONSE - Odpověď serveru na REQUEST obsahuje v těle AVP, které povolí, zamítnou nebo upraví restrikce pro služby a nastavení, o která uživatel žádal.
3. Účtovací
   (a) REQUEST - Stejné jako REQUEST pro autorizaci s doplněnou sadou AVP určených pro statistická data.
   (b) RESPONSE - Odpověď serveru obsahující v těle stav informací poslaných v REQUEST paketu (např. že data byla uložena).

3.3.4 Autentifikace

Autentifikace vždy začíná START paketem odeslaným z NAS na server. Server reaguje odpovědí, tedy paketem REPLY. Zpráva v REPLY paketu může po NAS požadovat poslání dodatečné informace, oznámit chybu v autentifikaci nebo oznámit ukončení relace s výsledkem prošel/neprošel. Na každý REPLY paket požadující dodatečné informace odpovídá NAS CONTINUE paketem až do chvíle dokud jedna ze stran neukončí relaci[14]. Informace obsažené ve všech třech paketech jsou závislé na typu autentifikační metody, podrobněji rozepsané je najdeme v [2].

3.3.5 Autorizace

Autorizace probíhá výměnou jednoho páru paketů, požadavkem (REQUEST) a po něm nasledující odpovědí (RESPONSE). Informace obsažené v obou paketech mohou být na-

[14] NAS ukončí relaci zprávou v CONTINUE paketu, server zprávou v REPLY paketu.

staveny jako nutné nebo dobrovolné. Informace nastavené jako nutné musí být srozimitelné pro NAS i pro server a oba je musí umět použít.

3.3.6 Účtování

Účtovací relace probíhají stejně jako relace autorizace. Struktura REQUEST paketu a AVP je také stejná, jenom je doplněna o AVP pro statistické informace.

3.4 Kerberos

Kerberos protokol byl vyvíjen na MIT a jeho aktuální verze 5 byla zformalizovaná organizací IETF v podobě dokumentu RFC 4120. Kvůli zákazu exportu kryptografických algortimů vládou Spojených států amerických nemůže být implementace Kerberos protokolu MIT distribuována mimo území Spojených států. Volně šiřitelná verze je proto vyvíjena na Royal Institute of Technology ve Švédsku.

Poslední verze Kerberos protokolu, verze pět, vylepšuje svého předchůdce flexibilní specifikací paketu popsanou pomocí ASN.1, zavádí možnost přeposlání a distribuce tiketů, multi-realm autentifikaci, předautentifikaci a přidává podporu pro různé šifrovací metody. My se v následujícim textu budeme zabývat jenom touto poslední verzí Kerberos protokolu.

3.4.1 Vlastnosti

Kerberos protokol je vyvíjen s ohledem na maximální bezpečnost, proto se v paketech nikdy nepřenáší hesla po síti, ale používají se časově limitované kryptografické zprávy, tikety.

Architektura protokolu postavená na tiketech také umožňuje přístup ke službám bez potřeby dalších přihlášení. Po prvním přihlášení a ověření uživatele tak na základě získaného tiketu může uživatel používat všechny služby implementující Kerberos protokol po celou dobu platnosti tiketu.

Infrastruktura Kerberos protokolu je centralizovaná. Její centrum tvoří autentifikační server, kterému důvěřuje jak uživatel, tak i služby, které uživatel používá. Důvěřují tedy třetí straně.

Při identifikaci se neověřuje jenom uživatel nebo aplikace vůči Kerberos serveru, ale také server se kterým komunikují prokazuje svou identitu.

Kerberos protokol páté verze používá časové značky pro důvěryhodnější identifikaci. Je proto důležité, aby byl čas synchronizován na všech zařízeních Kerberos infrastruktury.

3.4.2 Kerberos terminolgie

klientem jsme doposud označovali zařízení AAA protokolů, které komunikovalo s AAA serverem za uživatele, avšak Kerberos protokol tyto dva pojmy nerozlišuje. My, pro účely této práce a zachováni jednotného označení, budeme i nadále striktně odlišovat pojem uživatel a klient.

single-singn-on neboli SSO je metoda přístupu uživatele, při které se ověří jenom na počátku SSO relace a po celou relaci pak může uživatel využívat služby bez potřeby dalšího ověření

tiket je časově limitovaná šifrovaná zpráva určená k ověření identity uživatele nebo služby. Tiket je generován KDC na základě identifikace uživatele. Můžeme si jej představit také jako řidičský průkaz, který nás jednoznačně identifikuje, má autoritou určenou dobu platnosti a opravňuje nebo omezuje nás k použití určitých zdrojů. Kerberos protokol pozná dva typy tiketů, TGT a ST. Popíšeme si je níže.

realm neboli identifikátor sítě (domény) společné pro uživatele a služby, které bude uživatel vyžívat. Pro účely Kerberos protokolu se standardně používá název domény přepsaný na všechna velká písmena. Kerberos v5 umožňuje i autentifikaci mezi různými realm-y (s různym realm identifikátorem), touto problematikou se ale nebudeme zabývat, vysvětlení najdeme v [4, 8]

principal je jednoznačný identifikátor uživatele nebo služby, pro kterou je vyžádán tiket. Je složen podle vztahu: $principal{=}jméno/instance\,@\,realm$[15]. Všechny entity infrastruktury Kerberos protokolu musí mít svůj jedinečný principal.

instance doplňuje uživatelské jméno nebo název služby a tím blíže specifikuje principal záznam. Pro uživatelský principal se může uvést skupina uživatele, např. „administrátoři". Pro principal služby se uvádí plné doménové jméno (FQDN) zařízení které službu poskytuje.

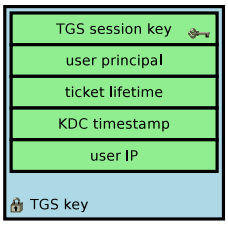
KDC (Key Distribution Center) je centrum infrastruktury Kerberos protokolu neboli Kerberos server. Ve výkladu budeme používat označení KDC. Skládá se ze tří logických částí, DB , AS a TGS. V jednom realm-u může být více KDC pro zajištění fail-over mechanizmu. Protože každý KDC obsahuje DB, je nutné zajistit jejich synchronizaci[16].

DB je databáze všech principal záznamů a šifrovacích klíčů k nim přiřazených.

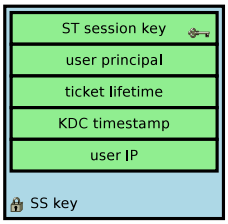
AS (Authentification Server), přiděluje uživateli TGT.

[15] Jméno je název služby, pro kterou je žádáno o tiket nebo uživatelské jméno. V případě žádosti o tiket pro vzdálené přihlášení na síťové zařízení se používá řetězec „host".

[16] Synchronizace není nijak standardizována a každá implementace Kerberos protokolu proto může využívat jinou metodu.



Obrázek 6: Struktura TGT Kerberos protokolu



Obrázek 7: Struktura ST Kerberos protokolu

TGS (Ticket Granting Server) přiděluje ST pro uživatele, který se ověří svým TGT.

TGT (Ticket Granting Ticket) je speciální tiket vydaný uživateli, který ho po vyžádání opravňuje získat ST. Jeho strukturu vidíme na obr. 6 , je složena z:

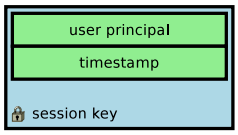
- TGS session key - klíče TGS relace vygenerované AS pro danou relaci uživatele
- user principal - principal uživatele
- ticket lifetime - čas do konce platnosti tiketu
- KDC timestamp - časová značka času vygenerování TGT dle času na KDC
- user IP - IP adresa uživatele

a zašifrována pomocí klíče TGS. Uživatel může mít na jednom počítači vždy pouze jeden TGT.

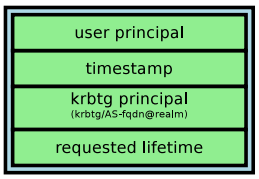
ST (Service Tiket) je tiket, kterým se uživatel ověřuje službě. Má strukturu podobnou jako TGT (obr. 7) . Liší se svou položkou ST session key, což je klíč ST relace vygenerovaný TGS pro danou relaci uživatele a specifickou aplikaci, ke které požaduje přístup. ST je uživateli vydán pro každou aplikaci, ke které požaduje přístup, a je zašifrován pomocí klíče SS.

SS (Service Server) neboli Kerberizovaný aplikační server je server poskytující kerberizovanou službu.

autentifikátor je řetězec dokazující autenticitu požadavku od uživatele (obr. 8) . Generuje se z:



Obrázek 8: Struktura autentifikátoru Kerberos protokolu



Obrázek 9: Požadavek AS_REQ Kerberos protokolu

- user principal - principal uživatele
- timestamp - časová značka vygenerování autentifikátoru

a je zašifrován pomocí klíče relace. Podle typu požadavku, ve kterém je uveden, to tedy může být klíč relace SS nebo TGS.

Kerberizovaná aplikace je aplikace, pomocí které uživatel přistupuje k SS[17].

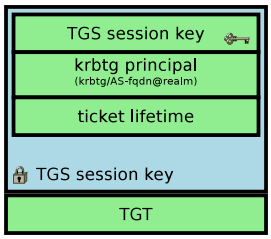
3.4.3 Autentifikace

Kerberos protokol je založen na principu Needham-Shroeder protokolu Garman [4], Neuman et al. [8] publikovaném v článku „Using Encryption for Authentication in large Networks of Computers" v roce 1978. Je založen na principu tiketů dokazujících jeho totožnost. Tikety získává od třetí strany, KDC, které důvěřují všechny entity Kerberos infrastruktury.

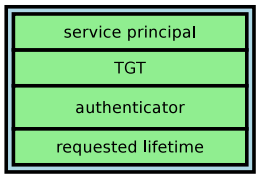
První požadavek od uživatele o autentifikaci, AS_REQ (obr. 9) , je poslán do AS v čistém textu. AS zkontroluje v DB, jestli daný uživatelský principal existuje a jestli je timestamp blízko času na AS serveru[18]. Pokud kterákoliv z kontrol selže, je uživateli poslána odpověď s chybovou hláškou. V opačném případě, tedy pokud ověření dopadlo úspěšně, AS vygeneruje klíč pro TGS relaci. Tento klíč použije dvakrát, jednou v TGT a po druhé v odpovědi AS_REP. TGT složí odpověď AS_REP (obr. 10) , zašifroje ji uživatelským heslem, přidá TGT a pošle ji uživateli. Uživatel své heslo zná, může tedy dešifrovat odpověd z KDC a tím získat klíč pro TGS relaci a samotný TGT s informací, kdy vyprší. Tyto informace mu umožní žádat o přístup k dalším službám bez potřeby znovu zadat heslo, proto si je uloží do své lokálni paměti. Uživatel nezná heslo TGS, nemůže tedy TGT dešifrovat a upravit.

Jelikož AS_REQ neni nijak šifrován, může být lehce podvržen útočníkem. Starší verze Kerberos protokolu posílaly odpověď AS_REP na jakýkoliv požadavek AS_REQ a tak mohl

[17] např. ssh je aplikace, kterou uživatel přistupuje ke službě vzdáleného síťového přístupu na server
[18] obvykle se toleruje odchylka do pěti minut.



Obrázek 10: Odpověď AS_REP Kerberos protokolu

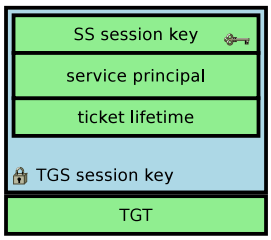


Obrázek 11: Požadavek TGS_REQ Kerberos protokolu

útočník pomocí bruteforce útoku odhalit uživatelovo heslo. Kerberos páté verze proto zavádí různé způsoby preautentifikace. Nejčastěji implementována je PA-ENC-TIMESTAMP založená na šifrované časové značce. AS, po obdržení požadavku AS_REQ, odešle oznámení o chybě doplněné žádostí o preautentifikaci. Jako reakci pošle uživatel dodatečně AS_REQ žádost doplněnou o časovou značku zašifrovanou svým heslem. AS dešifruje časovou značku a pokud je v tolerovaném posunutí času vůči svému lokálnímu času, pokračuje sestavením odpovědi AS_REP.

Pro přístup ke službě potřebuje uživatel získat ST. Sestaví proto požadavek TGS_REQ (obr. 11) . TGS dešifruje TGS_REQ požadavek použitím klíče pro TGS relaci, který má KDC uložen v DB, a porovná čas vygenerování požadavku se svým lokálním časem. Pokud nejsou od sebe posunuty více než je povoleno, pokračuje dešifrováním a kontrolou TGT pomocí svého klíče. Poté zkontroluje, jestli autentifikátoru a TGT neskončila platnost. Pokud některá kontrola skončí chybou, je požadavek zamítnut. V opačném případě vygeneruje SS, klíč pro SS relaci a sestaví odpověď TGS_REP (obr. 12) . TGS_REP je zašifrován pomocí klíče TGS relace a spolu s ST odeslán uživateli.

Uživatel pomocí klíče TGS relace, který má uložen v paměti, z odpověď dešifruje ST a klíč SS relace a uloží si je do paměti. Předloží je kerberizované aplikaci v čase využití služby. To, jak kerberizovaná aplikace a SS komunikují s KDC a uživatelem, aby ověřili



Obrázek 12: Požadavek TGS_REP Kerberos protokolu

ST, není protokolem definováno. Každá implementace Kerberos protokolu poskytuje svoje API, pomocí něhož může programátor SS a kerberizované aplikace ověřit ST. Výsledná implementace je však na jeho bedrech. Standardně uživatel pošle ST a nový autentifikátor zašifrovaný pomoci klíče SS relace pro daný ST. SS dešifruje ST a získá klíč SS relace. Může pak dešifrovat autentifikátor, navýší časovou značku z něj získanou o jedna a odešle ji zašifrovanou klíčem SS relace pro daný ST uživateli. Ten si může ověřit autenticitu SS a začne používat službu.

3.4.4 Autorizace a Účtováni

Kerberos protokol není AAA protokolem v plném rozsahu. Jeho specifikace implementuje jeno první „A", tedy autentifikaci. Pro potřeby autorizace a účtování má každá implementace Kerberos protokolu připravené programové rozhraní, kterým mohou vývojáři aplikaci tyto funkce implementovat. Nechává tedy poslední dvě „A" na aplikacích a službách, které uživatel používá.

3.4.5 Typy tiketů

Kerberos protokol páté verze přidává, oproti svým předchůdcům, nové typy tiketů

forwardable (přenositelný)
: Přenositelný tiket, TGT může být přenesen na jiné zařízení. Uživatel pak může vyžadovat ST nebo TGT z jiného zařízení.

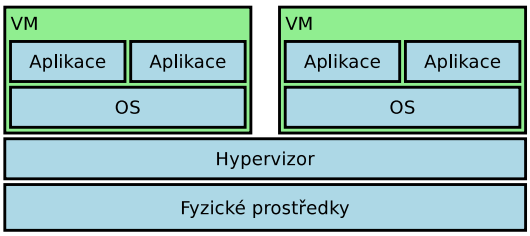
proxiable (přesměrovatelný)
: Přesměrovatelný tiket funguje podobně, jako přenositlený tiket, jenom nemůže být použit na vyžádání nového TGT.

renewable (obnovitelný)
: Uživatel dostane tiket s kratší životností, který může být obnovován maximálně n–krát. Po dosáhnutí maximálního počtu obnovení si musí uživatel vyžádat nový obnovitelný tiket. Obnovitelný tiket může být označen jako kompromitovaný, pak ho KDC odmítne obnovit.

postdated (časově posunutý)
: Tiket vydaný se začátkem platnosti v buducnosti. Může být použit například pro spuštění naplánovaných úloh.

4 Virtualizovaná univerzální AAA platforma

Virtualizace je všeobecně chápána jako abstrakce fyzických hranic technologií Wolf and Halter [10]. Jinými slovy se fyzické prostředky, jako např. procesor a operační paměť počítače, mohou rozdělit mezi více operačních systémů (dále jen OS) (obr. 13). Na jednom



Obrázek 13: Abstrakce fyzických prostředků ve virtualizaci

počítači tak můžeme provozovat spolu s hostitelským OS i další virtuální OS.

Každému virtuálnímu OS je vyčleněna část fyzických prostředků. Toto vyčlenění budeme nazývat virtuální stroj (dále jen VM). VM potřebujeme zapínat a vypínat, musíme určit kolik má mít paměti a jaký procesor, přidávat a odebírat síťové karty a jiné periferie, potřebujeme ho tedy nějak ovládat a spravovat. Pro tento účel slouží software zvaný hypervizor. Hypervizor běží jako aplikace na hostitelském OS, tedy OS který se startuje při zapnutí reálného počítače. Na hypervizoru závisí, jaký typ virtualizace je použit[19], jaké fyzické prostředky jsou potřeba a jaké OS mohou běžet ve VM.

4.1 Virtualizované prostředí

Pro naše potřeby univerzální platformy určené pro testování a výuku tedy musíme vybrat hypervizor, který může běžet na jakémkoli běžně používaném počítači[20] a bude podporovat virtuální síťové periferie potřebné pro plný přístup k síťovému rozhraní[21].

Podívejme se nyní na dnes nejčastěji používané hypervizory:

VMware vSphere je komerční plná virtualizace s velkou škálou funkcí, rozšíření a všeobecné podpory ze strany výrobců hardwaru, síťových zařízení a OS. VMware uvolňuje také verzi ESXi, která je dostupná pro nekomerční účely zdarma s omezenou sadou funkcí. U obou virtualizací je hostitelský OS zároveň hypervizorem a vyžaduje vlastní vyhrazené zařízení, tedy není možné je používat v ramci OS uživatele. Pro naše účely tedy není použitelný.
WMware také volně poskytuje aplikaci WMware player. Jedná se o hypervizor, který

[19] nejčastěji se virtualizace dělí z pohledu přístupu VM k hardware na plnou, částečnou a para virtualizaci
[20] Za běžné považujeme architektury x86 a x64 s operačními systémy Windows, MAC OS nebo Linux.
[21] Protokoly, keré budeme na virtuální platformě provozovat, používají různé typy paketů a je proto potřebné, aby tyto pakety nebyly filtrovány, tedy měly transparentní přístup k fyzické síti

umožňuje jenom ovládáni VM na hostitelském OS. VM se tedy musí připravit ve VMware vSphere a následně vyexportovat pro použití s WMware player. WMware player také neumožňuje úpravy nastavení VM a tedy přizpůsobení pro potřeby uživatele.

MS Hyper-V je také komerční plná virtualizace vydaná společností Microsoft. Hypervizor běží na hostitelském OS Windows Server[22].

XEN je open source hypervizor určený pro hostitelský OS typu Linux[23]. XEN hypervizor používá plnou, nebo para virtualizaci. Konfigurace a administrace XEN hypervizoru závisí na použité distribuci hostitelského OS a instalovaných nástrojích.

KVM je také open source hypervizor zabudovaný přímo do jádra OS Linux. KVM se řadí mezi plnou virtualizace.

VirtualBox hypervizor je od verze 4 uvolněn jako open source. Jedná sa o flexibilní hypervizor, který může běžet na různých hostitelských OS (Linux, Windows, MAC OS) a který podporuje jak plnou virtualizaci tak i virtualizaci emulovanou (pokud transparentní virtualizace není podporovaná hardwarem).

S ohledem na zamýšlené využívání virtualizace v naší virtuální AAA platformě, klademe na použitý hypervizor tyto požadavky:

- snadná instalace
- jednoduché ovládáni
- možnost dodatečné konfigurace
- použitelnost na počítačí studentů bez nutnosti zásahu do instalovaného OS
- podpora OS použitého pro naši platformu

Všechny požadavky splňuje pouze jeden hypervizor - VirtualBox.

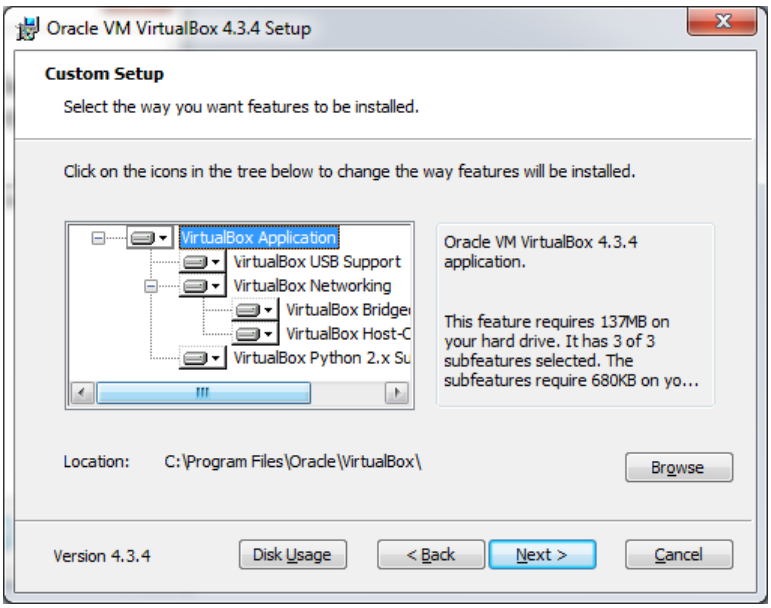
4.1.1 Instalace VirtualBox

VirtualBox stáhneme ze stánek `www.virtualbox.org` v sekci download. Instalaci zahájime spuštěním instalačního souboru (obr. 14). Instalátor nám v dalším kroku nabídne výběr instalovaných součástí (obr. 15), my vybereme všechny, pokud ještě nejsou vybrány. Po skončení instalace VirtualBox spustíme.

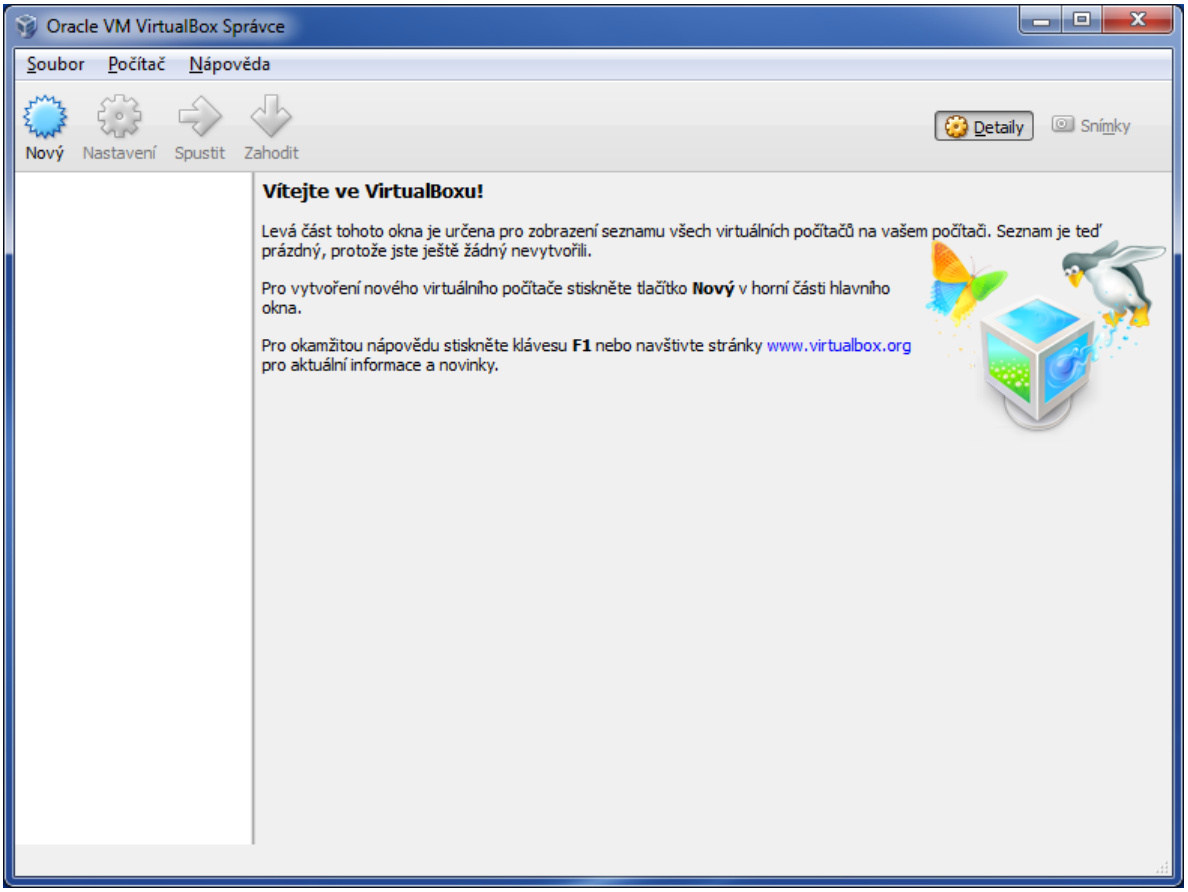
[22] Hyper-V se v okleštěné podobě nachází také ve Windows 8 jako WinXP mod.

[23] XEN hypervizor je použit také v Citrix XenServer, komerční verzi XEN serveru, která se řadí jako alternativa k VMware vSphere.

Obrázek 14: Instalační okno VirtualBox po spuštění instalace



Obrázek 15: Výber instalovaných součástí VirtualBox



Obrázek 16: VirtualBox manažér

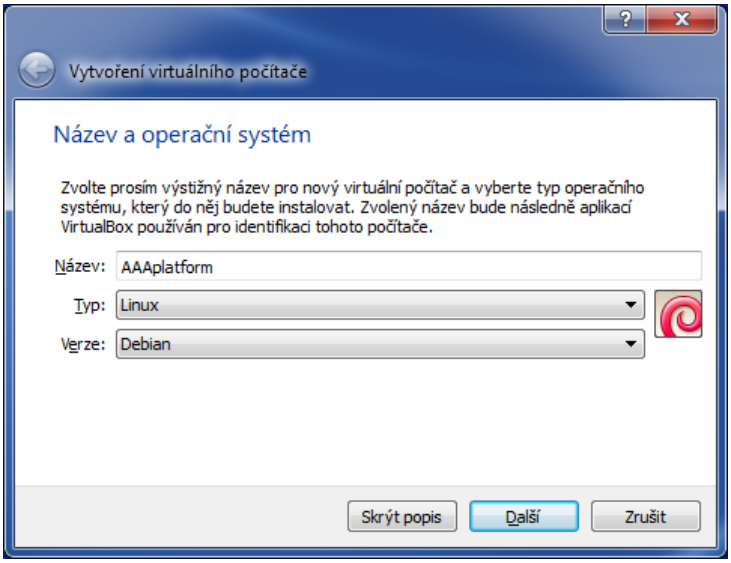
4.1.2 Konfigurace VM

Po instalaci a spuštění správce VirtualBoxu (obr. 16) si vytvoříme novou instanci VM kliknutím na tlačítko „Nový". Otevře se nám okno s průvodcem vytvoření VM. Postupně nastavíme jednotlivé parametry našeho VM:

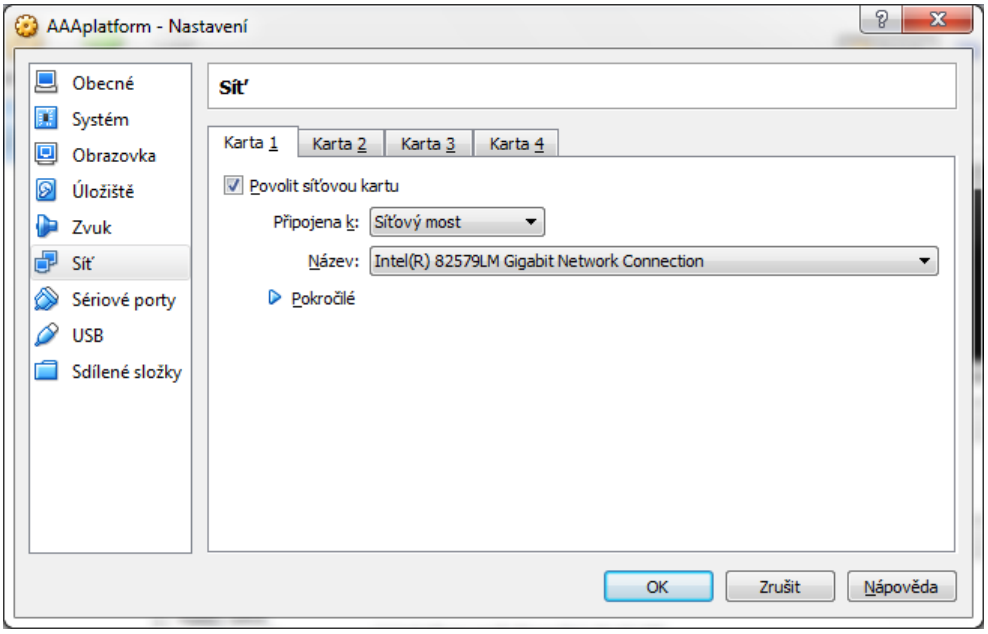
- typ Linux verze Debian (obr. 17)
- 1GB operační paměti
- 20GB dynamicky alokovaného diskového prostoru vytvořeného jako nový virtuální disk typu VMDK

Základni přípravu VM dokončíme pokročilým nastavením sítě a instalačního média[24]. Ve správci VirtualBoxu vybereme náš VM a klikneme na „Nastavení". V okně nastavení upravíme nastavení sítě dle obr. 18 a také virtuální CD médium v záložce „úložiště", kde vybereme obraz instalačního média OS Debian (obr. 19).

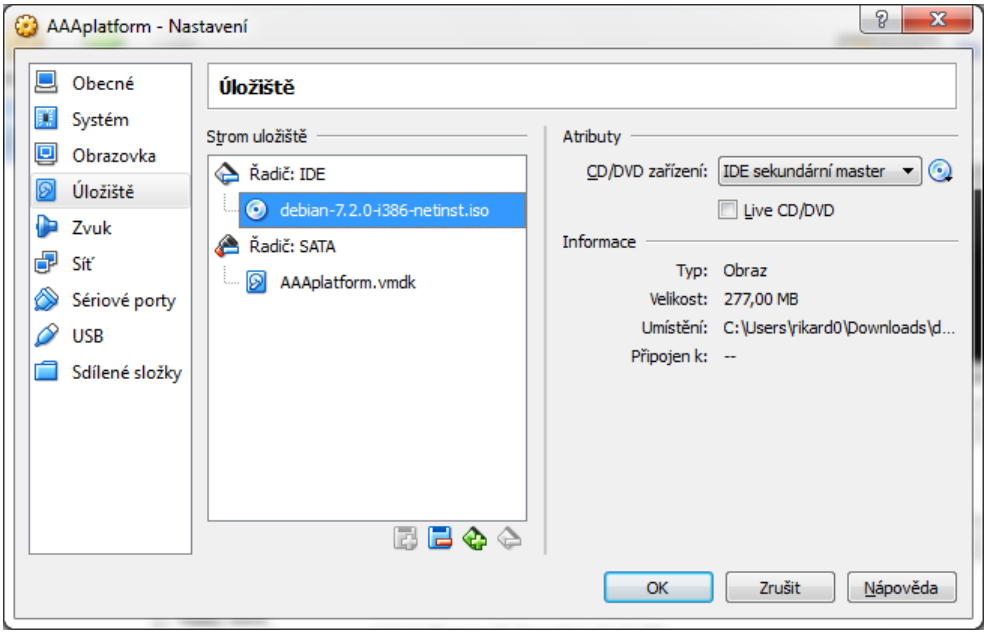
[24] instalační médium pro OS Debian stáhneme ze stránek `www.debian.org`. Pro naše účely je nejvhodnější malé instalační médium verze i386.



Obrázek 17: Konfigurace OS pro VM



Obrázek 18: Nastavení sítě pro AAAplatform VM



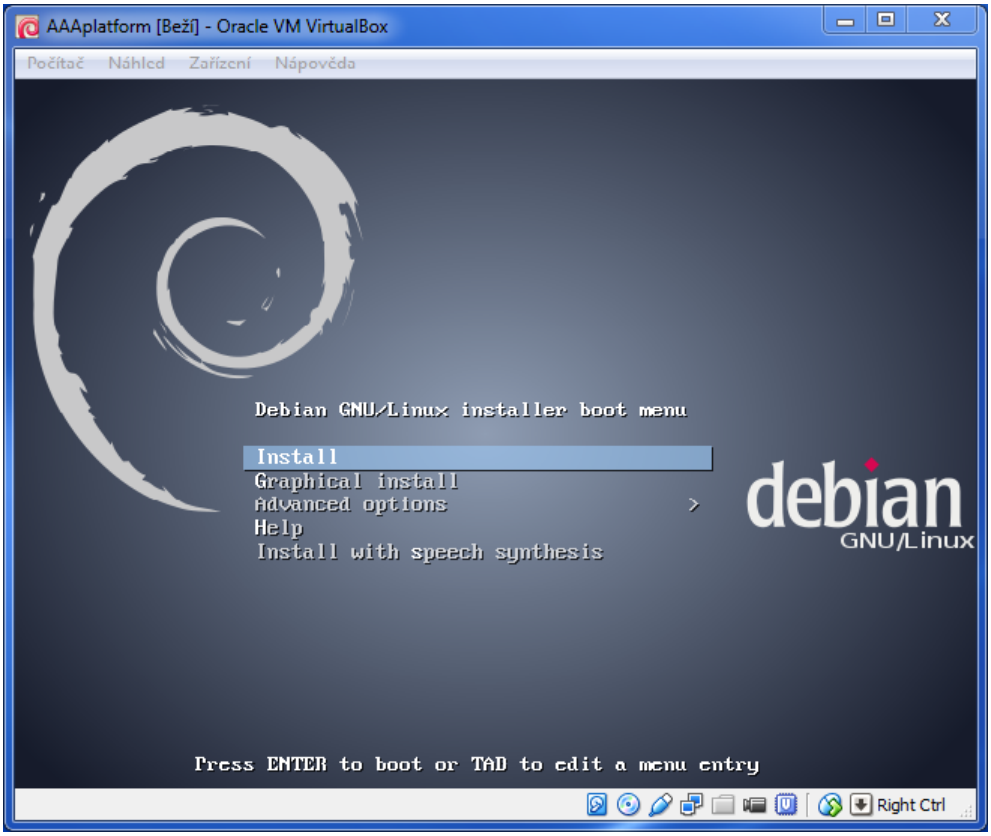
Obrázek 19: Nastavení instalačního média

4.1.3 OS pro VM

Naše virtuální platforma musí být legálně použitelná při výuce, a to bez licenčních poplatků. Z těchto důvodů si vybereme OS Linux, konkrétně distribuci Debian. Jedná sa o nejrozšířenější a nejčastěji používanou open source distribuci Linuxu instalovanou na servery. Její výhodou je predevším množství binárních balíčků instalovatelného software-u, dostupných také z FTP na serveru ČVUT.

Debian je vždy dostupný v několika větvích: unstable, testing, stable a old-stable. Unstable verze je čistě experimentální a vývojová a není doporučeno ji instalovat do produkčního prostředí, kde je vyžadováno stabilní chování. Unstable verze se po dosáhnutí určitého bodu ve vývoji zamrazí a tím vznikne nová testing větev. Vývoj a experimenty tak v unstable můžou dále pokračovat, kdežto software a nastavení systému se v testing větvi už jenom ladí na základě testů k maximální stabilitě. Když jsou všechny chyby opraveny, změní se testing větev na stable, čímž dojde k vydání nové verze Debianu a stará verze se změní na old-stable. Stable a old-stable větve jsou sice zmrazeny, tedy nové verze programů a jádra se do nich nepřidávají, ale chyby, které sa při používaní najdou, jsou opravovány. Rozdíl mezi nimi je nejenom ve verzích softwaru, ale taká v délce podpory, old-stable větev se udržuje jenom několik měsíců.

My použijeme aktuální stable větev zvanou wheeze uvolněnou v květnu 2013, což zaručuje podporu do roku 2015. Kvůli podpoře a stabilitě se také budeme snažit najít konkrétní software pro naše protokoly, který bude přímo dostupný v repozitářích Debianu nebo bude předpřipravený ve formě Debianích balíčků. Kompilací zdrojových kódů by se nám ztížila údržba systému.



Obrázek 20: Úvodni obrazovka instalátor spuštěného z instalačního média

Instalace Debianu se nám pustí automaticky po prvním zapnutí VM, kterou sme vytvořili v 4.1.2 (obr. 20). Instalací nás opět provázi instalační asistent, ve kterém postupně vybíráme nasledující možnosti:

- jazyk prostředí necháme v angličtině. Překlady do čestiny nejsou udělány pro všechno, tedy angličtine bychom se nevyhnuli a naši virtuálni platformu budou také používat anglicky hovořící studenti, kteři by měli problémy s případným přepnutím z čestiny do angličtiny
- vybereme Českou republiku jako krajinu lokace této instalace
- vzhledem na používaný anglický jazyk v instalaci vybereme locales vhodné pro tento jazyk, tedy en_us.UTF-8.
- rozložení klávesnice vybereme také anglické. V případě práce na konzoli nám to ulehčí práci s hledáním speciálních symbolů používaných ve skriptovacích jazycích a toto nastavení sa opět těžko přepíná
- konfigurace sítě proběhne automaticky nastavením všeho potrebného z DHCP serveru, nebo provedeme nastavení ručně zadánim IP adresy, masky sítě, atd. Ruční nastavení si můžeme zvolit libovolné, ale jako nastavení hostname (jména serveru)

použijeme „aaa.kme.fel.cvut.cz". Toto nastavení je důležité pro pozdější správnou funcki Kerberos serveru

- následuje nastavení uživatelů, kdy super uživateli root nastavíme heslo „pass" a vytvoříme neprivilegovaného uživatele „kme" s heslem „kme". Tato hesla jsou slabá a krátká, ale pro potřeby výukové platformy jsou postačující a hlavně lehko zapamatovatelná

- v části rozdělení disku necháme instalátor vytvořit diskový oddíl za nás přes celý virtuální disk a umístit všechny soubory na jediný diskový oddíl. Nastavení zkontrolujeme a potvrdíme jeho zápis do Master Boot Record (dále jen MBR)[25] části disku.

- po instalaci základní části systému se nás instalátor dotáže na výběr repozitáře pro doplňkový software. Vybereme repozitář umístěný na půdě školy a v dalším okně odebereme všechny rozširující části (obr. 21). Základní instalace systému je pro nás dostačujíci pro rozchození AAAplatformy.

- instalátor dokončí instalaci a restartuje VM. Po jeji restartu začne nabýhat náš čerstvě isntalovaný systém

Po prvním spuštění OS ve VM sa přihlásíme do systému a stáhneme si webové rozhraní pro naši platformu spolu s doplňkovými skripty pro snadnou instalaci dalších součástí.

```
apt-get update && apt-get install git
mkdir -p /srv/www
git clone https://github.com/kucharic/AAAplatform.git /srv/www/aaa
```

Po stáhnutí spustíme první skript, který nám upraví uživatelský shell rozhraní pro snadnější orientaci a práci na platformě.

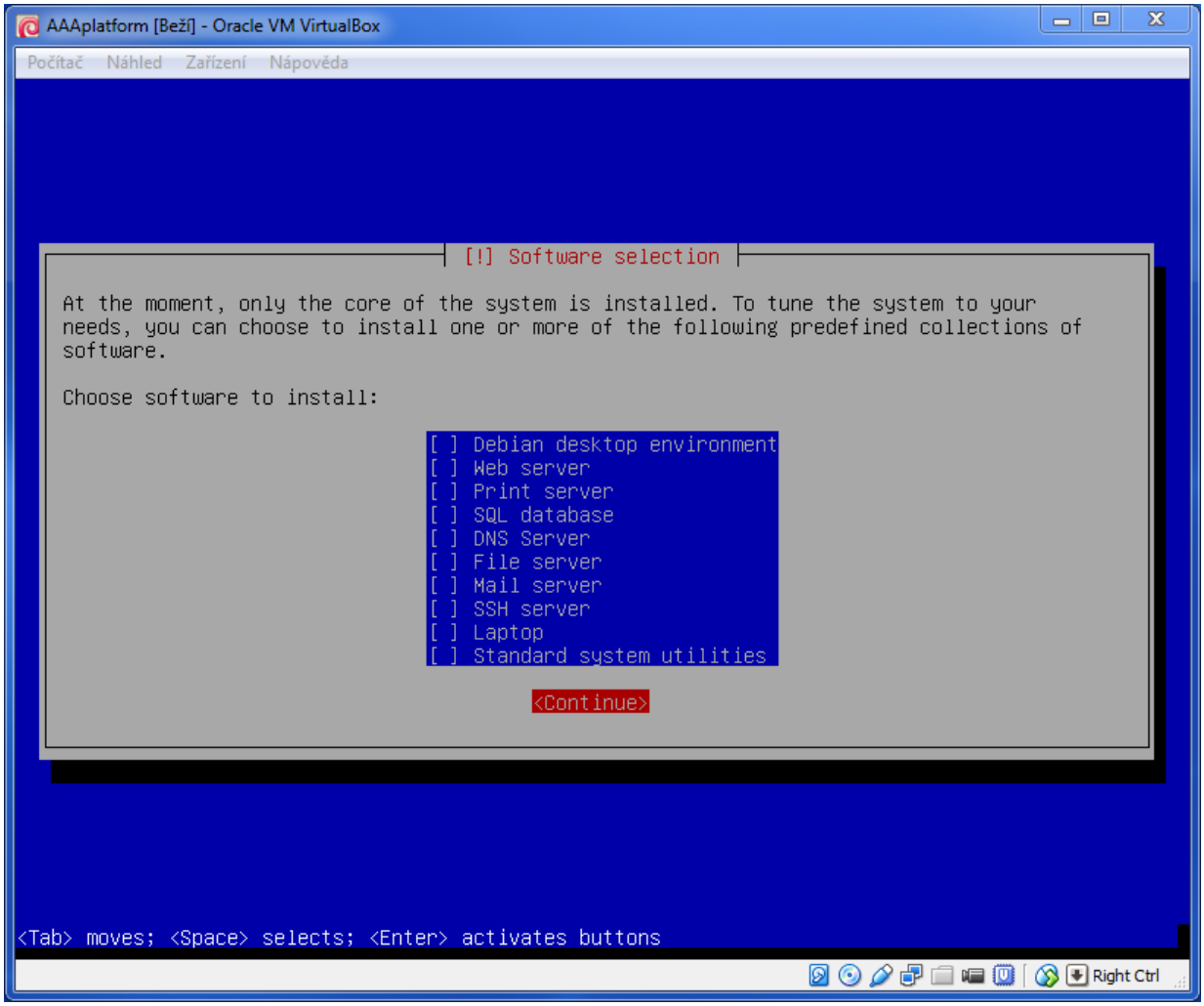
```
/srv/www/aaa/.install/workspace-configure.sh
```

4.2 RADIUS server software

Freeradius je open source software implementující protokol RADIUS na platformě Linux. Jedná se o implementaci, která je stále udržována a vyvíjena[26]. V repozitářích debianu najdeme také radiusd-livingston - první implementace radius protokolu a yardradius doplňující nové vlastnosti do radiusd-livingston. Vývoj radiusd-livingston a yardradius je již delší dobu pozastaven, proto je doporučován a také nasazován freeradius. V repozitáři můžeme dále nalézt hostapd, který také implementuje protokol RADIUS, ovšem je určen

[25] jedná se o prvních 512B na disku, ze kterých se čte zavaděč operačního systému, rozdělení disku na oddíly a identifikace disku

[26] za udržovaný open source software považujeme software, který nemá poslední úpravy v kódu starší než jeden rok



Obrázek 21: Nastavení výběru automaticky doinstalovaných součástí

pouze pro použití při vytváření softwarových WiFi AP, tedy není využitelný pro pevné síťe.

Freeradius nainstalujeme přikazem:

```
apt-get update && apt-get install freeradius freeradius-utils
```

Podíváme se nyní na možnosti použití různých databází udržujících nastavení uživatelů. K dispozici máme relační databáze MySQL a PostgreSql, dále také LDAP a iODBC komaptibilní databáze. Pro vývoj webového rozhraní se nejlépe používá relační databáze, která za řeši duplicitu dat a zjednodušuje tím výslední kód. Vebereme si proto nejrozšírenější relační databázi MySQL a doinstalujeme potřebné balíky

```
apt-get update && apt-get install freeradius-mysql
```

Konfiguraci Freeradius provedeme pomocí předpřipraveného skriptu:

```
/srv/www/aaa/.install/freeradius-configure.sh
```

4.3 Diameter server software

Pro Diameter protokol máme na výběr ze dvou open source implementací: OpenDiameter a FreeDiameter. Implementace OpenDiameter byla první, bohužel se již nevyvýjí. Zvolíme si proto FreeDiameter, kterého vývoj stále, ačkoliv pomalu, pokračuje. FreeDiameter je dělen na démona vycházejícího z Diameter protokolu (základní část která obsluhuje komunikaci se systémem, poslouchání na síti, příjem a odesílání Diameter zpráv) a na modul (konkrétní funkční celek) vycházející z Diameter aplikace. Jako modul jsou zatím implementovány aplikace Diameter SIP, Diameter EAP a překladač RADIUS/Diameter.

Freediameter se nenachází přímo v repozitáři debianu. Proto si před instalací přidáme repozitář vývojářů freediameter:

```
wget http://www.freediameter.net/packages/repository.key \
-O /tmp/freediameter_repo.key
apt-key add /tmp/freediameter_repo.key
cat >> /etc/apt/sources.list << EOF
deb http://www.freediameter.net/packages/debian wheezy contrib
deb-src http://www.freediameter.net/packages/debian wheezy contrib
EOF
```

Zjistíme ale, že závislosti balíku jsou nefunkční, proto budeme muset udělat menší úpravu systému nainstalováním starší verze knihovny, na které je balík závislý, aby nám instalace fungovala:

```
dpkg -i /srv/www/aaa/.install/libmysqlclient16_5.1.72-2_i386.deb
```

Nakonec nainstalujeme freediameter:

```
apt-get update && apt-get install freediameter-daemon \
freediameter-eap-server freediameter-common \
freediameter-accounting-server freediameter-dictionary-rfc4005 \
freediameter-dictionary-rfc4072 freediameter-dictionary-rfc4006 \
freediameter-dictionary-rfc4740 freediameter-dictionary-mip6 \
freediameter-dictionary-legacy freediameter-sip-server \
freediameter-radius-gateway
```

FreeDiameter ukládá accounting data do MySQL databáze. Naproti tomu FreeDiameter EAP modul, DiamEAP, vyžaduje pro svoji práci PostgreSQL databázi. Ve webovém rozhraní proto budeme implementovat přístupy do obou.

Konfiguraci FreeDiameter opět provedeme pomocí připraveného skritptu:

```
/srv/www/aaa/.install/freediameter-configure.sh
```

4.4 Kerberos server software

Pro Linux máme několik open source implementací, referenční od univerzity MIT, Heimdal od univerzity ve Švédsku a Shishi. My si vybereme verzi od MIT.

Kerberos pro svoji databázi klíčů používá buď to svoji vlastní strukturu nebo je možné použít také LDAP. My použijeme nativní strukturu a budeme ji ovládat pomocí nativních příkazů spouštěných přímo z naší webové aplikace. Umožní nám to v budoucnu jednodužší přechod na nové verze.

Kerberos pro svoji správnou funkčnost vyžaduje dobře nakonfigurovaný DNS server s reverzními záznamy pro každé zařízení v naši síti. Pro laboratorní účely však použijeme statické nastavení „hosts" souboru na každém zařízení. Pro instalaci a konfiguraci serveru použijeme připravený skript, který patřičné úpravy udělá za nás:

```
/srv/www/aaa/.install/kerberos-configure.sh
```

4.5 Tacacs+ server software

Existují dvě implementace Tacacs+ protokolu vydané jako open source. Jedna je vyvíjena Markem Huberem a je dostupná na stránkách `http://www.pro-bono-publico.de/projects/tac_plus.html` , druhá, TacPlus je vývíjena společnosti SHRUBBERY networks, Inc. My si zvolím tu druhou nejenom proto, že je dostupná přímo v repozitáři, ale také proto, že je široce nasazená a je pro ni dostupná spousta dokumentace pro konkrétní výrobce síťového hardware.

Nainstalujeme ji pomocí přikazu:

```
apt-get update && apt-get install tacacs+
```

TacPlus implementace nepodporuje žádný databazový backend, proto jí s naši webovou aplikaci propojíme oklikou. TacPlus umožňuje pro autentifikaci použít systémové uživatele a ty v linuxu můžeme pomocí pam_mysql modulu propojit s MySQL databázi. Pro autorizaci zase TacPlus umožňuje použití externích programů nebo skriptů. Pomocí projektu do_auth od autora Jathana McColluma tak získame přístup k jednoduše generovatelnému konfiguračnímu souboru[27]. Oběma oklikami získáme vyšší volnost bez potřeby restartování TacPlus serveru při každé změně nastavení práv a uživatelů.

A konfiguraci provedeme pomocí skriptu:

```
/srv/www/aaa/.install/tacplus-configure.sh
```

4.6 Webové rozhrani pro administraci

Webové rozhraní pro naši platformu budeme programovat v jazyku PHP verze 5.4, který propojime s webserverem Apache. Doinstalujeme potřebné balíky

```
apt-get update && apt-get install apache2 apache2-mpm-prefork \
apache2-utils libapache2-mod-php5 php-pear php5-cli php5-mysql \
php5-pgsql php5-dev
```

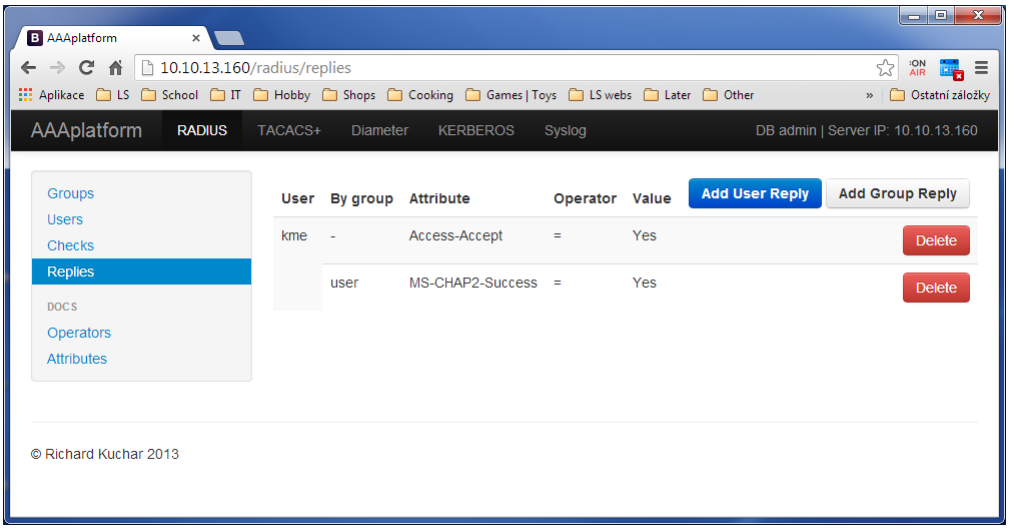
a nastavíme webserver předpřipraveným skriptem

```
/srv/www/aaa/.install/apache-configure.sh
```

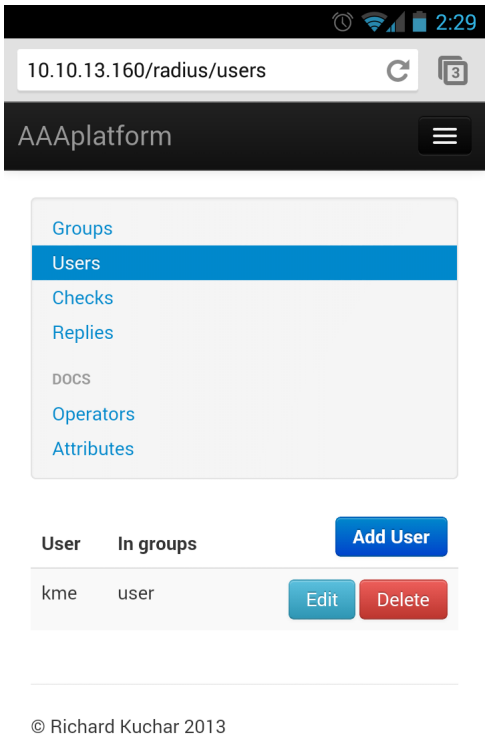
Vzhled webového rozhraní je vytvořen pomocí projektu Bootstrap verze 2.3.3. Jedná se o sadu CSS šablon a předpřipravených vzorů HTML kódu pro různé objekty. Díky tomu se nebudeme muset věnovat grafické stránce našeho webového rozhraní (obr. 22). Získame také odladěné a funkční rozhraní pro mobilní zařízení, jakými jsou tablety (obr. 23) nebo mobilní telefony (obr. 24).

Rozhraní rozdělíme do hlavních sekcí podle aplikace, kterou daná sekce bude ovládat, a pro každou sekci uděláme vlastní menu dle možností dané aplikace. Rozdělení doplníme o webový nástroj pro přímou administraci používaných databází a rozhraní pro konfiguraci sítě. Výsledná struktura aplikace bude vypadat následovně:

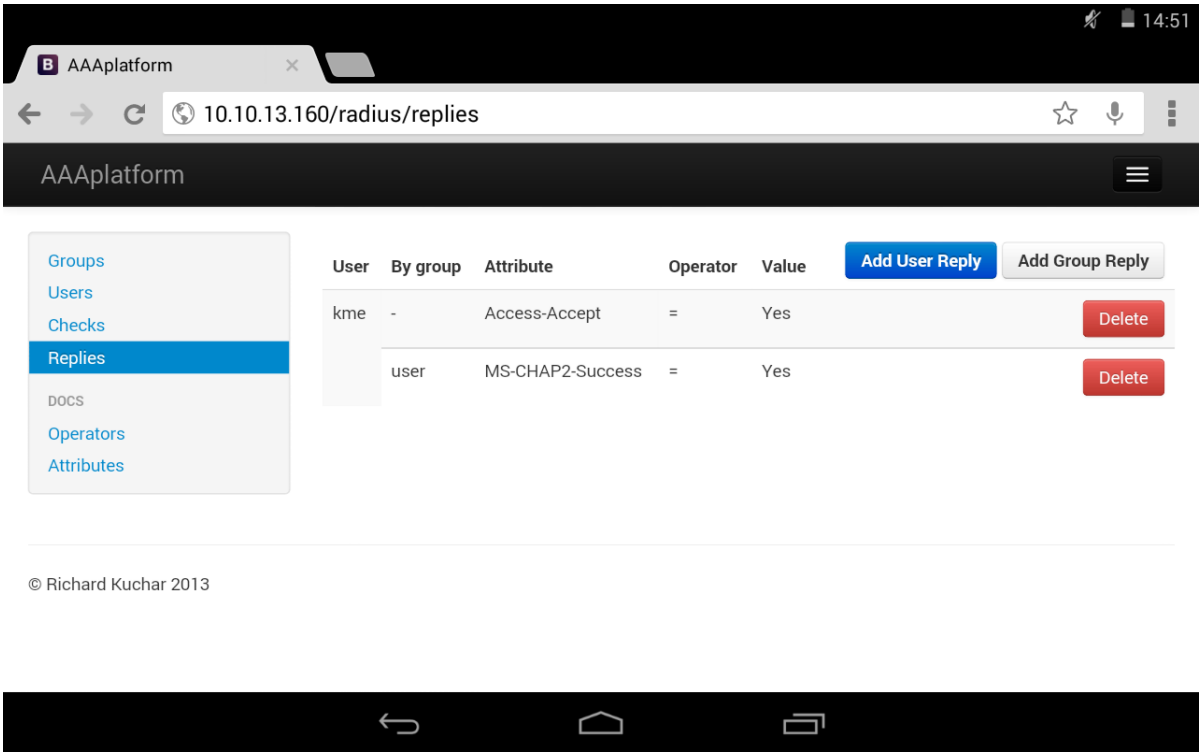
[27] v porovnání s konfiguračním souborem TacPlus, který se primárně používá jako zdroj nastavení pro autorizaci.



Obrázek 22: Webové rozhraní administrace AAAplatformy na PC



Obrázek 23: Webové rozhraní administrace AAAplatformy na chytrém mobilu



Obrázek 24: Webové rozhraní administrace AAAplatformy na tabletu

- AAAplatform:
  - RADIUS
    - Groups
      - Add Group
      - Delete Group
    - Users
      - Add User
      - Delete User
      - Edit User
    - Checks
      - Add Rule by User
      - Add Rule by Group
      - Delete Rule
    - Replies
      - Add Rule by User
      - Add Rule by Group
      - Delete Rule
  - TACACS+
    - Groups
      - Add Group
      - Delete Group
    - Users
      - Add User
      - Delete User
      - Edit User
    - Authorizations
      - Add Rule by Group
      - Delete Rule
  - Diameter
    - Groups
      - Add Group
      - Delete Group
    - Users
      - Add User
      - Delete User
      - Edit User
    - Authorizations
      - Add Rule by Group
      - Delete Rule
    - Authentications
      - Add Rule by Group
      - Delete Rule
  - KERBEROS
    - Policices
    - Principals
      - Add Principal
      - Delete Principal
      - Edit Principal
  - Syslog
  - DB admin
  - Server IP:

5 Ověření funkčnosti

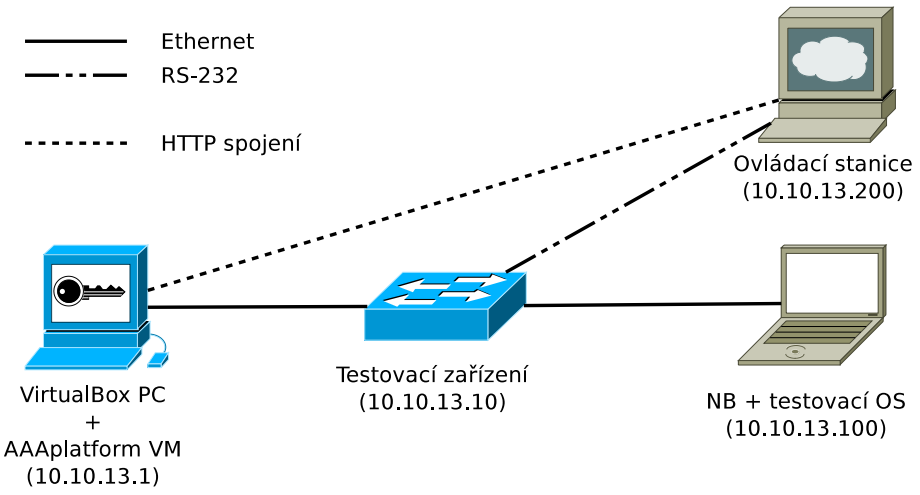
Vytvořenou virtuální platformu otestujeme vůči několika zařízením od různých výrobců, čímž nejen ověříme její funkčnost, ale také připravíme základní návod, jak s platformou pracovat. Testovat budeme podporu protokoloů pro správu uživatelů a kontrolu přístupu k síti (standard IEEE 802.1x[28]) na zařízeních výrobců Cisco Systems, Inc. (dále jen Cisco), Juniper Networks, Inc. (dále jen Juniper) a MikroTik. Konkrétní typy zařízení a verze jejich softwarového vybavení shrnuje tabulka 1.

Výrobce	Typ	Verze SW
Cisco Systems, Inc.	ISR 871	IOS 12.3(8r)YI3
	Catalyst 2950	IOS 12.1(22)EA11
Juniper Networks, Inc.	EX4200-48T	Junos 12.3R2.5
	SRX210	Junos 11.4R9.4
MikroTik	RB751G-2HnD	RouterOS v5.16

Tabulka 1: Zařízení použitá pro testování vrtuální AAA platformy a jejich softwarové vybavení

Ačkoliv jsou zařízení od společnosti Cisco různého typu a s různou verzi softwaru, u obou je podpora testovaných AAA funkcí stejná. Obdobně jsou na tom zařízení společnosti Juniper.

Zařízení budou spolu s virtuální platformou propojena v testovací topologii (obr. 25). Pro



Obrázek 25: Topologie zapojení testovacího prostředí

testovaná zařízení nastavíme základní konfiguraci dle přílohy A.1 pro zařízení společnosti Cisco, B.1 pre zařízení společnosti Juniper a C.1 pro zařízení společnosti MikroTik.

Testovat budeme jenom protokoly RADIUS a TACACS+, protože protokoly Diameter a Kerberos nejsou podporovány ani jedním z testovaných zařízení. Dle aplikace „Cisco

[28]pro zjednodušení budeme používat protokol EAP-MD5 nakonfigurovaný na testovacím notebooku

	RADIUS	Diameter	TACACS+	Kerberos
Cisco	systém + 802.1x	-	systém	systém
Juniper	systém + 802.1x	-	systém	-
MikroTik	systém + 802.1x	-	-	-

Tabulka 2: Přehled podpory AAA protokolů výrobci. Označení „systém" je použito pro správu uživatelů systému, označení 802.1x je použito pro kontrolu přístupu k síti.

Feature Navigator" [3] společnosti Cisco je Kerberos protokol podporován na novějších modelech zařízení této značky. Protokol Diameter není podporován ani jedním z testovaných výrobců. Výsledné srovnání podpory protokolů ukazuje tabulka 2.

5.1 RADIUS

Pro otestováni RADIUS protokolu budeme potřěbovat založit testovacího uživatele. Založíme proto uživatele „kme" a nastavíme pro něj kontrolní pravidlo „Cleartext-Password := kme" a odpověď „Reply-Message = Hello, %u".

Pro otestováni správy uživatelů se zařízeními společnosti Cisco použijeme konfiguraci v příloze A.2 a pro test kontroly přístupu k síti konfiguraci z přílohy A.4 a připojený testovací notebook.

Obdobným spůsobem budeme testovat také zařízení společnosti Juniper, jen použijeme konfigurace z přílohy B.2 pro správu užívatelů a přílohy B.4 pro kontrolu přístupu k síti protokolem 802.1x. Pro správu uživatelů musíme také doplnit odpověď RADIUS serverem: „User-Name = remote", protože software zařízení vyžadaje definování uživatele a jeho nastavení. Touto odpovědí získáme možnost používat generického užívatele „remote" a tudíž nemusíme definovat uživatele samotné.

Zařízení společnosti MikroTik podporuje, kromě správy uživatelů, protokol RADIUS jenom pro kontrolu přístupu k bezdrátové síti. Testovací topologie bude proto poupravena, k zařízení se testovací notebook bude připojovat prostřednictvím bezdrátové sítě místo drátového propojení ethernetem. Pro konfiguraci zařízení použijeme příkazovou řádku do které vložíme konfiguraci z přílohy C.3 pro kontrolu přístupu k síti a konfiguraci z přílohy C.2 pro správu uživatelů.

5.2 TACACS+

I pro TACACS+ protokol si vytvoříme uživatele „kme" s heslem „kme". TACACS+ protokol nepodporuje standard 802.1x, proto bude testovat jenom správu uživatelů testovacích zařízení.

K testům použijeme přílohu A.3 pro zařízení společnosti Cisco a B.3 pro zařízení společnosti Juniper. Zařízení společnosti MikroTik nepodporuje TACACS+ protokol.

Pro správu uživatelů zařízení Juniper je potřeba, podobně jako pro RADIUS protokol, použít generického užívatele v konfiguraci zařízení. Avšak v tomto případě je potřebný zásah do konfigurace TACACS+ serveru v naší virtuální platformě, protože není možné toto nastavení ovládat pomocí databáze a vyžaduje restart TACACS+ serveru. Dostupnou proto budeme mít jenom jednou generickou třídu uživatele spravovaného TACACS+ serverem naší virtuální platformy.

6 Závěr

Ve své diplomové práci jsem se zabýval přípravou virtuální AAA platformy vhodné k výuce AAA protokolů. Práce je členěna do čtyř hlavních částí, kde v 2) jse věnuji základnímu popisu AAA standardu (RFC 2903), v 3) popisuji protokoly rodiny AAA (RADIUS, Diameter, TACACS+, Kerberos), dále pak v 4) využívám získané teoretické znalosti k přípravě virtuální platformy a jejího rozhraní, kterou následně v 5) testuji vůči několika různým zařízením.

Zjistil jsem, že nejlepší podporu u výrobců má protokol RADIUS, který oproti druhému nejčastěji podporovanému protokolu TACACS+ těží predevším z kompatibility se standardem 802.1x. Naopak se mi nepodařilo ověřit funkčnost protokolu Diameter, který nebyl podporován ani jedním z použitých zařízení, ačkoliv byl Diameter protokol definován již v roce 2003 normou RFC 3588 ani funkčnost protokolu Kerberos, který je dle informací výrobce podporován teprve na novějších zařízeních než těch použitých v testu Výsledné shrnutí podpory AAA protokolů testovanými zařízeními popisuji v tabulce 2.

Reference

[1] M Beadles and D. Mitton. Criteria for Evaluating Network Access Server Protocols. RFC 3169, September 2001. URL `http://tools.ietf.org/html/rfc3169`.

[2] D. Carrel and L. Grant. The TACACS+ Protocol Version 1.78. draft-grant-tacacs-02, January 1997. URL `http://tools.ietf.org/html/draft-grant-tacacs-02`.

[3] Inc. Cisco Systems. Cisco feature navigator, 2013. URL `http://tools.cisco.com/ITDIT/CFN/`.

[4] Jason Garman. *Kerberos: The Definitive Guide*. Definitive Guide Series. O'Reilly Media, 2003. ISBN 9780596004033. URL `http://books.google.cz/books?id=dGMd-uay-lkC`.

[5] Jonahan Hassell. *RADIUS: Securing Public Access to Private Resources*. Oreilly Series. O'Reilly Media, 2002. ISBN 9780596003227. URL `http://books.google.cz/books?id=o5xQNbuvJ7QC`.

[6] C. Laat, G. Gross, L. Gommans, and J. Vollbrecht. Generic AAA Architecture. RFC 2903, August 2000. URL `http://tools.ietf.org/html/rfc2903`.

[7] Madjid Nakhjiri and Mahsa Nakhjiri. *AAA and Network Security for Mobile Access: Radius, Diameter, EAP, PKI and IP Mobility*. John Wiley & Sons, 2005. ISBN 9780470011942. URL `http://books.google.cz/books?id=cfRSAAAAMAAJ`.

[8] C. Neuman, T. Yu, S. Hartman, and D. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120, July 2005. URL `http://tools.ietf.org/html/rfc4120`.

[9] C. Rigney, A. Rubens, W. Simpson, and S. Willens. Remote Authentication Dial In User Service (RADIUS). RFC 2865, June 2000. URL `http://tools.ietf.org/html/rfc2865`.

[10] Chris Wolf and M. Halter, Eric. *Virtualization: From the Desktop to the Enterprise*. Expert's voice in networking. Apress, 2005. ISBN 9781430200277. URL `http://books.google.cz/books?id=qXw9p1nzb9QC`.

A Testovací konfigurace pro zařízení Cisco ISR 871 a Catalyst 2950

A.1 Základní konfigurace

```
!
interface FastEthernet0
 description Interface connected to AAA
 switchport access vlan 1
!
interface FastEthernet1
 switchport access vlan 1
!
interface Vlan1
 ip address 10.10.13.10 255.255.255.0
!
```

A.2 Správa uživatelů protokolem RADIUS

```
!
aaa new-model
!
aaa authentication login default group radius local
aaa authorization exec default group radius local
!
radius-server host 10.10.13.1 auth-port 1812 acct-port 1813 key pass
!
```

A.3 Správa uživatelů protokolem TACACS+

```
!
aaa new-model
!
aaa authentication login default group tacacs+ local
aaa authorization exec default group tacacs+ local
!
tacacs-server host 10.10.13.1 key pass
!
```

A.4 Kontrola přístupu k síti protokolem RADIUS

```
!
aaa new-model
!
aaa authentication dot1x default group radius
!
dot1x system-auth-control
!
radius-server host 10.10.13.1 auth-port 1812 acct-port 1813 key pass
!
interface FastEthernet1
 description Interface connected to testing NB
 dot1x pae authenticator
 dot1x port-control auto
 dot1x guest-vlan 10
!
```

B Testovací konfigurace pro zařízení Juniper EX4200-48T a SRX210

B.1 Základní konfigurace

```
set interfaces ge-0/0/0 description aaa
set interfaces ge-0/0/0 family ethernet-switching port-mode access
set interfaces ge-0/0/1 description testing NB
set interfaces ge-0/0/1 family ethernet-switching port-mode access
set interfaces vlan unit 1 family inet address 10.10.13.10/24
set vlans aaa vlan-id 1
set vlans aaa interface ge-0/0/0.0
set vlans aaa interface ge-0/0/1.0
set vlans aaa l3-interface vlan.1
```

B.2 Správa uživatelů protokolem RADIUS

```
set system authentication-order [ radius password ]
set system radius-server 10.10.13.1 secret "$9$3IfHnA0B1hrK8Rh"
set system radius-options password-protocol mschap-v2
set system login user remote full-name "All remote users" uid 9999
    class operator
```

B.3 Správa uživatelů protokolem TACACS+

```
set system authentication-order [ tacplus password ]
set system tacplus-server 10.10.13.1 secret "$9$3IfHnA0B1hrK8Rh"
set system login user remote full-name "All remote users" uid 9999
    class operator
```

B.4 Kontrola přístupu k síti protokolem RADIUS

```
set protocols dot1x authenticator authentication-profile-name aaa
set protocols dot1x authenticator interface ge-0/0/1.0 supplicant
    single-secure
set access radius-server 10.10.13.1 secret "$9$Hkfz3nCuBE/C"
set access profile aaa authentication-order radius
set access profile aaa radius authentication-server 10.10.13.1
```

C Testovací konfigurace pro zařízení MikroTik RB751G-2HnD

C.1 Základní konfigurace

```
/interface bridge
add admin-mac=D4:CA:6D:67:CD:09 auto-mac=no l2mtu=1598\
 name="BR - LAN" protocol-mode=stp
/interface ethernet
set 0 name="aaa"
set 1 arp=disabled name="E2 - LAN"
set 2 arp=disabled master-port="E2 - LAN" name="E3 - LAN"
set 3 arp=disabled master-port="E2 - LAN" name="E4 - LAN"
set 4 arp=disabled master-port="E2 - LAN" name="E5 - LAN"
/interface bridge port
add bridge="BR - LAN" interface="E2 - LAN"
add bridge="BR - LAN" interface=WiFi
/ip address
add address=10.10.13.10/24 comment="" interface="BR - LAN"
/ip service
set ssh address=10.10.13.0/24
/system leds set 0 interface=WiFi
```

C.2 Správa uživatelů protokolem RADIUS

```
/radius
add address=10.10.13.1 secret=pass service=login
/user aaa
set use-radius=yes
```

C.3 Kontrola přístupu k síti protokolem RADIUS

```
/interface wireless security-profiles
add authentication-types=wpa-eap,wpa2-eap eap-methods=eap-tls\
 group-ciphers=tkip,aes-ccm management-protection=allowed\
 mode=dynamic-keys name=aaa supplicant-identity=""\
 tls-mode=dont-verify-certificate unicast-ciphers=tkip,aes-ccm
/interface wireless
set 0 antenna-gain=8 band=2ghz-onlyg\
 basic-rates-a/g=6Mbps,9Mbps,12Mbps,18Mbps,24Mbps,36Mbps,48Mbps,54Mbps\
 basic-rates-b="" bridge-mode=disabled country="czech republic"\
 disabled=no distance=indoors frequency=2442 ht-rxchains=0,1\
 ht-txchains=0,1 l2mtu=2290 mode=ap-bridge name=WiFi\
 rate-selection=legacy rate-set=configured security-profile=aaa\
 ssid=aaa supported-rates-b="" tx-power-mode=all-rates-fixed\
 wireless-protocol=802.11
/radius
add address=10.10.13.1 secret=pass service=wireless
```

D Adresářová struktura přiloženého DVD

```
DVD/
  kuchar_richard.pdf
  AAAplatform_i386.ova
  AAAplatform_x64.ova
  AAAplatform_webinterface.zip
  prilohy/
    A1_konfigurace.txt
    A2_konfigurace.txt
    A3_konfigurace.txt
    A4_konfigurace.txt
    B1_konfigurace.txt
    B2_konfigurace.txt
    B3_konfigurace.txt
    B4_konfigurace.txt
    C1_konfigurace.txt
    C2_konfigurace.txt
    C3_konfigurace.txt
```