UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matej Kocmur

Primerjava metod in orodij za avtentikacijo

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentorica: doc. dr. Mojca Ciglarič

Ljubljana, 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00211/2012
Datum: 02.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATEJ KOCMUR**

Naslov: **PRIMERJAVA METOD IN ORODIJ ZA AVTENTIKACIJO**
COMPARISON OF AUTHENTICATION METHODS AND TOOLS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Podrobno preučite področje avtentikacije v porazdeljenih sistemih. Opišite danes najpomembnejše avtentikacijske protokole in komentirajte varnostni nivo, ki ga zagotavljajo. Pojasnite delovanje Radiusa, LDAPa in Kerberosa ter pojasnite podobnosti in razlike v uporabljenih konceptih. Preglejte obstoječe implementacije avtentikacijskih strežnikov in izberite tri najzanimivejše, te potem preizkusite v praksi in jih primerjajte med seboj. V zaključku kritično poglejte na možnosti in primernost uporabe vsakega od njih za specifične sisteme.

Mentor:

M. Ciglarič

doc. dr. Mojca Ciglarič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Matej Kocmur,

z vpisno številko 63080211,

sem avtor diplomskega dela z naslovom:

Primerjava metod in orodij za avtentikacijo

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom

  doc. dr. Mojce Ciglarič,

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 24. 9. 2012 Podpis avtorja:

Zahvaljujem se raziskovalcu asistentu Mihi Groharju za nasvete in pomoč pri praktičnem delu diplomskega dela. Prav tako se zahvaljujem svoji mentorici doc. dr. Mojci Ciglarič za usmerjanje, nasvete in spodbudo pri izdelavi diplomskega dela.

Kazalo vsebine

Kazalo slik

Kazalo tabel

Seznam uporabljenih kratic

PIN – *Personal Identification Number* (uporabniško geslo)
AAA – *Authentication, Authorization and Accouting* (varna arhitektura, ki vključuje avtentikacijo, avtorizacijo in beleženje)
RADIUS – *Remote Authentication Dial In User Service* (protokol, ki vključuje proces avtentikacije, avtorizacije in beleženja)
LDAP – *Lightweight Directory Access Protocol* (protokol za dostop in upravljanje z imeniškimi storitvami)
PPP – *Point to Point Protocol* (protokol, ki se uporablja za vzpostavitev neposredne povezave med dvema omrežnima vozliščema)
PAP – *Password Authentication Protocol* (avtentikacijski protokol, ki za proces avtentikacije uporablja geslo)
HTTP – *Hypertext Transfer Protocol* (aplikacijski protokol za pošiljanje sporočil na svetovnem spletu)
CHAP – *Challenge Handshake Authentication Protocol* (avtentikacijski protokol, ki temelji na izzivu)
EAP – *Extensible Authentication Protocol* (avtentikacijski protokol oziroma okvir, ki nudi številne metode)
MD5 – *Message Digest Algorithm* (zgoščevalna funkcija, ki danes ni več varna za uporabo)
UDP – *User Datagram Protocol* (primer protokola na transportni plasti, ki ni zanesljiv pri prenosu podatkov)
AVPs – *Attribute Value Pairs* (prilastki, ki vsebujejo konkretne podatke za prenos, pri protokolu RADIUS in DIAMETER)
VSAs – *Vendor Specific Attributes* (prilastki, ki predstavljajo razširitev obstoječih pri protokolu RADIUS in DIAMETER)
IP – *Internet Protocol* (enolično določa posamezno napravo v omrežju)
TLV – *Type Length Value* (format, ki ga uporabljajo prilastki protokola RADIUS in DIAMETER)
TCP - *Transmission Control Protocol* (protokol na transporni plasti, ki nudi zanesljiv prenos podatkov)
LDIF – *LDAP Data Interchange Format* (format datotek, ki ga uporablja imeniška storitev LDAP)
SSL/TLS – *Secure Socket Layer/Transport Layer Security* (protokol, ki omogoča varno komunikacijo na medmrežju)
DES – *Data Encryption Standard* (simetrični algoritem za šifriranje sporočil, ki danes ni več varen)

AES – *Advanced Enryption Standard* (varni simetrični algoritem za šifriranje sporočil)
SHA – *Secure Hash Algorithm* (zgoščevalna funkcija, ki izhaja v več inačicah)
MIT – *Massachusetts Institute of Techonology* (inštitut, na katerem je bil razvit protokol KERBEROS)
MAC – *Message Authentication Code* (algoritem, s katerim sta zagotovljeni integriteta in avtentičnost spročila)
HMAC – *Hash based Message Authentication Code* (primer algoritma MAC, ki vključuje zgoščevalno funkcijo)
PKI – *Public Key Infrastructure* (varna infrastruktura, ki temelji na asimetrični kriptografiji)
CA – *Certificate Authority* (certifikacijska agencija, ki je del infrastrrukture PKI)
RA – *Registration Authority* (registracijska agencija, ki je del infrastrukture PKI)
SASL – *Simple Authentication and Secure Layer* (mehanizem, ki ga nudi protokol LDAP inačice tri)
RFC – *Request For Comments* (serije standardov)
PAM – *Pluggable Authentication Module* (mehanizem, ki aplikacijam omogoča avtentikacijo z uporabo različnih avtentikacijskih mehanizmov)
NTP – *Network Time Protocol* (protokol za sinhronizacijo sistemske ure med več računalniki)
DNS – *Domain Name System* (sistem domenskih imen)
SSH – *Secure Shell* (protokol, ki omogoča varno oddaljeno povezavo v drug sistem)
HTML – *HyperText Markup Language* (označevalni jezik za izdelavo spletnih strani)
PHP – *Hypertext Preprocessor* (skriptni jezik, ki omogoča izdelavo dinamičnih spletnih strani)
SQL – *Structured Query Language* (strukturirani povpraševalni jezik)
DIT – *Directory Information Tree* (podatki, predstavljeni v hierarhični drevesni strukturi)
WPA – *Wi-Fi Protected Access* (protokol za zavarovanje brezžičnih omrežij)

Povzetek

Diplomsko delo smo pričeli s pregledom področja, v katerem smo zajeli: pregled različnih avtentikacijskih protokolov, podrobnejši opis treh pomembnejših avtentikacijskih protokolov (RADIUS, LDAP in KERBEROS), kriptografske metode, zgoščevalne funkcije in infrastrukturo PKI. Teoretični del diplomskega dela smo zaključili z varnostnimi vidiki protokola RADIUS in LDAP ter ranljivostmi protokola KERBEROS, kjer smo pridobili ustrezne citate ter jih komentirali. Nato smo v nadaljevanju najprej prikazali pregled izbranih avtentikacijskih strežnikov, pri katerih smo nato opravili primerjavo avtentikacije, kar je bil tudi cilj diplomskega dela. Tu smo uporabili strežnik FreeRADIUS, OpenLDAP in FreeIPA, ki smo jih namestili na virtualni strežnik ter jih ustrezno skonfigurirali. V okviru strežnika FreeRADIUS smo omogočili RADIUS avtentikacijo s pomočjo datoteke, podatkovne baze MySQL in imeniške storitve LDAP. Prav tako smo omogočili RADIUS avtentikacijo Linux uporabnikov, kjer smo uporabili ustrezen modul PAM. Za vse uspešne avtentikacije v okviru strežnika FreeRADIUS smo omogočili tudi beleženje ter pridobili čas opravljene avtentikacije za določenega uporabnika. Z uporabo strežnika OpenLDAP smo omogočili LDAP avtentikacijo Linux uporabnikov ter pri tem prav tako uporabili ustrezen modul PAM. Zadnji strežnik, imenovan FreeIPA, smo uporabili za pridobitev KERBEROS avtentikacije, kjer se je določen uporabnik lahko prijavil v sistem ter pri tem opravil avtentikacijo s pomočjo protokola KERBEROS. Na koncu smo razvili spletno aplikacijo ter prikazali možnost dostopa do nje z uporabo RADIUS in LDAP avtentikacije. Tako smo, poleg prijave v sistem, prikazali tudi možnost prijave v spletno aplikacijo z uporabo protokola RADIUS in LDAP.

Ključne besede:

protokol, strežnik, odjemalec, avtentikacijski strežnik, virtualni strežnik

Abstract

We started this thesis with the review in area of many different authenticational protocols, more detailed description of the three most important authenticational protocols (RADIUS, LDAP and KERBEROS) follow, also cryptographic methods, hash functions and PKI infrastructure are described. We concluded the theoretical part of the thesis with safety aspects of protocols RADIUS and LDAP and also vulnerability of KERBEROS protocol, where we obtained proper quotes, which we also commented. The next part brings an overview of the selected authenticational servers and also comparison of authentications follow. Finding differences between them was also the goal of this thesis. We used servers FreeRADIUS, OpenLDAP and FreeIPA, which we installed on the virtual server and configured properly. Within the FreeRADIUS server we enabled RADIUS authentication with file, MySQL database and LDAP directory service. We also enabled the RADIUS authentication for Linux users, with the use of suitable PAM module. Later we also enabled accounting and gained time of successful authentications within FreeRADIUS server. By using the OpenLDAP server we enable LDAP authentication for Linux users and also used suitable PAM module. We used the last server, called FreeIPA, to gain KERBEROS authentication where certain user can log in the system and carrying out the authentication by the help of KERBEROS protocol. Eventually we developed web application and showed the ability that it's possible to access it, with the use of RADIUS and LDAP authentication. Therefore we concluded, besides logging into the system, there is also a possibility of logging into the web application within the use of RADIUS and LDAP protocol.

Key words:

protocol, client, server, authentication server, virtual server

1 Uvod in opis problema

V vsakodnevnem življenju se, kakor tudi na področju računalništva, srečujemo z uporabo določenih storitev, kjer moramo izkazati svojo identiteto. Proces izkazovanja identitete imenujemo avtentikacija, v katerem moramo določeni storitvi dokazati, da smo res tisti, za katerega se izdajamo, da nam je posledično odobrena njena uporaba. Najpogostejši primer avtentikacije v vsakdanjem življenju steče pri nakupih z uporabo pametne kartice, kjer moramo podati geslo PIN za izvršitev transakcije ter z njim izkazati svojo identiteto. Na področju računalništva je avtentikacija potrebna skoraj povsod, kjer določena storitev ni na voljo vsem uporabnikom, sicer bi lahko vsi uporabniki dostopali do te storitve. Tipičen primer avtentikacije je prijava v sistem oziroma prijava v določeno spletno storitev, kjer moramo podati uporabniško ime in pripadajoče geslo. Torej nam proces avtentikacije zagotavlja določeno stopnjo varnosti, saj onemogoča uporabo določenih storitev neznanim uporabnikom. Ko izkažemo svojo identiteto sistemu [9], mora ta s pomočjo določenih procesov preveriti veljavnost vnesenih podatkov (uporabniško ime in geslo) in na podlagi teh določiti naše pravice ter zabeležiti avtentikacijo. Tako tu, poleg procesa avtentikacije, nastopi tudi proces avtorizacije in beleženja (znan pod kratico AAA).

Sistem lahko uporabi, poleg sistemske avtentikacije, tudi avtentikacijo s pomočjo zunanjega vira (avtentikacijski strežnik). To lahko opravi [9] s pomočjo protokola RADIUS, ki združuje vse omenjene procese (avtentikacija, avtorizacija, beleženje). Poleg protokola RADIUS lahko sistem za sam proces avtentikacije uporabi tudi protokol LDAP ali KERBEROS. V primeru protokola RADIUS in LDAP se uporabniško geslo pošlje prek medmrežja (angl. *internet*) do strežnika, ki preveri veljavnost, medtem ko v primeru protokola KERBEROS to ni tako. Pri protokolu KERBEROS pridobi uporabnik vstopnico (angl. *ticket*), ki jo mora uspešno dešifrirati z uporabo svojega gesla (le pravilno geslo dešifrira vstopnico) ter jo nato uporablja za sam proces avtentikacije. Posledično se uporabniško geslo ne prenese skozi medmrežje, kar poveča varnost. Čeprav protokol RADIUS in LDAP ne prenašata gesla v čistopisu (angl. *cleartext*), je avtentikacija s pomočjo protokola KERBEROS še kako dobrodošla. V sklop varne komunikacije sodita poleg varnega avtentikacijskega protokola tudi uporaba varnega šifrirnega algoritma za zakrivanje prenesenih sporočil in tudi uporaba varne zgoščevalne funkcije z namenom preverjanja integritete sporočila, saj želimo ohraniti zaupnost in celovitost sporočil v sami komunikaciji.

V nadaljevanju diplomskega dela smo najprej, v sklopu drugega poglavja, prikazali širši pregled računalniške varnosti, kjer smo podrobneje opisali zgoraj omenjene avtentikacijske protokole (RADIUS, LDAP in KERBEROS) in številne zgoščevalne funkcije ter šifrirne

algoritme. Poleg tega smo zajeli tudi infrastrukturo PKI in se osredotočili na protokol SSL/TLS, katerega uporaba je danes zelo razširjena. Nato smo v tretjem poglavju opisali različne avtentikacijske strežnike, med katerimi smo izbrali tri, v četrtem poglavju pa smo prikazali primerjavo avtentikacij s pomočjo teh avtentikacijskih strežnikov (FreeRADIUS, OpenLDAP in FreeIPA). V okviru strežnika FreeRADIUS smo prikazali avtentikacijo s pomočjo protokola RADIUS, medtem ko smo v primeru strežnika OpenLDAP prikazali avtentikacijo s pomočjo protokola LDAP. Z zadnjim strežnikom FreeIPA smo zajeli avtentikacijo s pomočjo protokola KERBEROS. S temi primeri smo prikazali slabosti oziroma prednosti teh avtentikacijskih protokolov ter različne avtentikacijske protokole za izvedbo avtentikacije. Iz te primerjave je vidna tudi varnost posameznega avtentikacijskega protokola, ki je danes še kako pomembna, saj si ne želimo, da bi morebitni napadalec pridobil naše geslo ter se izdajal za nas. V zadnjem, petem poglavju, smo se osredotočili na znanje, ki smo ga pridobili v diplomskem delu in opisali, kaj bi lahko glede avtentikacij še izboljšali. Prav tako smo navedli, zakaj je bila pomembna primerjava avtentikacije med različnimi avtentikacijski strežniki.

2 Pregled področja

2.1 Avtentikacija, avtorizacija in beleženje

Varnostni arhitekturni model [9, 15], znan pod kratico AAA, omogoča nadzor nad uporabniki določene storitve oziroma omrežja. Omenjeni model nadzira tako, da za vsakega uporabnika preveri, ali ima dostop do neke storitve oziroma omrežja, ter mu nato dodeli pravice, ki mu pripadajo, in beleži uporabo te storitve oziroma omrežja. Model AAA tako zajema proces avtentikacije, avtorizacije in beleženja, ki smo jih podrobneje opisali v nadaljevanju. Prav tako velja model AAA za standardni način, s katerim je omogočeno preverjanje identitete uporabnika, dodelitev njegovih pravic ter beleženje uporabe določene storitve oziroma omrežja.

Avtentikacija je proces, ki se izvede največkrat med prvimi [15] procesi ter služi za pridobitev dostopa do omrežja oziroma neke storitve. Ta proces namreč preveri identiteto uporabnika, in sicer največkrat na podlagi posredovanega uporabniškega imena in gesla. Po posredovanem uporabniškem imenu in geslu ima uporabnik omogočen dostop do te storitve oziroma omrežja, seveda v primeru pravih posredovanih podatkov. Storitev oziroma omrežje s pomočjo pravilnega pridobljenega gesla verjame, da je uporabnik na drugi strani res tisti, za katerega se izdaja, saj samo ta uporabnik z uporabniškim imenom pozna svoje geslo ter posledično avtenticira uporabnika. Poleg avtentikacije z geslom poznamo tudi avtentikacijo s pomočjo certifikatov in PIN gesla (pametne kartice). Avtentikacija s pomočjo certifikatov [9], ki je del kriptografije z javnim ključem (angl. *public key infrastructure*), predstavlja varnejšo avtentikacijo ter posledično bolj zanesljivo. Zanimiva pa je tudi avtentikacija s pomočjo biometričnih značilnosti, kot so prstni odtis, razpoznava obraza in glasu.

Ko je uporabnik avtenticiran oziroma pridobi dostop do neke storitve oziroma omrežja, pridobi določene pravice, kar je znano pod imenom avtorizacija. Proces avtorizacije namreč uporablja množico pravil [9], s katerim uporabniku določi, kaj sme početi oziroma česa ne na določenem sistemu. Vsak uporabnik ima lahko neke omejitve ali določene privilegije, odvisno od tipa uporabnika, ki je avtenticiran. Običajno postane uporabnik član [15] določene skupine ali več skupin, ki imajo določene pravice. Značilen primer avtorizacije je dodeljen bralni dostop do datoteke za nekega uporabnika, ki se nahaja v določeni skupini.

Zadnji proces, ki sledi avtorizaciji, imenujemo beleženje. Ko je namreč uporabnik avtenticiran ter ima dodeljene pravice, nastopi še nadzor njegove seje v okviru procesa beleženja. Proces beleženja zajema statistiko [9] o aktivni seji uporabnika in informacijo o

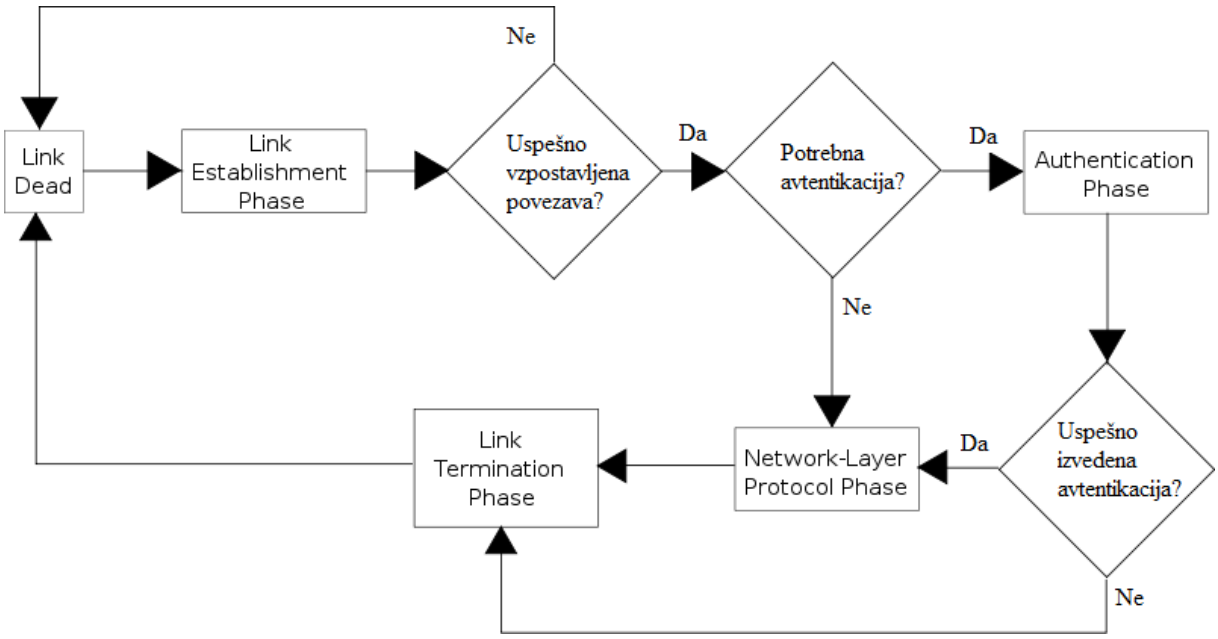
času uporabe storitve oziroma omrežja z namenom zaračunavanja uporabe. Tako se proces beleženja uporablja za namene, kot so [15]: zaračunavanje uporabe, analizo, planiranje zmogljivosti in aktivni nadzor. Prav tako lahko proces beleženja zajema dogodke, kot je napaka pri avtentikaciji in avtorizaciji. Osnovne informacije, ki se beležijo za določeno uporabniško sejo, so identiteta uporabnika in začetek ter konec uporabe neke storitve oziroma omrežja. Pogosta informacija, ki se beleži, je tudi podatek o prenesenih oziroma prejetih podatkih uporabnika v času trajanja njegove seje. Podatki, ki se pridobijo v okviru procesa beleženja, so zelo potrebni s strani administratorjev določenega strežnika tipa AAA.

2.2 Avtentikacijski protokoli

Eden izmed načinov avtentikacije je uporaba avtentikacijskega protokola. Avtentikacijski protokol se značilno uporablja na povezavni plasti, ki povezuje odjemalca s ponudnikom storitve (angl. *network access server*), saj se omrežna plast vzpostavi po uspešni avtentikaciji. Ponudnik storitve deluje kot posrednik med uporabnikom in strežnikom, saj namesto uporabnika dostopa do strežnika, kjer pridobi potrebno informacijo. V nadaljevanju smo prikazali najbolj razširjene avtentikacijske protokole, njihov namen ter jih opisali.

2.2.1 PPP

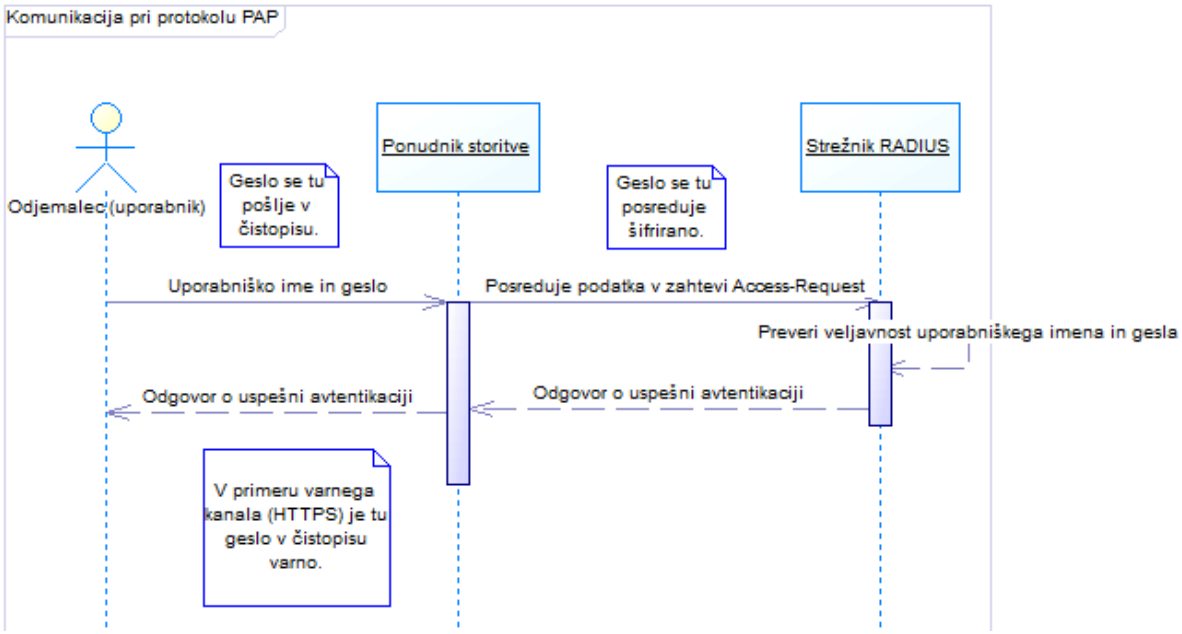
Protokol [33, 53], ki se pogosto uporablja za vzpostavitev neposredne povezave med dvema vozliščema v omrežju, je znan pod kratico PPP (postopek vzpostavitve pozave prikazuje spodnja slika 1, ki prikazuje vse nastopajoče faze). Ta protokol zagotavlja avtentikacijo, ki privzeto ni obvezna, ter nudi šifriranje prenesenih podatkov. Ponudnik internetnih storitev (angl. *internet service provider*) zagotavlja svojim strankam povezavo s pomočjo protokola PPP, saj se na ta način lahko ponudnik internetnih storitev odzove na zahteve stranke. Te zahteve vodi v internetno omrežje ter posreduje odgovore na zahteve nazaj k strankam. Protokol PPP uporablja omrežni protokol IP in nadomešča oziroma zagotavlja povezavno plast protokolarnega sklada TCP/IP. Sam protokol v osnovi pripravi TCP/IP pakete našega računalnika in jih posreduje strežniku, kjer se nato pošljejo v medmrežje (angl. *internet*). Poleg tega protokol nudi PPP dvosmerno komunikacijo (angl. *Full duplex*) in se uporablja na več fizičnih medijih. Prav tako lahko upravlja sinhrono in tudi asinhrono komunikacijo.



Slika 1: Postopek vzpostavitve pozave pri protokolu PPP [53].

2.2.2 PAP

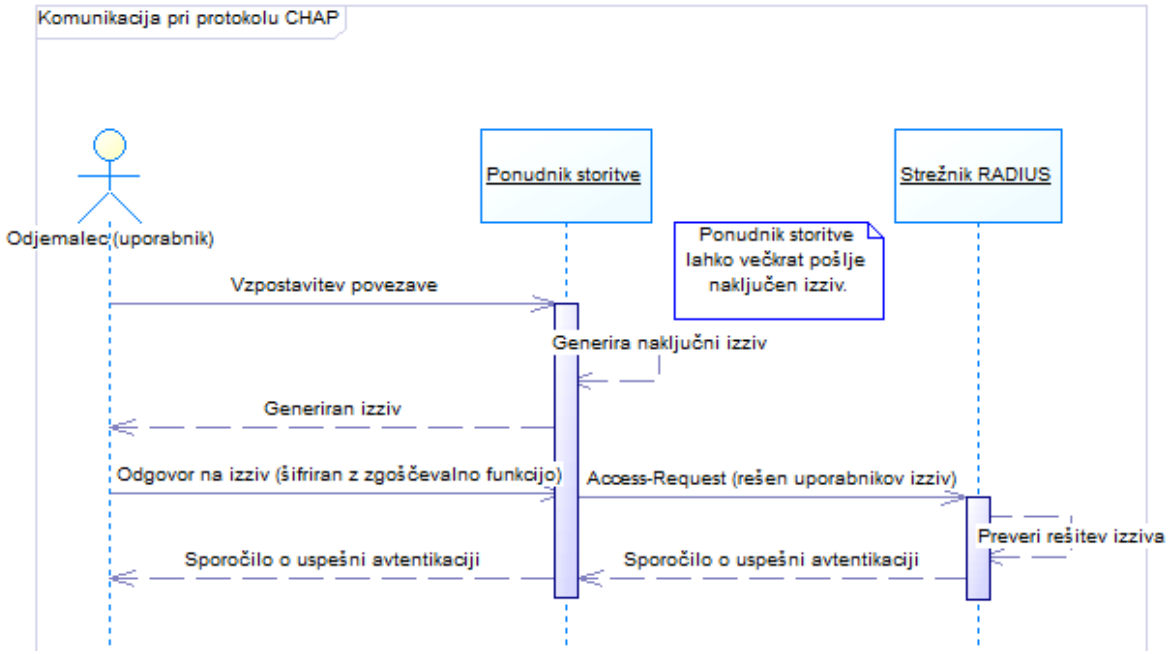
Kratica PAP označuje avtentikacijski protokol, ki preveri pravilnost uporabniškega imena in gesla uporabnika, preden mu je upravičen dostop do virov strežnika. Pri tem protokolu nastopi tako imenovana avtentikacija s pomočjo gesla, saj se uporabnik predstavi s pomočjo posredovanega gesla. Zgoraj opisani protokol PPP uporablja avtentikacijski protokol PAP za preverjanje identitete uporabnika ter je podprt s strani številnih strežnikov. Ponudnik storitve (angl. *network access server*) v tem primeru pridobi [15] identifikacijsko število (angl. PAP ID) oziroma uporabniško ime in geslo ter ju pošlje v obliki zahteve do strežnika, kjer se podatka preverita. Vendar ima protokol PAP veliko pomanjkljivost, saj ta med uporabnikom in ponudnikom storitve prenaša geslo v čistopisu (angl. *cleartext*), kar seveda ni varno. Prenos gesla med ponudnikom storitve in strežnikom namreč ni v čistopisu, saj je geslo v tem primeru šifrirano. Tako se protokol PAP uporablja največkrat takrat, ko oddaljeni strežnik ne podpira varnejšega avtentikacijskega protokola, kot je PAP. Da bi povečali varnost protokola PAP, se lahko protokol uporabi znotraj varnega kanala (angl. *secure channel*), kot je to pri protokolu HTTPS, ki prav tako uporablja varnostni kanal. Po vzpostavitvi varnega kanala lahko varno pošiljamo po medmrežju (angl. *internet*) občutljive podatke, med katere sodijo gesla. Opisan potek komunikacije prikazuje spodnja slika 2.



Slika 2: Primer komunikacije pri protokolu PAP.

2.2.3 CHAP

Varnejši avtentikacijski protokol, predstavljen s kratico CHAP, je bil zasnovan z namenom izboljšave protokola PAP. Pomembno pri protokolu CHAP je to, da preprečuje pošiljanje gesla po medmrežju v čistopisu (angl. *cleartext*) ter je prav tako, kot je protokol PAP, pripravljen za potrebe protokola PPP. Protokol CHAP avtenticira uporabnika na ta način [15]: ko je povezava do ponudnika storitve (angl. *network access server*) vzpostavljena, ponudnik storitve generira naključni izziv (angl. *random challenge*) ter ga pošlje uporabniku v berljivi obliki skupaj z identifikatorjem (angl. *identifier*). Uporabnik nato odgovori na izziv s pomočjo rezultata enosmerne zgoščevalne funkcije (angl. *one-way hash function*), ki ga izračuna z uporabo pridobljenega identifikatorja, izziva in njegovega gesla. Odgovor uporabnika uporabi ponudnik storitve, da bi kreiral zahtevo, ki se nato pošlje do strežnika. Ko ponudnik storitve pridobi odgovor strežnika, sporoči uporabniku, ali ima omogočen dostop do vira oziroma ali je pravilno odgovoril na podan izziv. Ponudnik storitve lahko naključno ponovi zahtevo za avtentikacijo, in sicer s ponovno poslanim izzivom uporabniku. Vendar je ta izziv drugačen kot prejšnji, kar zagotavlja še večjo varnost v primerjavi s protokolom PAP. Ker se izziv menja oziroma ni enak predhodnemu, je težko napasti s ponavljanjem (angl. *replay attack*). Opisan potek komunikacije prikazuje tudi spodnja slika 3.



Slika 3: Primer komunikacije pri protokolu CHAP.

2.2.4 MS-CHAP

Je avtentikacijski protokol, ki je identičen protokolu CHAP, le da je protokol MS-CHAP razvilo podjetje Microsoft. Torej gre za identičen postopek avtentikacije, kot smo ga opisali v zgornjem primeru, kjer smo govorili o protokolu CHAP. Edina razlika med protokolom CHAP in MS-CHAP je v formatu polja Value [15], ki vsebuje specifična polja za protokol MS-CHAP. Eden izmed teh specifičnih polj, je tako imenovano polje `NT-Response`, ki vsebuje šifrirano uporabniško ime in geslo. Protokol MS-CHAP vsebuje tudi dodatne lastnosti oziroma izboljšave, ki jih standardni protokol CHAP ne vsebuje. Primer uporabne dodatne lastnosti je uporabnikova zmožnost spreminjanja gesla ter bolj opisna sporočila v primeru napak, kar posledično privede do lažje odprave napake, za katere so definirane kode. Protokol MS-CHAP izhaja v dveh inačicah, in sicer protokol MS-CHAPv1 in MS-CHAPv2. Druga inačica protokola MS-CHAP zagotavlja boljšo varnost za oddaljene dostope do določenega strežnika [55]. Nudi namreč dvosmerno avtentikacijo (angl. *two-way authentication*), ki je znana tudi pod imenom vzajemna avtentikacija (angl. *mutual authentication*). Prav tako niso v drugi inačici protokola MS-CHAP prisotne nekatere lastnosti, ki jih vsebuje protokol prve inačice, saj te niso več potrebne.

2.2.5 EAP

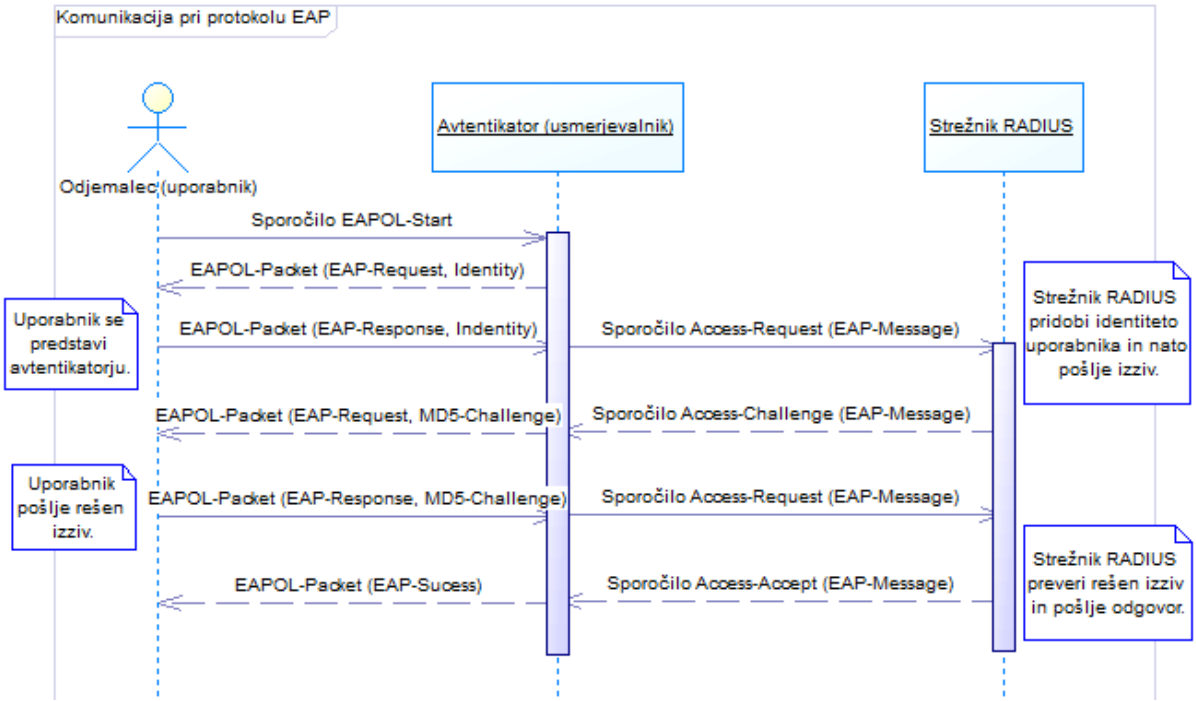
Eden izmed najbolj razširjenih avtentikacijskih protokolov je protokol, znan pod kratico EAP. Uporablja se tako v industrijskem standardu 802.1x kot tudi pri WPA2-Enterprise, ki se uporabljata za varnost pri dostopu do medmrežja (angl. *internet*). Standard 802.1x uporablja protokol EAP prek lokalnega omrežja (angl. *local area network*), medtem ko ga WPA2-Enterprise uporablja prek brezžičnih omrežij (angl. *Wi-Fi*). EAP v resnici ne nastopa kot pravi protokol, temveč nastopa kot okvir za protokole, saj definira zgolj obliko sporočil. EAP kot okvir uporablja številne metode, s pomočjo katerih avtenticira uporabnika. Med bolj poznane metode sodijo metode, kot so: MD5, LEAP, GTC, TLS, TTLS, PEAP in MSCHAPv2 [15], vendar v praksi največkrat zasledimo uporabo metode TLS oziroma TTLS. Metoda TLS temelji na avtentikaciji s pomočjo certifikatov. Tu nastopi tako imenovana vzajemna avtentikacija (angl. *mutual authentication*), saj strežnik kot tudi odjemalec predložita svoj certifikat, s katerim se predstavita in izkažeta, da sta res tista, za katera se izdajata. Ta metoda je zelo varna, vendar je zato implementacija toliko težja. Poleg metode TLS je velikokrat uporabljena metoda TTLS, ki kreira varnostni kanal (angl. *secure channel*), v katerem je lahko vsebovana druga avtentikacijska metoda. Tipično tu nastopi uporaba protokola PAP znotraj varnostnega kanala, kar omogoča varnejšo uporabo protokola PAP, saj ta prenaša geslo v čistopisu (angl. *cleartext*). Tudi ta metoda je zelo varna in zelo razširjena med uporabniki.

Okvir EAP vključuje tri temeljne komponente. Prva komponenta je avtentikator (angl. *authenticator*), ki nadzira kontrolo dostopa do omrežja. Primer avtentikatorja je lahko stikalo (angl. *switch*) oziroma usmerjevalnik (angl. *router*), dostopna točka (angl *access point*) ali ponudnik storitve (angl. *network access server*). Avtentikator velja kot vmesnik med uporabnikom in avtentikacijskim strežnikom, ki posreduje in prevaja komunikacijo med njima. Uporaba avtentikatorja kot vmesnika privede do dveh naslednjih prednosti [15]. Prva prednost privede do razširljivosti okvirja EAP, saj se lahko uveljavijo nove avtentikacijske metode na strani avtentikacijskga strežnika ali uporabnika brez spreminjanja funkcionalnosti na avtentikatorju. Kot drugo prednost uporabe avtentikatorja navajamo uporabo centralnega strežnika za avtentikacijo.

Ostali dve temeljni komponenti okvirja EAP pa sta že omenjeni uporabnik in avtentikacijski strežnik. Pri uporabniku nastopa del programske opreme za avtentikacijo, ki se nahaja na njegovem računalniku. Ko je avtentikacija uspešna oziroma ko avtentikacijski strežnik odobri dostop uporabniku, mu avtentikator dodeli dostop do omrežja [15]. Uporabnik navadno podpira številne EAP avtentikacijske metode, vendar pri nekaterih uporabnikih to ni tako. Nekateri uporabniki namreč ne podpirajo določenih metod protokola EAP. Za izvedeno

komunikacijo oziroma uporabo določene avtentikacijske metode morata uporabnik in tudi avtentikacijski strežnik podpirati izbrano avtentikacijsko metodo. Kot zadnja je omenjena komponenta, imenovana avtentikacijski strežnik, pri avtentikaciji glavnega pomena. Strežnik namreč odloči oziroma odgovori avtentikatorju, ali je določenemu uporabniku dovoljen dostop do medmrežja ali ne.

Sama komunikacija med tremi temeljnimi komponentami poteka na naslednji način [15]: komunikacija se prične z uporabnikom, ki pošlje sporočilo (`EAPOL-Start`) avtentikatorju, s katerim avtentikatorju sporoči svojo prisotnost. Avtentikator nato pošlje sporočilo nazaj, v katerem povpraša uporabnika o njegovi identiteti. Ko uporabnik odgovori avtentikatorju s podatki o svoji identiteti, ta to sporočilo posreduje avtentikacijskemu strežniku. Strežnik nato v primeru, da gre za veljavno identiteto, odgovori s sporočilom (`Access challenge`), v katerem zahteva rešitev izziva (angl. *challenge*). To sporočilo seveda najprej pridobi avtentikator, ki ga nato posreduje uporabniku. Uporabnik nato odgovori na zastavljen izziv s pošiljanjem sporočila nazaj k strežniku, prek avtentikatorja. Avtentikacijski strežnik lahko sedaj preveri rešitev izziva in v primeru prave rešitve odgovori s sporočilom `Access-Accept`, ki omogoča uporabniku dostop do medmrežja. To sporočilo avtentikator pošlje uporabniku. Celoten potek komunikacije, skupaj z vsemi nastopajočimi sporočili, prikazuje spodnja slika 4.



Slika 4: Primer komunikacije pri protokolu EAP.

Kot je razvidno z zgornje slike 4, avtentikator in uporabnik komunicirata neposredno le v začetnih korakih, medtem ko v nadaljnji komunikaciji avtentikator posreduje odgovore uporabnika do avtentikacijskega strežnika in obratno [15]. Naloga avtentikatorja je namreč tudi posredovanje sporočil, kakor tudi pretvorba sporočil med uporabnikom in avtentikacijskim strežnikom (pretvorba je prav tako vidna na zgornji sliki 4).

2.3 Protokol RADIUS

Gre za primer protokola (določen v dokumentu RFC 2865), ki podpira model AAA ter tako zagotavlja avtentikacijo, avtorizacijo in beleženje [9]. Temelji na transportnem protokolu UDP, ki je nepovezavni (angl. *connectionless*), kar posledično privede do komunikacije med odjemalcem in strežnikom brez predhodnega dogovora. V okviru procesa avtentikacije podpira več protokolov, kot so: PAP, (MS)CHAP in EAP, ter omogoča avtentikacijo s pomočjo protokola PAP in CHAP prek protokola PPP. Poleg tega uporablja varnostni model, imenovan hop-by-hop (angl. *hop-by-hop security model*), ter ponuja številne prilastke, ki vsebujejo potrebne informacije za izvedbo procesa avtentikacije oziroma avtorizacije in beleženja. Poleg že definiranih prilastkov omogoča tudi kreiranje lastnih, znanih pod kratico VSA, kar privede do razširitve protokola RADIUS.

Protokol ima poleg svojih prednosti tudi številne slabosti oziroma omejitve, med katerimi [9] so naslednje:

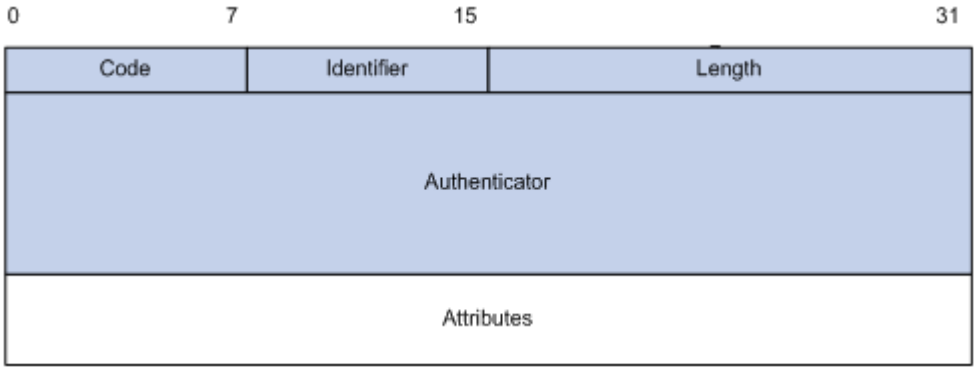
- v primeru več medstrežnikov (angl. *proxy*) RADIUS, so vsi podatki dosegljivi na vsakem skoku (angl. *hop*) med strežniki RADIUS. To seveda ni varno, ko nastopijo občutljivi podatki, kot so certifikati in gesla,
- protokol RADIUS ne sledi oziroma si ne zapomni informacije v zvezi s prejšnjo uporabniško sejo, ki bi jih lahko uporabil pri naslednji (nastavitve uporabnika) in je tako brezstanjski (angl. *stateless*),
- v primeru šifriranja oziroma zakrivanja gesel uporablja protokol zgoščevalno funkcijo MD5.

Protokol RADIUS se uporablja na različnih lokacijah, vendar je bolje poznan internetnim ponudnikom storitev (angl. *internet service provider*s) in omrežnim administratorjem. Pogost primer uporabe protokola RADIUS je pri brezžičnih dostopnih točkah z WPA-2-Enterprise šifriranjem in v primeru, ko želi uporabnik dostopati do medmrežja (angl. *internet*) prek internetnega ponudnika storitev [15]. Protokol RADIUS je prisoten oziroma dobro uveljavljen tudi v industriji, kar pomeni, da ima določeno prihodnost, vendar ne bo večno standard pri

izbiri protokola tipa AAA. Protokol namreč že ima svojega naslednika, imenovanega DIAMETER, ki odpravlja pomanjkljivosti protokola RADIUS, vendar še ni uveljavljen.

2.3.1 Sporočila in oblika sporočil

RADIUS sporočila oziroma paketi imajo določen format oziroma strukturo, ki je razdeljena na pet posameznih polj. Sporočilo vsebuje, poleg ostalih polj, polji `Code` in `Attributes`, ki sta ključnega pomena [15]. Polje `Code` označuje tip sporočila, medtem ko polje `Attributes` vsebuje bistvene podatke za sam protokol RADIUS. Spodnja slika 5 prikazuje celoten format sporočila protokola RADIUS.



Slika 5: Format sporočila protokola RADIUS.

Polje `Code`, dolžine enega zloga (angl. *octet*), označuje tip sporočila. Njegova vrednost je vsebovana v vsakem sporočilu, ki je predstavljeno kot decimalna številka. Z vrednostjo polja `Code` sporočilo določa nekatere zahteve, kot je vsebovanost določenih AVP prilastkov v sporočilu, ki so podani v polju `Attributes`. Sporočila, ki vsebujejo neveljaven tip sporočila oziroma neveljavno vrednost polja `Code`, se enostavno zavržejo brez opozorila. Spodnja tabela 1 prikazuje seznam definiranih kod oziroma veljavne vrednosti polja `Code` za sporočila protokola RADIUS [15].

Koda ukaza	**Tip sporočila**	**Pošiljatelj**
1	`Access-Request`	Ponudnik storitve
2	`Access-Accept`	Strežnik RADIUS
3	`Access-Reject`	Strežnik RADIUS
4	`Accouting-Request`	Ponudnik storitve (angl. *network access server*)
5	`Accouting-Response`	Strežnik RADIUS
11	`Access-Challenge`	Strežnik RADIUS
12	`Status-Server` (poskusno)	
13	`Status-Client` (poskusno)	
255	Rezervirano	

Tabela 1: Seznam vrednosti polja `Code` znotraj sporočila protokola RADIUS.

Naslednje je polje `Identifier`, ki predstavlja za vsako sporočilo enolični identifikator in je dolžine enega zloga. Njegov namen je povezava odgovorov z zahtevami [15], saj je protokol RADIUS koračni protokol, kar pomeni, da mora odjemalec vedeti, kateri odgovor pripada določeni zahtevi. Odjemalec v primeru izgubljenega sporočila pošlje ponovno zahtevo, ki vsebuje isti enolični identifikator, kot ga je vsebovalo izgubljeno sporočilo. Strežnik RADIUS se nato odzove na zahtevo z identifikatorjem, ki se je nahajal v odjemalčevi ponovno poslani zahtevi. Polje `Identifier` pomaga tudi pri ugotavljanju oziroma prestrezanju duplikatov. Pri prestrezanju podvojenih sporočil pripomorejo tudi [9] izvorni naslov IP, izvorna UDP vrata in pretečen čas med morebitnima podvojenima sporočiloma.

Polje dolžine dveh zlogov, ki vsebuje dolžino celotnega sporočila, vključno z glavo, imenujemo `Length`. To pomeni, da je dolžina sporočila oziroma vrednost polja izračunana s pomočjo vseh polj sporočila ter je podana v zlogih [9, 15]. Najmanjša dolžina celotnega sporočila je omejena na 20 zlogov, medtem ko je največja možna dolžina sporočila omejena na 4096 zlogov. Dolžino sporočila oziroma vrednost polja `Length` preveri strežnik RADIUS, ko prejme sporočilo, in sicer z namenom preverjanja integritete sporočila. Če je sporočilo daljše, kot je največja možna dolžina sporočila, se sporočilo skrajša oziroma se ignorira preostale podatke, če pa je sporočilo krajše od najmanjše možne dolžine, se ga enostavno zavrže.

Za samo varnost oziroma preverjanje verodostojnosti sporočil je poskrbljeno s poljem `Authenticator`, saj tu nastopi podpis sporočila. S pomočjo tega polja se pregleda oziroma

preveri integriteta vsebine sporočila [9]. Dolžina polja znaša 16 zlogov. Polje je lahko različno oblikovano oziroma njegova vrednost je lahko pridobljena s pomočjo različnih atributov. Odvisno je, ali gre za zahtevo, ki jo pošlje odjemalec strežniku, ali gre za odgovor, ki ga pošlje strežnik odjemalcu. Prav tako je odvisno, ali gre za tip sporočila `Access-Request` ali za `Accouting-Request`. V primeru [15], ko gre za zahtevo, se polje nanaša na `Request Authenticator`, medtem ko se v primeru odgovora polje nanaša na `Response Authenticator`. Ko imamo opravka s tipom paketa `Access-Request`, je vrednost polja `Request Authenticator` 128-bitno naključno število, ki se ne ponavlja in so tako napadi težji. Vrednost polja `Response Authenticator`, ko nastopi tip paketa `Access-Accept` oziroma `Access-Reject` ali `Access-Challenge`, pa je izračunana s pomočjo zgoščevalne funkcije MD5, ki vsebuje več podanih atributov za izračun. Ti atributi so v tem primeru vrednost polja `Code`, `Identifier`, `Length`, `Request-Authenticator` in `Attribues`, med njimi pa nastopi še vrednost skupne skrivnosti (angl. *shared secret*), ki je poznana odjemalcu in strežniku RADIUS. Skupno skrivnost si bomo podrobneje ogledali v podpoglavju, ki se nanaša na varnost protokola RADIUS.

Ko zahteva vsebuje uporabniško geslo, je ta šifrirana s pomočjo zgoščevalne funkcije MD5. Ta funkcija poleg ostalih atributov prejme tudi omenjeno skupno skrivnost kot atribut, zato je potrebno [15], da je skupna skrivnost na strani odjemalca kakor tudi na strani strežnika enaka. Saj lahko le tako strežnik dešifrira uporabnikovo geslo.

Zadnje polje sporočila RADIUS je polje `Attributes`, ki vsebuje prilastke, znane pod kratico AVP. Celotna RADIUS komunikacija temelji na prilastkih strežnika in odjemalca, saj ti vsebujejo dejanske podatke, ki jih potrebujeta strežnik RADIUS in odjemalec v določeni uporabniški seji. Tako lahko odjemalec, s pomočjo prilastkov, posreduje strežniku RADIUS potrebne podatke ter prav tako tudi strežnik RADIUS odjemalcu. Da bi povečali varnost, je omejena prisotnost oziroma pošiljanje nekaterih prilastkov v samem sporočilu protokola RADIUS [9]. V primeru prilastka `User-password` se ta lahko pojavi le v odjemalčevi zahtevi, medtem ko ga strežnikov odgovor ne sme vsebovati. Tako se lahko ta prilastek prenese le enkrat v okviru procesa avtentikacije oziroma avtorizacije. Prilastki so predstavljeni s posebnim formatom, imenovanim TLV, ki zajema polje `Type`, `Length` in `Value`. Kadar je zahtevana avtentikacija uporabnika, vsebuje zahteva prilastek `User-Name` in `User-Password` poleg ostalih potrebnih prilastkov znotraj sporočila `Access-Request`. Prilastke smo podrobneje predstavili v naslednjem podpoglavju.

2.3.2 Prilastki

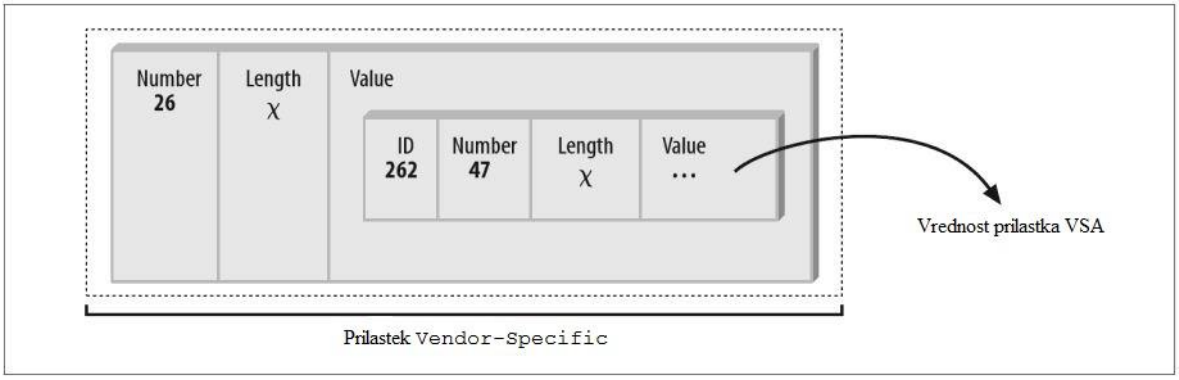
Prilastki predstavljajo vrednosti polja `Attributes` v sporočilu protokola RADIUS. So ključnega pomena v samem protokolu, saj z njimi posredujemo specifične informacije, ki se uporabijo pri sami avtentikaciji oziroma avtorizaciji in beleženju. Prilastki so razvrščeni bodisi v skupino `check` ali `reply` [15]. Tisti iz skupine `check` se pošiljajo od odjemalca k strežniku, medtem ko se prilastki iz skupine `reply` pošiljajo v obratni smeri. Prilastki tako služijo kot prenašalci informacije med odjemalcem in strežnikom.

Prilastki so zapisani v posebnem formatu, znanim pod kratico TLV, ki določa tri polja. Vsak prilastek je tako predstavljen s tremi polji, katerih skupna dolžina mora znašati najmanj tri zloge (angl. *octets*), kjer je vsak zlog namenjen enemu polju prilastka. Prvo polje formata TLV je polje `Type`, ki označuje tip prilastka s pomočjo vsebovane številke. Vsaki številki pripada določeno ime, s katerim lažje prepoznamo tip prilastka, vendar polje `Type` vsebuje le številko, ki označuje določen prilastek. Značilni primeri prilastka so `User-Name(1)`, `User-Password(2)` in `NAS-IP-Address(4)`, kjer ima vsak izmed njih v oklepaju podano pripadajočo številko [15]. Obseg številk je med 1 in 255, kar pomeni, da je toliko tudi možnih prilastkov. Zanimiva je številka 26 oziroma prilastek, imenovan `Vendor-Specific(26)`, ki omogoča razširitev protokola RADIUS oziroma je namenjen posameznikom (angl. *vendors*) za implementacijo lastnih prilastkov. Vrednost prilastka `Vendor-Specific(26)` vsebuje posebne prilastke, znane pod kratico VSA. Naslednje polje, ki nam pove dolžino prilastka, je polje `Length`, kjer njegova vrednost znaša najmanj tri oziroma več. Zadnje polje formata TLV oziroma polje prilastka, ki ga imenujemo `Value`, pa vsebuje vrednost določenega prilastka. To polje je zahtevano tudi v primeru, če ta nima definirane vrednosti (angl. *null*). Polja `Value` se razlikujejo po velikosti, saj lahko polje zaseže več zlogov, kar je odvisno od prilastka. Vsebuje lahko različne podatkovne tipe, ki jih prikazuje spodnja tabela 2 z njihovo dolžino [9].

Podatkovni tip	**Dolžina v zlogih**	**Velikost**
Integer (INT)	4	32-bitna
Enumerated (ENUM)	4	32-bitna
String (STRING)	1−253	Spremenljivka
Naslov IP (IPADDR)	4	32-bitna
Date (DATE)	4	32-bitna
Binary (BINARY)	1	1 bit

Tabela 2: Obstoječi podatkovni tipi znotraj polja `Value` pri prilastkih protokola RADIUS.

Kot smo omenili, omogoča protokol RADIUS razširitev protokola s pomočjo prilastka `Vendor-Specific(26)`. Ta prilastek kot vrednost vsebuje posebne prilastke, znane pod kratico VSA. To pomeni, da ti prilastki (VSA) predstavljajo razširitev običajnih prilastkov (AVP) in se izvajajo v polju `Value` prilastka `Vendor-Specific(26)`. Ti prilastki (VSA) omogočajo posameznikom definirati svoje lastne prilastke, kar privede do večje fleksibilnosti protokola RADIUS in to omogoča posameznikom prilagoditi implementacijo protokola [9]. Format prilastkov (VSA) je v osnovi enak formatu običajnih prilastkov (AVP), vendar z dodatnim poljem `Vendor ID`. To dodatno polje vsebuje štiri zloge, ki predstavljajo razvijalca oziroma lastnika prilastka. Spodnja slika 6 prikazuje format prilastka (VSA) znotraj polja `Value` v prilastku `Vendor-Specific(26)`.



Slika 6: Format prilastka VSA znotraj prilastka `Vendor-Specific` [9].

Pri procesu beleženja nastopi sporočilo tipa `Accouting-Request`, ki zahteva vključitev nekaterih prilastkov oziroma so za proces beleženja pomembni. V nadaljevanju bomo prikazali nekatere izmed teh prilastkov. Prilastek, ki ni obvezen, vendar je vedno vključen, imenujemo `Acct-Status-Type(40)`. Ta prilastek lahko vsebuje več različnih vrednosti, med katerimi so najpogostejše: `Start`, `Interim-Update` in `Stop`. Ponudnik storitve (angl. *network access server*) lahko z uporabo vrednosti `Start` ali `Stop`, sporoči strežniku RADIUS začetek oziroma konec procesa beleženja za določenega uporabnika. Med samo uporabniško sejo pa lahko ponudnik storitve uporabi vrednost `Interim-Update` ter s tem posodobi podatke beleženja za določeno uporabniško sejo. Preostali pomembni prilastki oziroma nekateri tudi obvezni pri procesu beleženja se nahajajo v spodnji tabeli 3 skupaj z navedenim namenom uporabe [15].

Prilastek	Namen
Acct-Input-Octets(42)	Označuje prejete zloge med sejo in se uporablja skupaj s prilastkom Acct-Status-Type(40), ki vsebuje vrednost Interim-Update oziroma Stop.
Acct-Output-Octets(43)	Označuje poslane zloge med sejo in se uporablja skupaj s prilastkom Acct-Status-Type(40), ki vsebuje vrednost Interim-Update oziroma Stop.
Acct-Session-Id(44)	Obvezen za vse pakete tipa Accouting-Request, saj predstavlja enolično vrednost, ki se uporablja za povezavo vrednosti Start, Interim-Update in Stop prilastka Acct-Status-Type(40). Vse te vrednosti morajo imeti znotraj ene seje isto vrednost prilastka.
Acct-Session-Time(46)	Označuje čas trajanja seje v sekundah in se uporablja s prilastkom Acct-Status-Type(40), ki vsebuje vrednost Interim-Update oziroma Stop.
Acct-Terminate-Cause(49)	Nastopi skupaj s prilastkom Acct-Status-Type in z vrednostjo Stop. Vrednost prilastka definira tistega, ki je zahteval zaključek procesa beleženja.

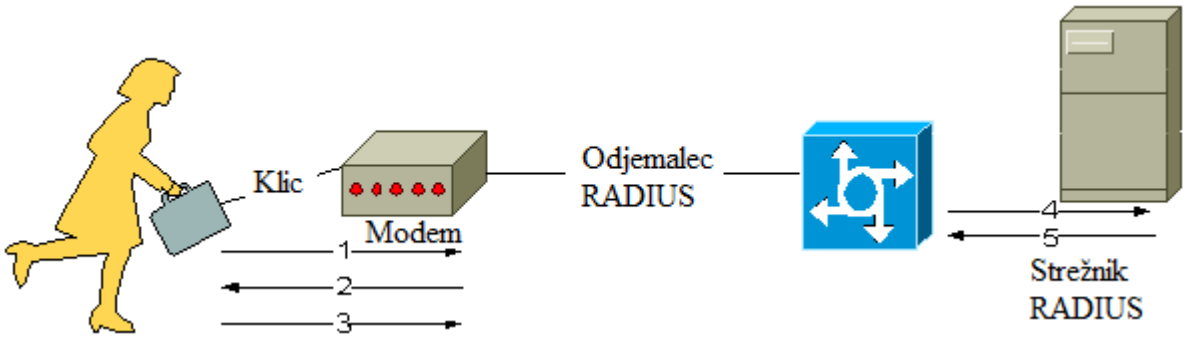
Tabela 3: Pomembni prilastki pri procesu beleženja.

2.3.3 Arhitektura in komunikacija

Osnovna arhitektura pri protokolu RADIUS vključuje tri udeležene stranke. Prva med njimi je uporabnik, ki želi dostopati do storitve oziroma jo uporabljati. Sledi ponudnik storitve (angl. *network access server*), ki nudi uporabnikom svoje storitve in je hkrati tudi odjemalec strežnika RADIUS. Odjemalec deluje kot posrednik med uporabnikom in strežnikom RADIUS, saj posreduje podatke med njima oziroma jih pretvarja v primerno obliko. Z odgovorom strežnika pridobi odjemalec tudi podatke, s katerimi določi uporabniku nekatere omejitve (gre za proces avtorizacije). Primer omejitve je lahko omejitev trajanja seje

uporabnika ali omejitev hitrosti povezave. Ker je protokol RADIUS brezstanjski (angl. *stateless*), strežnik RADIUS ne more vedeti, ali je odjemalec določil priporočene omejitve uporabniku. Zadnja stranka, ki je vključena v arhitekturo protokola, pa je strežnik RADIUS. Strežnik RADIUS je lahko tudi le vmesni člen pri dostopu do drugega strežnika RADIUS (angl. *proxy*) in odloča, ali je določenemu uporabniku odobrena uporaba storitve ali ne.

Komunikacija med ponudnikom storitve in strežnikom RADIUS je koračna in temelji na podlagi UDP protokola na transportni plasti, kar pomeni, da lahko pride do izgube paketov. V tem primeru je treba paket ponovno poslati, kar pa je naloga omogočenih RADIUS naprav oziroma temu namenjenega algoritma, ne pa samega protokola [18]. Komunikacija se prične z dostopom uporabnika do storitve, ki posreduje uporabnikove podatke naprej do strežnika RADIUS in na podlagi strežnikovega odgovora ustrezno ukrepa. Ko strežnik prejme posredovane podatke uporabnika, ta uporabnika avtenticira in to sporoči odjemalcu ter pošlje potrebne informacije za odjemalca. Te informacije se lahko nanašajo na pravice, ki jih ima določen uporabnik pri uporabi te storitve (avtorizacija). Spodnja slika 7 prikazuje primer komunikacije med uporabnikom, ponudnikom storitve oziroma odjemalcem in strežnikom RADIUS.



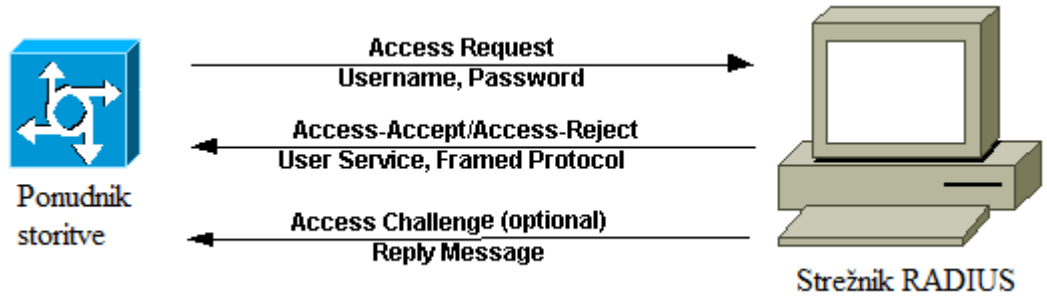
Slika 7: Primer komunikacije pri protokolu RADIUS [18].

Kot je razvidno z zgornje slike 7 in je bilo pred tem omenjeno, prične komunikacijo uporabnik. Ta pošlje zahtevo do ponudnika storitve z namenom pridobitve dostopa do storitve, ki jo ta ponudnik storitve nudi. To zahtevo uporabnik sproži največkrat neposredno na povezavni plasti s pomočjo avtentikacijskega protokola PPP. V nekaterih primerih se komunikacija med uporabnikom in ponudnikom storitve prične na višjih plasteh, kot je npr. z uporabo protokola HTTPS. Nato ponudnik storitve zahteva od uporabnika uporabniško ime in geslo, če gre za avtentikacijski protokol PAP, oziroma zahteva rešitev izziva (angl. *challenge*), če gre za avtentikacijski protokol CHAP [18]. Uporabnik lahko namesto avtentikacijskega protokola PAP oziroma CHAP uporabi tudi MS-CHAP ali EAP, saj ju strežnik RADIUS prav tako podpira. Izbira avtentikacijskega protokola je poljubna, vendar

mora tako odjemalec kot tudi strežnik podpirati izbrani protokol. Ko uporabnik pošlje zahtevane podatke ponudniku storitve, ta ustvari in pošlje zahtevo tipa `Access-Request` na strežnik RADIUS.

Ta zahteva v primeru avtentikacijskega protokola PAP navadno vsebuje uporabniško ime, šifrirano uporabniško geslo, naslov IP ponudnika storitve ter številko vrat (angl. *port*). Zahteva pa lahko vsebuje tudi nekatere dodatne podatke, ki so ponudniku storitve znane o uporabniku (omrežni naslov uporabnika, telefonska številka, itd.) [18, 52]. Strežnik nato preveri pravilnost podatkov uporabnika tako, da razišče, ali se uporabniško ime nahaja v njegovi podatkovni bazi. Če se nahaja, mora seveda preveriti tudi pravilnost podanega gesla. Današnje moderne implementacije strežnika RADIUS lahko preverijo pravilnost podatkov uporabnika s pomočjo zunanjih virov. Primer zunanjega vira je lahko podatkovna baza MySQL, imeniška storitev LDAP, sistem Kerberos in Active Directory. Ti zunanji viri lahko namreč vsebujejo prave podatke za določenega uporabnika ter jih nato strežnik RADIUS preveri z dostopom do zunanjega vira. Ko strežnik preveri pravilnost podatkov, odgovori odjemalcu z enim izmed treh tipov odgovorov (`Access-Accept`, `Access-Reject`, `Access-Challenge`), ta pa nato posreduje prejeti odgovor uporabniku. Vsak izmed treh možnih odgovorov lahko vsebuje atribut `Reply-Messege`. Ta atribut lahko v primeru odgovora tipa `Access-Reject` vsebuje obrazložitev, zakaj uporabniku ni odobren dostop do storitve. Pri tipu odgovora `Access-Challenge` lahko atribut vsebuje poziv k rešitvi zastavljenega izziva, ali če gre za odgovor tipa `Access-Accept`, ta vsebuje sporočilo, ki uporabnika seznani o dovoljeni uporabi storitve. Te vrednosti atributa `Reply-Messege` se posredujejo uporabniku prek ponudnika storitve.

Kadar strežnik RADIUS odgovori odjemalcu s tipom odgovora `Access-Accept`, kar pomeni, da je uporabniku dovoljena uporaba storitve, ta odgovor vsebuje tudi seznam določenih prilastkov. Ti prilastki opišejo oziroma podajo podatke, ki se nato uporabijo pri tej uporabniški seji, kot je na primer podatek o tipu storitve ali naslovu IP [18]. Odgovor strežnika RADIUS poleg tega vsebuje tudi informacijo o pravicah oziroma omejitvah, ki so uporabniku dodeljene. Te pravice oziroma omejitve so prav tako podane v obliki prilastkov. Na ta način ponudnik storitve nudi uporabniku uporabo storitve z morebitnimi omejitvami oziroma določenimi pravicami. Spodnja slika 8 prikazuje komunikacijo med ponudnikom storitve oziroma odjemalcem in strežnikom RADIUS, s prikazom sporočil protokola RADIUS in vsebovanih prilastkov.



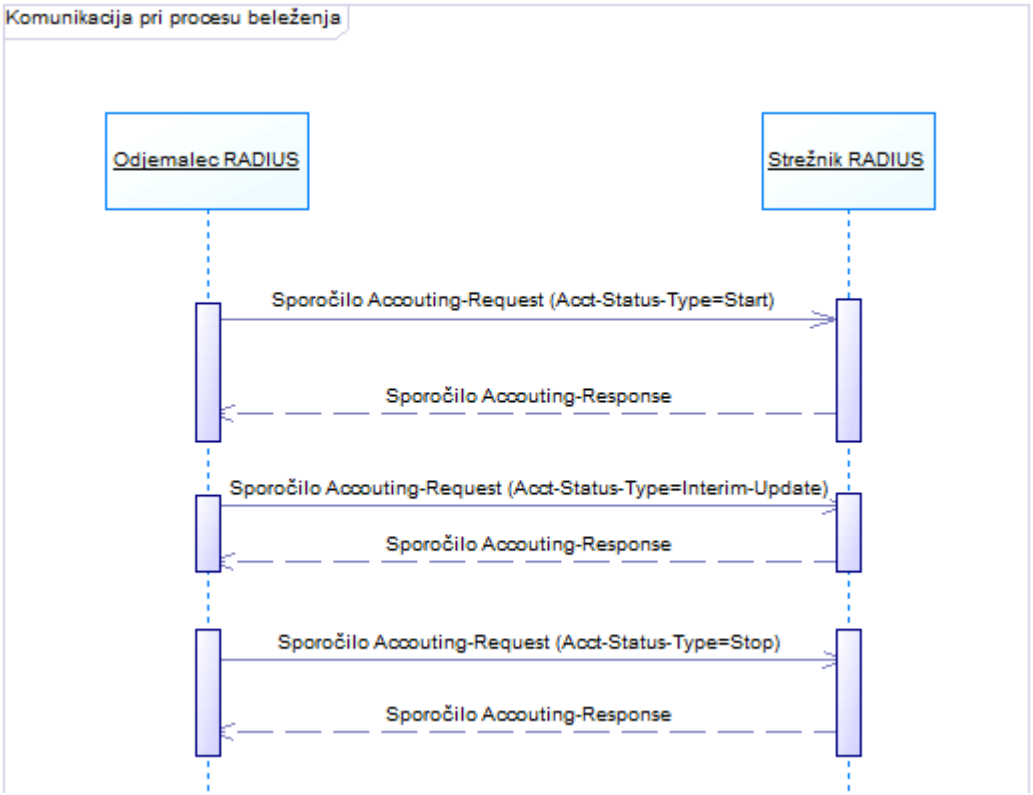
Slika 8: Primer komunikacije med odjemalcem in strežnikom RADIUS pri procesu avtentikacije oziroma avtorizacije [18].

2.3.4 Komunikacija pri procesu beleženja

Proces beleženja se lahko pri protokolu RADIUS uporablja neodvisno od procesa avtentikacije oziroma avtorizacije. Sporočila, namenjena beleženju, se pošiljajo ob začetku in koncu uporabniške seje ter velikokrat tudi med časom trajanja seje. Tako pri procesu beleženja beležimo tri vrste dogodkov. Sporočila pri procesu beleženja vsebujejo številne prilastke, s pomočjo katerih posredujemo določene podatke, ki so potrebni v korist beleženja. Strežnik RADIUS za potrebe beleženja posluša na vratih (angl. *port*) 1813, kar nam dodatno navaja, da je proces beleženja povsem ločen od procesa avtentikacije oziroma avtorizacije, saj v okviru procesa avtentikacije oziroma avtorizacije strežnik RADIUS posluša na vratih 1812.

Ko se prične uporabniška seja oziroma ko strežnik RADIUS odobri uporabniku uporabo določene storitve, pošlje ponudnik te storitve (angl. *network access server*) zahtevo tipa `Accouting-Request` na strežnik RADIUS. Ta zahteva je prvo sporočilo, poslano po uspešni avtentikaciji uporabnika [15, 52]. Zahteva vsebuje več potrebnih prilastkov, med katerimi je pomembna vrednost prilastka `Acct-Status-Type`, ki je v tem primeru `Start`, saj se zahteva pošilja na strežnik RADIUS z namenom pričetka beleženja. Strežnik RADIUS nato potrdi sprejem zahteve s pošiljanjem odgovora ponudniku storitve. Tip odgovora, ki ga pošlje, je sporočilo `Accouting-Response` (če strežnik RADIUS ne odgovori v določenem času, se zahteva pošlje ponovno). V času trajanja uporabniške seje lahko ponudnik storitve ponovno pošlje zahtevo tipa `Accouting-Request` na strežnik RADIUS, vendar ta vsebuje vrednost `Interim-update` v prilastku `Acct-Status-Type`. Ta ponovna zahteva sporoča strežniku trenuten čas trajanja uporabnikove seje ter njegovo trenutno porabo podatkov. S to zahtevo ponudnik storitve strežniku tudi sporoča, da uporabnik še vedno uporablja storitev. Ko uporabnik zaključi svojo sejo oziroma preneha uporabljati storitev, ponudnik storitve obvesti strežnik RADIUS s pošiljanjem zahteve tipa

`Accoting-Request`, ki vsebuje vrednost `Stop` v prilastku `Acct-Status-Type`. S to zahtevo strežnik preneha s procesom beleženja, sama zahteva pa vsebuje končne podatke o času trajanja seje in porabi podatkov. Vsi ti podatki, poleg ostalih, se seveda posredujejo s pomočjo prilastkov. Primer komunikacije pri procesu beleženja med odjemalcem in strežnikom RADIUS prikazuje tudi spodnja slika 9.



Slika 9: Primer komunikacije pri procesu beleženja.

Če strežnik RADIUS [15] ne deluje ali se ta ne odziva, odjemalec oziroma ponudnik storitve ponovno pošlje zahtevo ali jo posreduje drugemu strežniku RADIUS. Ali se odjemalec odloči za ponovno pošiljanje zahteve ali za pošiljanje zahteve drugemu strežniku, je odvisno od konfiguracije odjemalca.

Kadar je strežnik RADIUS v vlogi medstrežnika (angl. *proxy*) [15], ki posreduje zahteve naprej, velikokrat shrani podatke beleženja lokalno pri sebi ter jih nato posreduje naprej. Strežniku RADIUS v vlogi medstrežnika smo se posvetili v naslednjem podpoglavju.

2.3.5 RADIUS v vlogi medstrežnika in gostovanja

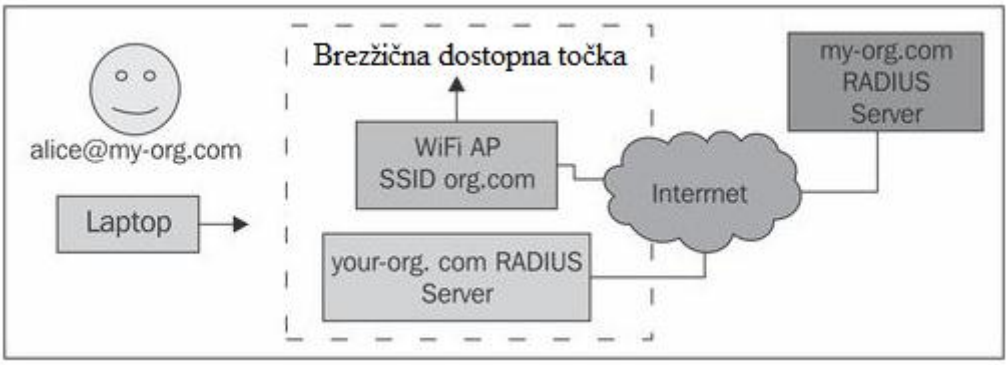
Ko strežnik RADIUS nastopi v vlogi medstrežnika (angl. *proxy*), ta deluje kot odjemalec drugega strežnika RADIUS, kar pomeni, da gre za komunikacijo med strežniki RADIUS. Medstrežnik RADIUS tako posreduje zahtevo pravemu oziroma matičnemu strežniku RADIUS ter jo pred posredovanjem lahko preoblikuje. Medstrežnik to sporočilo šifrira pred pošiljanjem matičnemu strežniku, ta pa mu vrne šifriran odgovor. Povezave med strežniki RADIUS so lahko varne, in sicer s pomočjo navideznega zasebnega omrežja (angl. *virtual private network*). Medstrežnikov RADIUS je lahko več, v tem primeru tvorijo verigo strežnikov RADIUS.

Ko govorimo o posredovanju sporočil med strežniki RADIUS, govorimo o nastanku področij (angl. *realms*). Tu gre za razdelitev uporabnikov na posamezna področja oziroma sfere, kjer vsako področje vsebuje svoj strežnik RADIUS [15]. Običajno je določeno področje definirano s poljubnim nizom, ki je podoben imenu domene. Področja se uporabljajo za grupiranje uporabnikov v posamezne skupine, kjer so ti uporabniki ločeni znotraj istega področja s pomočjo uporabniškega imena. Uporabniško ime je ločeno od področja z uporabo ločilnega znaka. Področje je lahko zapisano v prefiksni ali postfiksni notaciji, vendar se pogosto uporablja postiksna notacija z uporabo ločilnega znaka @ (`alice@freeradius.org`), ko govorimo o uporabnikih operacijskega sistema Linux. Pri uporabnikih operacijskega sistema Windows pa se pogosto pojavi prefiksna notacija. Ta oblika notacije, za razliko od postfiksne, vsebuje najprej zapis področja in nato uporabniško ime, ki je od področja ločen z znakom \ (`my_domain\alice`). Strežniki RADIUS omogočajo za uporabo ločilnega znaka katerikoli znak, čeprav se največkrat pojavi znak @ ali /. Prav tako je lahko ime področja poljubno in ni obvezno, da je ime področja enako, kot je ime domene.

Ko medstrežnik RADIUS prejme zahtevo, ki vsebuje uporabniško ime in njegovo področje, ta izvrši zahtevo ali jo posreduje drugemu strežniku RADIUS. Če zahtevo izvrši sam, pomeni, da strežnik obravnava področje, v katerem se nahaja uporabnik oziroma je strežnik matični za to področje. Kadar [15] strežnik posreduje zahtevo drugemu strežniku, pa pomeni, da sam ne obravnava področja, v katerem se nahaja uporabnik in tako posreduje zahtevo tistemu strežniku RADIUS, ki obravnava to področje oziroma jo posreduje matičnemu strežniku RADIUS za to področje. Seveda v primeru posredovanja zahteve strežnik RADIUS najprej preveri, ali pozna določeno področje, saj lahko le tako zahtevo posreduje matičnemu strežniku, ki obravnava to področje.

Takšno posredovanje zahtev med strežniki RADIUS se izvaja velikokrat za potrebe gostovanja (angl. *roaming*). Gostovanje omogoča uporabniku dostop do medmrežja (angl.

internet) na različnih lokacijah, in sicer z enakimi podatki o svoji identiteti. Tu lahko uporabnik z drugega področja uporablja storitev oziroma pridobi dostop do medmrežja v drugem oziroma gostujočem področju. Ponudnik storitve (angl. *network access server*) mu namreč prek strežnika RADIUS dovoli gostovanje na svojem področju. Na ta način pride do vzpostavitve sodelovanja med področji in do avtentikacije na drugo področje [15]. Temu tipu gostovanja pravimo sporazum med dvema organizacijama ter se uporablja v brezžičnem omrežju EDUROAM. Spodnja slika 10 prikazuje ta tip gostovanja, kjer nastopa sporazum med organizacijo `your-org.com` in `my-org.com`.



Slika 10: Primer gostovanja pri protokolu RADIUS [15].

Ko uporabnik želi (`alice@my-org.com`) z zgornje slike 10 dostopati do medmrežja znotraj področja `your-org.com`, čeprav pripada področju `my-org.com`, se ta poveže na brezžično dostopno točko (angl. *Wi-Fi access point*) znotraj področja `your-org.com` [15]. Brezžična dostopna točka je v našem primeru vidna pod imenom `org.com` (angl. *service set identifier*) in je pod tem imenom vidna tudi znotraj področja `my-org.com`. Nato brezžična dostopna točka posreduje to avtentikacijsko zahtevo uporabnika strežniku RADIUS znotraj področja `your-org.com`. Strežnik RADIUS ugotovi, da uporabnik (`alice@my-org.com`) pripada področju `my-org.com` in zato posreduje to zahtevo do strežnika RADIUS na področju `my-org.com`. Na ta način postane strežnik RADIUS na področju `your-org.com` odjemalec strežnika RADIUS na področju `my-org.com`. Nato matični strežnik RADIUS za področje `my-org.com` preveri pravilnost poslanih podatkov uporabnika in to sporoči nazaj strežniku RADIUS na področju `your-org.com`. Ta seveda sporočilo posreduje do brezžične dostopne točke, ki v primeru, da strežnik odgovori z odgovorom tipa `Access-Accept`, odobri uporabniku dostop do medmrežja.

2.3.6 Varnost

Za zagotavljanje integritete in avtentičnosti posameznega poslanega sporočila služi polje `authenticator` v sporočilu protokola RADIUS (to je zagotovljeno tudi s pomočjo prilastka `Message-Authenticator`, če ta nastopa). Tu gre namreč za podpis poslanega sporočila, ki ga imenujemo `authenticator`. Vsebina polja `authenticator` je izračunana s pomočjo različnih atributov, kjer so ti izbrani v odvisnosti od tipa sporočila. Pri tipu sporočila `Access-Request` se vsebina polja `authenticator` izračuna s pomočjo drugih atributov, kot je to pri paketu tipa `Accouting-Request`, kar ni povsem varno, ko govorimo o napadih. Prav tako se izračun vrednosti polja razlikuje, če gre za zahtevo odjemalca ali odgovor strežnika. Pomemben atribut, ki se uporabi pri izračunu vrednosti polja `authenticator` in vzpostavi zaupanje med odjemalcem in strežnikom RADIUS, je skupna skrivnost (angl. *shared secret*), ki smo jo podrobneje predstavili v naslednjem odstavku in v nadaljevanju podpoglavja. Poleg tega protokol RADIUS nudi zakrivanje oziroma šifriranje prilastkov, kot je standardiziran prilastek `User-Password`. (Poleg standardiziranih prilastkov nudi tudi šifriranje za posebne prilastke, znane pod kratico VSA.) [1]. Sam protokol RADIUS nima zaščite pred prisluškovanjem, kar pomeni, da komunikacija ni zakrita med odjemalcem in strežnikom RADIUS. Prav tako protokol nima podpore za zaščito napada s ponavljanjem (angl. *replay attack*). Ti dve pomanjkljivosti se lahko odpravita z uporabo protokola RADIUS prek protokola IPsec. V tem primeru je namreč poskrbljeno za integriteto sporočil, avtentikacijo, zaupnost in zaščito pred napadom s ponovitvijo za vsa sporočila, ki se pojavijo pri procesu avtentikacije in beleženja.

Skupna skrivnost je sestavljen niz, ki služi kot geslo med odjemalcem in strežnikom RADIUS ter je znana samo njima. Tu je treba omeniti, da se skupna skrivnost v primeru medstrežnika (angl. *proxy*) nahaja tudi med odjemalcem in medstrežnikom RADIUS ter med medstrežnikom in strežnikom RADIUS [9]. Vrednost skupne skrivnosti je enaka med določenim parom odjemalca in strežnika ter drugačna od drugih parov odjemalca in strežnika, kar pomeni, da je skupna skrivnost enolična oziroma enaka samo med določenim parom odjemalca in strežnika. Skupna skrivnost se uporablja z namenom povečanja varnosti, saj pripada le določenemu paru odjemalca in strežnika ter vpliva na rezultat pri izračunu polja `authenticator`. Ker se namreč skupna skrivnost uporabi pri izračunu polja `authenticator`, je pravilen rezultat odvisen od skupne skrivnosti med določenim odjemalcem in strežnikom.

Skupna skrivnost se uporablja tudi za preverjanje sporočil v smislu, ali je bilo sporočilo med prenosom spremenjeno (integriteta sporočila) ter tudi za šifriranje določenih prilastkov v sporočilu, kot je prilastek `User-Password`. V primeru prilastka `User-Password` se

njegova vrednost šifrira, čeprav se hkrati uporabi skupna skrivnost s pomočjo polja `Authenticator` in zgoščevalne funkcije MD5.

Izjemo predstavlja sporočilo tipa `Access-Request`, za katerega ni preverjanja ali je poslano od odjemalca, ki ga strežnik RADIUS pozna oziroma ali ima ta enako vrednost skupne skrivnosti kot strežnik RADIUS. V tem primeru se v poslanem sporočilu uporabi prilastek `Message-Authenticator` oziroma se njegovo uporabo omogoči ter se na ta način zagotovi to preverjanje. V osnovi se to preverjanje zgodi s pomočjo naslova IP. Ob prejeti zahtevi `Access-Request` namreč strežnik RADIUS preveri, ali se izvorni naslov IP, od katerega je bilo poslano sporočilo, ujema s katerim izmed naslovov IP, ki so mu poznani. Tako ima vsak odjemalec pri strežniku RADIUS zabeležen svoj naslov IP in na ta način strežnik preveri, ali lahko odjemalcu dovoli dostop do njega. Ta način pa ni povsem varen, saj se lahko naslovi IP nadomestijo z drugimi, kar bi privedlo do dostopa odjemalca do strežnika RADIUS, čeprav mu ta ni dovoljen [12, 28, 56]. Zaradi te nevarnosti se uporablja omenjeni prilastek `Message-Authenticator`, ki vsako sporočilo dodatno zavaruje. Vrednost prilastka se izračuna s pomočjo zgoščevalne funkcije MD5 na podlagi celotnega sporočila, ki uporablja kot ključ skupno skrivnost. Strežnik RADIUS preveri vrednost prilastka `Message-Authenticator`, če je ta priložen in v primeru, da njegova vrednost ni prava, sporočilo zavrže. Prav tako v primeru, ko strežnik RADIUS zahteva prisotnost prilastka `Message-Authenticator` in ta ni podan, preprosto zavrže odjemalčevo sporočilo. Navadno je prisotnost prilastka obvezna samo v primeru uporabe avtentikacijskega protokola EAP, vendar je uporaba prilastka priporočljiva tudi v takrat, ko gre za ostale avtentikacijske protokole.

Vrednost skupne skrivnosti se lahko, da bi povečali varnost, izračuna večkrat v naključnem času. Njena dolžina je omejena, saj mora biti večja od 0, vendar je priporočena dolžina vsaj 22 oziroma več znakov [9]. Skupno skrivnost dolžine več kot 22 znakov je skoraj nemogoče napasti z metodo grobe sile (angl. *brute force attack*). Poleg tega je priporočljivo, da je skupna skrivnost sestavljena naključno s pomočjo zaporednih različnih znakov, da bi zavarovali tako odjemalca kot tudi strežnik pred napadom s slovarjem (angl. *dictionary attack*). Znaki, ki jih lahko vsebuje skupna skrivnost, so črke (velike in majhne), številke in določeni simboli (klicaj, zvezdica, dvopičje, itd.) [28]. Če imamo skupno skrivnost sestavljeno iz več različnih znakov, so varnejši tudi prilastki, kot je `User-Password`, ki so šifrirani z njeno vrednostjo. Vrednost skupne skrivnosti se lahko spreminja avtomatično, vendar poleg te prednosti nastopi tudi nevarnost, saj ni zagotovljeno, da bosta odjemalec in strežnik RADIUS začela uporabljati novo skupno skrivnost oziroma da se bosta sinhronizirala v primernem času.

Ko govorimo o varnosti protokola RADIUS, je treba omeniti, da protokol uporablja na transportni plasti UDP prenosni protokol. Uporaba protokola UDP na transportni plasti ni povsem varna, vendar obstajajo določeni razlogi, zakaj protokol RADIUS uporablja protokol UDP namesto protokola TCP, ki je sicer zanesljivejši. Protokol UDP je bil izbran predvsem zaradi določenih lastnosti, ki jih ima protokol RADIUS, saj so te lastnosti značilnost protokola UDP. Razlogi za uporabo prenosnega protokola UDP so naslednji [13]:

- lastnost ponovnega pošiljanja transportnega protokola TCP ni zahtevana, prav tako tudi ni zahtevana uporaba potrditve poslanih sporočil. Uporabnik namreč ni pripravljen čakati več minut za končanje procesa avtentikacije oziroma do prejetja odgovora, zato v tem primeru nima prednosti uporaba zanesljive dostave podatkov, saj lahko uporabnik v tem primeru čaka odgovor dlje časa,
- protokol RADIUS je brezstanjski (angl. *stateless*) kakor tudi transportni protokol UDP. Protokol RADIUS tako poenostavi uporabo protokola UDP,
- k izbiri protokola UDP veliko pripomore tudi preprostost uporabe protokola UDP, saj tako protokol UDP poenostavi implementacijo strežnika RADIUS.

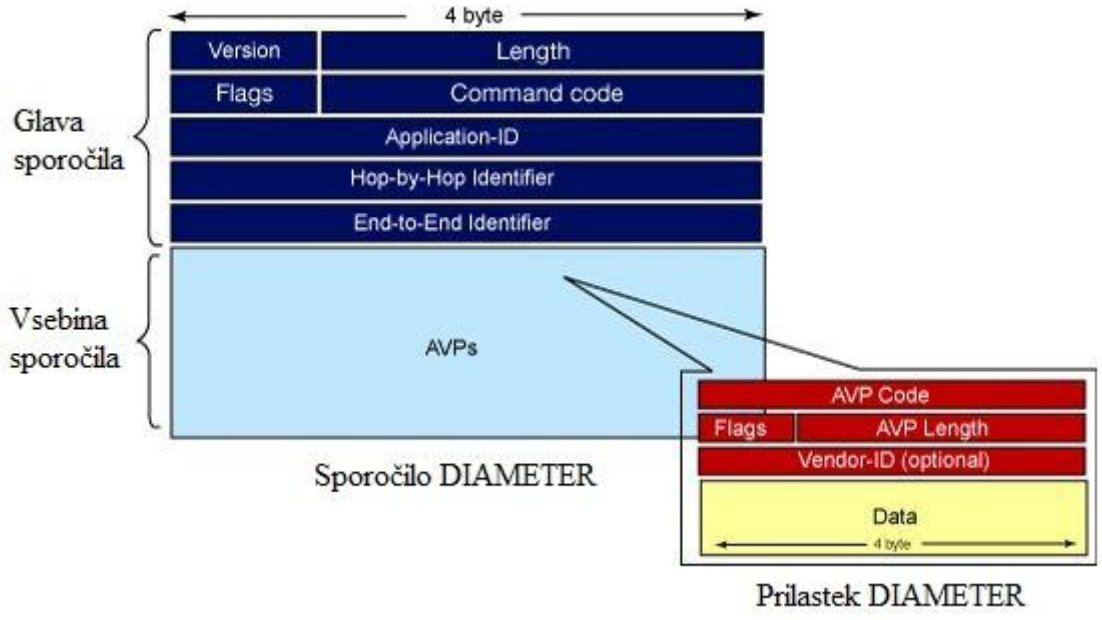
Edina slabost transportnega protokola UDP je v tem, da morajo razvijalci razviti lasten mehanizem oziroma algoritem, ki izvaja ponovno pošiljanje sporočil po določenem času ter ga morajo tudi upravljati [13]. Ta algoritem je namreč že vgrajen v transportni protokol TCP, ki poskrbi za izgubljena sporočila. Vendar pa je glede na prednosti, ki jih prinaša protokol UDP pri protokolu RADIUS, ta še vedno boljša izbira kot uporaba prenosnega protokola TCP.

2.3.7 Primerjava s protokolom DIAMETER

Kljub priljubljenosti in razpoložljivosti protokola RADIUS ima ta nekaj omejitev, katerim se je bilo treba posvetiti. Aplikacije, ki uporabljajo protokol RADIUS, so omejene v smislu zagotavljanja varnejšega in zanesljivejšega procesa [24, 29]. Zaradi tega je bil razvit protokol DIAMETER (določen v dokumentu RFC 3588), ki ga veliko uporabljajo modernejše aplikacije, kot so aplikacije za dostop do omrežja in za IP telefonijo. Protokol predstavlja naslednika protokola RADIUS in je bil zasnovan kot izboljšana inačica protokola RADIUS v različnih vidikih. Protokol DIAMETER je bil razvit, da bi odpravili omejitve protokola RADIUS in izpolnili nove zahteve za protokole tipa AAA. Z uvedbo novih tehnologij (brezžični dostop do medmrežja (angl. *internet*), IP telefonija) in rastjo medmrežja so postali usmerjevalniki (angl. *routers*) kompleksnejši, kar je privedlo do novih zahtev za protokole tipa AAA. Sam protokol služi istemu namenu kot protokol RADIUS, saj gre za isti tip

protokola, vendar vsebuje naprednejše oziroma dodane lastnosti, ki privedejo do zanesljivosti protokola (zanesljiva dostava sporočila, obvestila o napakah, itd.). Protokol DIAMETER ni povsem skladen s protokolom RADIUS, vendar predstavlja njegovo nadgradnjo predvsem z varnostnega vidika. Zaradi neskladnosti s protokolom RADIUS se morajo aplikacije, ki so uporabljale protokol RADIUS, prilagoditi spremembam, katere prinaša protokol DIAMETER. Aplikacije, ki uporabljajo protokol DIAMETER, imajo možnost definirati svoje lastne razširitve in tako izkoristiti zmogljivosti protokola.

Seveda se format sporočila protokola DIAMETER razlikuje od formata sporočila protokola RADIUS. Prav tako se razlikuje format vsebovanih prilastkov. Spodnja slika 11 prikazuje format sporočila in prilastka pri protokolu DIAMETER.



Slika 11: Format sporočila in prilastka protokola DIAMETER [24].

Kot je razvidno z zgornje slike 11, so pri protokolu DIAMETER prisotna nova polja v samem sporočilu. Nova polja v sporočilu protokola DIAMETER so naslednja [3, 26]:

- polje `Version`, ki označuje inačico protokola DIAMETER,
- polje `Flags` dolžine 8 bitov, ki označuje tip sporočila. S pomočjo ustreznih zastavic (R, P, E in T) v polju prepoznamo tip sporočila,
- polje `Command code`, ki se uporablja za prepoznavo ukazov protokola DIAMETER in vsebuje ukaze, povezane s sporočilom. Dolžina samega polja znaša 3 zloge (angl. *octets*),

- polje `Aplication-ID`, ki se uporablja za prepoznavo aplikacij protokola DIAMETER oziroma nam poda oznako aplikacije, za katero je sporočilo namenjeno. Njegova dolžina znaša 4 zloge,
- polje `End-to-end-identifier`, ki se uporablja za zaznavo sporočil, ki so se ponovila, oziroma služi za zaznavo duplikatov. Polje je 4 zloge dolgo oziroma ima 32 bitov.

Poleg novih polj nastopijo tudi polja, ki imajo podobno vlogo kot pri protokolu RADIUS, le da nekatera niso poimenovana enako [3, 26]:

- Polje `Hop-by-Hop-Identifier` ima podobo vlogo kot polje `Identifier` pri sporočilih protokola RADIUS, saj povezuje odgovore z zahtevami. Dolžina polja znaša 4 zloge, medtem ko znaša dolžina polja pri sporočilih protokola RADIUS 1 zlog. Vrednost polja `Hop-by-Hop-Identifier` v poslani zahtevi mora biti enaka vrednosti tega polje v poslanem odgovoru, kar pomeni, da gre za odgovor na poslano zahtevo,
- Polje `Length` vsebuje celotno dolžino sporočila DIAMETER skupaj z glavo kot pri sporočilih protokola RADIUS, vendar dolžina polja pri slednjem znaša 3 zloge,
- Polje `AVPs`, ki je pri sporočilih protokola RADIUS znan pod imenom `Attributes`, ima prav tako podobno vlogo kot pri sporočilu protokola RADIUS. Polje namreč vsebuje prilastke, ki kot vrednost vsebujejo vse potrebne podatke, ki se prenašajo s pomočjo protokola DIAMETER.

Format prilastka pri protokolu DIAMETER prav tako vsebuje dodatna polja v primerjavi s prilastki pri protokolu RADIUS, kot je to razvidno z zgornje slike 11. Vsebovana sta dva nova polja, imenovana `Flags` in `Vendor-ID`, medtem ko imajo ostala polja podoben pomen kot pri prilastkih protokola RADIUS.

Polje `Flags` obvešča prejemnika o tem [3, 26], kako mora biti vsak izmed prilastkov obdelan. Vrednosti tega polja so nastavljeni biti oziroma zastavice (V, M, P), kjer ima vsak nastavljen bit svoj namen (v tem polju se nahajajo tudi rezervirani biti). Če je v polju nastavljena zastavica V, ta označuje prisotnost naslednjega novega polja `Vendor-ID`, katerega dolžina znaša 4 zloge, vendar njegova prisotnost ni obvezna. Ob prisotnosti tega polja je določeno, da je prilastek implementiran na novo (v nasprotnem primeru prilastek ni implementiran na novo). Vsak posameznik (angl. *vendor*), ki želi implementirati nov prilastek, mora uporabiti svojo lastno vrednost polja `Vendor-ID`, s katero je prilastek enolično določen. Preostala polja `AVP Code`, `AVP Length` in `Data` imajo podobno vlogo kot pri prilastkih protokola RADIUS. Polje `AVP Code`, skupaj s poljem `Vendor-ID`

enolično določa prilastek. Polje `AVP Length` označuje dolžino celotnega prilastka vključno z glavo ter tudi zajema morebitno prisotnost polja `Vendor-ID`. Njegova dolžina za razliko od prilastkov protokola RADIUS znaša 3 zloge. Preostalo polje `Data` ima podobno vlogo kot polje `Value` pri prilastkih protokola RADIUS, saj ta vsebuje dejanske vrednosti prilastka oziroma podatke, le da pri slednjem lahko polje kot vrednost vsebuje dve vrsti oziroma dva tipa podatkov. Prvi tip podatkov, imenovan `Basic AVP data formats`, vsebuje splošne podatkovne tipe, ki lahko nastopijo kot vrednost prilastka (`OctetString`, `Integer32`, `Integer64`, `Unsigned32`, `Unsigned64`, `Float32`, `Float64`, `Grouped`). Poleg osnovnih podatkovnih tipov lahko aplikacije definirajo svoje podatkovne tipe, ki spadajo pod drug tip podatkov, imenovan `Derived AVP formats` ter izvirajo iz osnovnih podatkovnih tipov.

V spodnji tabeli 4 smo prikazali še pomembne splošne razlike oziroma prednosti protokola DIAMETER pred protokolom RADIUS [3, 24].

Lastnosti	**Protokol DIAMETER**	**Protokol RADIUS**
Protokol na transportni plasti	TCP ali SCTP, ki zagotavljata zanesljiv in varen prenos sporočil.	UDP, ki povzroča velik problem, predvsem pri procesu beleženja.
Omrežna varnost	Uporaba protokola IPsec ali protokola TLS.	Možna uporaba protokola IPsec.
Podpora agentov	Podpira agente, kot so: Relay, Proxy, Redirect in Translation.	Ni izrecne podpore, agenti so lahko implementirani le v okviru strežnika RADIUS.
Velikost podatkovnega polja za posamezen prilastek (angl. *size of attribute data*)	Preko 16,000,000 zlogov, k čemer pripomore daljše polje `Length` v prilastku.	255 zlogov.
Odkrivanje odjemalcev (angl. *peer discovery*)	Poleg statične konfiguracije omogoča tudi dinamično iskanje.	Statična konfiguracija.
Podpora gostovanja (angl. *roaming*)	Nudi varno gostovanje.	Podpira, vendar je ranljiva za napade.

Tabela 4: Splošna primerjava med protokolom RADIUS in DIAMETER.

Kot je razvidno iz zgornje tabele 4, ima protokol DIAMETER številne prednosti v primerjavi s protokolom RADIUS. Pri protokolu DIAMETER se tako lahko verjame, da bo nadomestil protokol RADIUS oziroma postal naslednji standard pri izbiri protokola tipa AAA [15].

Čeprav ima protokol DIAMETER številne prednosti, protokol RADIUS še vedno ostaja v široki uporabi oziroma ostaja standard pri izbiri protokola tipa AAA. Glavni razlog tiči v tem, da so številne izboljšave, ki jih prinaša protokol DIAMETER, že zajete v okviru več razširitev protokola RADIUS. Tak primer razširitve je lahko protokol Radsec, ki omogoča prenos sporočil s pomočjo zanesljivega transportnega protokola TCP ter prav tako omogoča prenos z uporabo varnostnega protokola TLS.

2.4 Protokol LDAP

Protokol LDAP je zasnovan kot omrežni protokol za dostop do obstoječih imeniških storitev in omogoča upravljanje z njimi s pomočjo določenih ukazov. LDAP se nanaša tudi na tip imeniške storitve, kar pomeni, da gre za primer imeniške storitve. Vendar se v osnovi LDAP nanaša le na protokol, ki zgolj zagotavlja dostop do različnih tipov podatkov v imeniških storitvah ter ne določa načina shranjevanja podatkov v imeniški storitvi oziroma na strežniku LDAP. Protokol LDAP je protokol, ki temelji na sporočilih, posredovanih med odjemalcem in strežnikom, in je opisan v številnih dokumentih RFC (4510-4519). Gre za asinhroni protokol, kar pomeni [4], da lahko v primeru več odjemalčevih zahtev odgovori strežnika na zahteve prihajajo v nepričakovanem vrstnem redu: poznejše poslane zahteve odjemalca lahko pridobijo odgovor pred zahtevami, ki so bile poslane med prvimi. Imeniška storitev LDAP je tesno povezana s tipom imeniške storitve, imenovane X.500, ki definira imenski prostor, v katerem se nahajajo predmeti oziroma posamezni podatki, ter med njima veljajo številne podobnosti. Cilj implementacije protokola LDAP je bil v tem [25], da se zamenja obstoječi protokol DAP, ki je bil namenjen izključno dostopu do imeniške storitve X.500 in s tem prehod na protokolarni sklad TCP/IP. Pri imeniški storitvi X.500 je bila zahtevana uporaba protokolarnega sklada ISO/OSI v komunikaciji med odjemalcem in strežnikom.

Trenutna inačica protokola LDAP je inačica 3, medtem ko je prehodna inačica 2 že umaknjena iz uporabe. Inačica protokola LDAPv3 se razlikuje od inačice LDAPv2 predvsem na področju varnosti. Inačica protokola LDAPv3 definira številne mehanizme, potrebne za avtentikacijo uporabnika, med katerima sta prva dva že bila vsebovana v inačici LDAPv2 [4]:

- **anonimna avtentikacija** (angl. *anonymous authentication*),
- **preprosta avtentikacija** (angl. *simple authentication*),
- **preprosta avtentikacija prek protokola SSL/TLS** (angl. *simple authentication over SSL/TLS*),
- **preprosta avtentikacija in varnostna plast** (angl. *simple authentication and security layer*).

Imeniška storitev LDAP oziroma strežnik LDAP vsebuje oziroma vzdržuje številne podatke in zagotavlja storitve, s pomočjo katerih lahko dostopamo do vsebovanih podatkov. Strežnik LDAP lahko hrani oziroma vsebuje različne vrste podatkov o različnih vnosih, saj je bil strežnik LDAP zasnovan za splošen namen (angl. *general-purpose directory server*) [2].

Ti podatki so na strežniku LDAP urejeni hierarhično v obliki drevesa, kjer vsako vozlišče v hierarhiji predstavlja predmet, ki lahko vsebuje več atributov. Ta hierarhična predstavitev podatkov je znana pod kratico DIT [2]. Podatki so lahko tudi porazdeljeni med več strežnikov LDAP, kar pomeni, da se lahko posamezni deli drevesne strukture (angl. *directory information tree*) nahajajo na več strežnikih LDAP (v tem primeru gre za porazdeljeno imeniško storitev). Podrobnejši pregled predmetov, atributov in hierarhične predstavitve podatkov smo prikazali v podpoglavjih, ki sledijo naslednjemu.

2.4.1 Ukazi nad imeniško storitvijo LDAP

Imeniška storitev LDAP nudi številne ukaze oziroma funkcije, določene v dokumentu RFC 4511, s katerimi lahko izvajamo določene operacije nad vsebovanimi podatki. Ukazi, ki jih lahko odjemalec izvrši nad imeniško storitvijo LDAP, so naslednji [2]:

- **bind** – ukaz omogoča avtentikacijo odjemalca, ki želi dostopati do podatkov imeniške storitve LDAP. Odjemalec tako običajno posreduje geslo in njegovo razločevalno ime (angl. *distinguished name*), s katerim se predstavi strežniku LDAP. Strežnik LDAP lahko omogoča tudi neavtenticirano sejo, pri čemer odjemalcu ni treba podati njegovega razločevalnega imena in gesla,
- **unbind** – s pomočjo ukaza lahko prekinemo obstoječo komunikacijo oziroma sejo. Ko strežnik LDAP prejme ukaz, ta nemudoma prekine določeno sejo,
- **search** – nudi iskanje posameznih predmetov v podatkovni bazi imeniške storitve LDAP. Ob izvajanju ukaza `Search` je treba podati vrednosti štirih atributov (`Base DN`, `Scope`, `Attributes` in `Filter`), kjer ima vsak določen pomen pri iskanju predmetov. Z atributom `Base DN` posredujemo razločevalno ime podatkovne baze, medtem ko z atributom `Scope` povemo, kako globoko v drevesni strukturi bomo iskali predmete. Z vrednostjo atributa `Attributes` podamo atribute, ki jih želimo pridobiti o določenem predmetu in nato še z atributom `Filter` filtriramo rezultat s pomočjo logičnih operatorjev oziroma z določenimi znaki. Ukaz `Search` lahko uporabljamo tudi v neavtenticirani seji, vendar je lahko rezultat ukaza drugačen kot pri avtenticirani seji,

- **add** – ukaz omogoča dodajanje novih predmetov v podatkovno bazo. V tem primeru mora seveda odjemalec definirati nov predmet z določenim razločevalnim imenom in atributi ter ga nato dodati v podatkovno bazo,
- **modify** – ukaz v osnovi ponuja spreminjanje vrednosti atributov določenega predmeta v podatkovni bazi. Poleg tega lahko za določen predmet doda, zamenja ali izbriše atribute in celo kombinira omenjene operacije v isti zahtevi. Kadar zahtevamo spreminjanje več vrednosti atributov predmeta v eni zahtevi, je zahteva uspešna le, če se izvedejo vse zahtevane spremembe,
- **delete** – s pomočjo ukaza zbrišemo določen predmet iz podatkovne baze, in sicer s podanim razločevalnim imenom predmeta (ob izvršitvi ukaza se izbriše celoten predmet skupaj z vsemi vsebovanimi atributi). Izbrišemo lahko le predmete, ki nimajo svojega naslednika v drevesni strukturi (angl. *directory information tree*) oziroma le končne predmete v drevesni strukturi,
- **modify DN** – čeprav se spreminjanje razločevalnega imena predmeta izvaja redko, je ukaz `Modify DN` namenjen prav temu, saj obstajajo primeri, kjer je to potrebno. Ukaz omogoča spreminjanje razločevalnega imena ali le spremembo relativnega razločevalnega imena,
- **compare** – izvajanje ukaza `Compare` omogoča primerjavo vrednosti določenega atributa v določenem predmetu. Ukaz prejme kot atribut razločevalno ime predmeta in vrednost določenega atributa, ki ga vsebuje predmet. Strežnik LDAP nato preveri enakost vrednosti atributa in odgovori, ali se podana vrednost atributa ujema s tistim v njegovi podatkovni bazi. Ukaz je uporaben v primerih, ko primerjamo pravilnost gesla,
- **abandon** – ukaz omogoča odjemalcu, da zahteva od strežnika prekinitev izvajanja nedokončanega ukaza, ki ga je poslal. Tako lahko prekinemo iskanje ali primerjanje, ter morebitne popravke v podatkovni bazi,
- **extended** – ukaz zagotavlja implementacijo poljubnega dodatnega ukaza, kar pomeni, da gre tu za razširitev. Primer ukaza, ki je definiran z uporabo ukaza `Extended`, je ukaz `start TLS`. Ukaz `start TLS` se uporablja za preklop na SSL/TLS način komunikacije oziroma na preklop varne komunikacije.

Treba je omeniti, da ukazi, kot so: `Add`, `Modify`, `Modify DN`, `Compare` in `Delete`, zahtevajo avtenticirano sejo (kadar je omogočena neavtenticira seja), saj jih drugače ni možno uporabljati nad podatkovno bazo strežnika LDAP. Tako mora odjemalec najprej uporabiti ukaz `Bind` z namenom avtentikacije.

2.4.2 Sheme, razredi in atributi

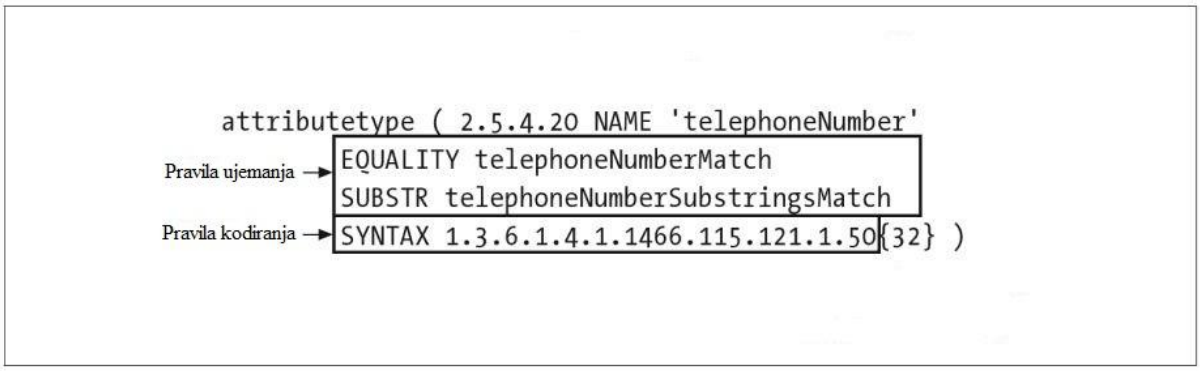
Vsak predmet v podatkovni bazi imeniške storitve LDAP vsebuje določene atribute, ki hranijo podatke oziroma informacije o samem predmetu. Vsak atribut ima določen tip, s pomočjo katerega je določen tip vrednosti, ki jo lahko atribut vsebuje (niz, število, itd.). Tip atributa pa je pomemben tudi v primeru, ko želimo primerjati dva atributa med seboj oziroma njuni vrednosti, saj morata biti tipa atributa v tem primeru enaka [4]. Atributi lahko imajo več vrednosti, kar pomeni, da v primeru nove vrednosti atributa, le-ta ohranja tudi svojo prejšnjo vrednost. Ob določitvi nove vrednosti atributu, se ta le doda v seznam že obstoječih vrednosti za ta atribut, vendar določeni atributi ne morejo vsebovati več vrednosti. Ali lahko določen atribut vsebuje več vrednosti ali ne, je določeno z njegovo definicijo v shemi LDAP. Spodnja slika 12 prikazuje primer predmeta z vsebovanimi atributi. Prav tako je s spodnje slike 12 razvidno, da ima atribut `telephoneNumber` dve vrednosti.

```
dn: ou=devices,dc=plainjoe,dc=org
objectclass: organizationalUnit
ou: devices
telephoneNumber: +1 256 555-5446
telephoneNumber: +1 256 555-5447
description: Container for all network enabled
 devices existing within the plainjoe.org domain
```

Slika 12: Primer predmeta z določenimi atributi v podatkovni bazi imeniške storitve LDAP [4].

Poleg rednih atributov (angl. *regular attributes*), ki so vsebovani v zgornjem primeru s slike 12, lahko posamezen predmet vsebuje posebne operativne atribute (angl. *operational attributes*), ki jih k predmetu doda imeniška storitev LDAP [2]. Te atribute uporablja le imeniška storitev LDAP za shranjevanje informacij o samih predmetih. Tako ti atributi niso namenjeni uporabi odjemalcev ter so za njih nevidni.

Kot je bilo omenjeno, shema LDAP vsebuje definicije atributov, ki med drugim vsebujejo informacijo, ali lahko določen atribut vsebuje le eno vrednost ali več. Slika 13 prikazuje definicijo atributa `telephoneNumber` z zgornjega primera slike 12.

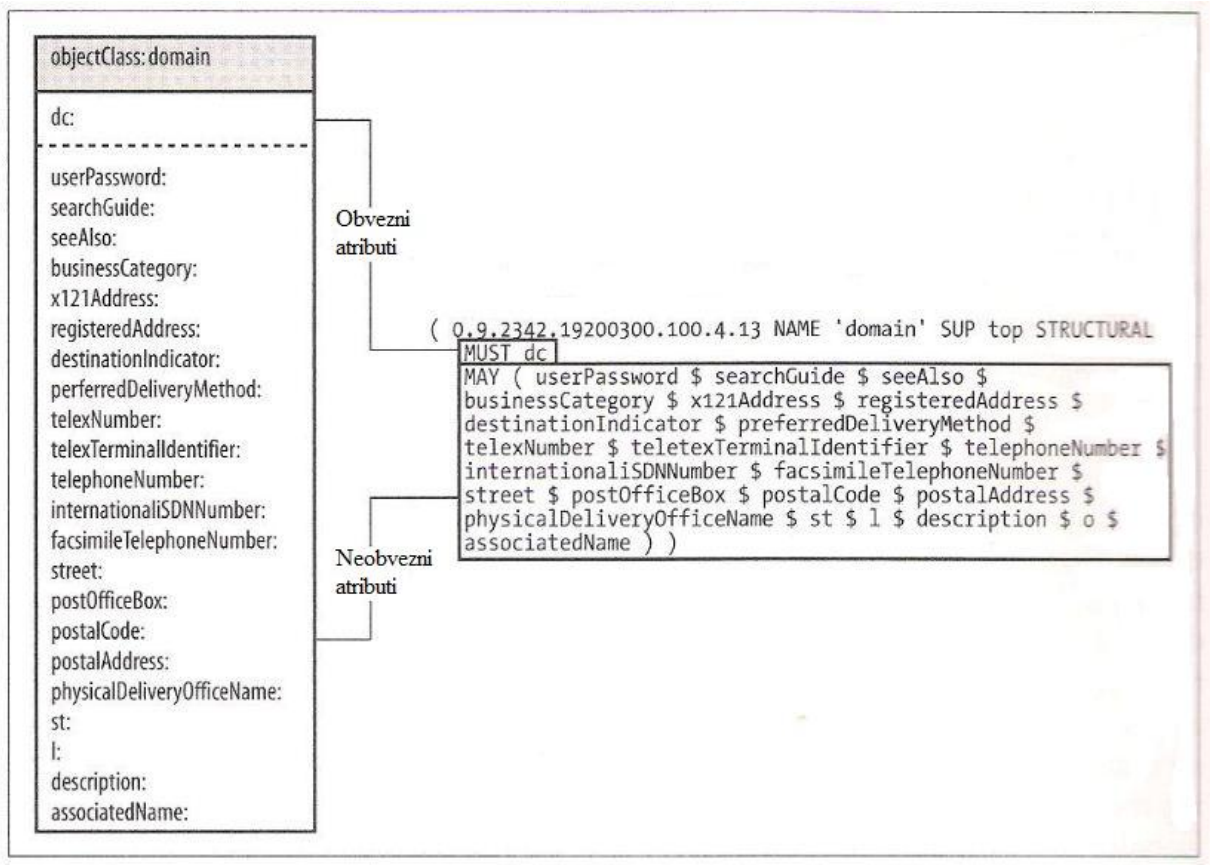


Slika 13: Primer definicije atributa telephoneNumber v shemi LDAP [4].

Kot vidimo v definiciji atributa `telephoneNumber` na zgornji slike 13, ni vidne informacije o tem, da atribut lahko vsebuje več vrednosti. Kadar atribut lahko vsebuje več vrednosti, ta informacija ni vsebovana, temveč je podana le takrat, ko atribut lahko vsebuje le eno vrednost. Definicija atributa `telephoneNumber` v shemi LDAP, kot je razvidno z zgornje slike 13, vsebuje tri pravila, kjer so pravila pod ključno besedo `EQUALITY` in `SUBSTR` med seboj povezana [4]. Pravilo `telephoneNumberMatch` se uporablja za primerjanje enakosti dveh atributov `telephoneNumber`, medtem ko se pravilo `telephoneNumberSubstringMatch` uporablja za primerjanje enakosti posameznih delov niza. Zadnje pravilo, ki se nahaja pod ključno besedo `SYNTAX` in je sestavljeno iz zaporedja števil in pik, predstavlja identifikator objekta (angl. *object identifier*), s katerim je določeno pravilo kodiranja, ki se uporablja za shranjevanje in prenašanje vrednosti atributa `telephoneNumber`. Število, ki je podano na koncu zaporedja števil in pik znotraj zavitih oklepajev, določa največjo dolžino vrednosti atributa `telephoneNumber`. Kot vidimo na zgornji sliki 13, atribut `telephoneNumber` vsebuje na začetku definicije podobno zaporedje števil in pik kot pravilo pod ključno besedo `SYNTAX`. V tem primeru gre prav tako za identifikator objekta, ki enolično določa atribut `telephoneNumber`. Nato sledi pod ključno besedo `NAME` le še ime atributa, ki pripada identifikatorju objekta.

Poleg običajnih atributov mora vsak predmet vsebovati poseben atribut, imenovan razred (`objectClass`), ki označuje tip predmeta (ta mora imeti vsaj eno določeno vrednost). Razred oziroma njegova vrednost v posameznem predmetu določa množico atributov, ki jih predmet lahko vsebuje oziroma ki jih mora vsebovati. V našem primeru zahteva na zgornji sliki 12, vrednost `organizationalUnit` razreda uporabo atributa `ou`, medtem ko so ostali prisotni atributi neobvezni, vendar znotraj množice dovoljenih [4]. Tako imeniška storitev LDAP s pomočjo vrednosti razreda določi, katere atribute mora vsebovati posamezni predmet, katere lahko vsebuje in katerih ne sme vsebovati. Posamezni predmet lahko ima več

razredov, vendar mora v tem primeru biti edini izmed teh tipa `STRUCTURAL`. Obstaja namreč več tipov razredov, ki jih bomo predstavili v nadaljevanju. Poleg tega lahko razred vsebuje več vrednosti, prav tako kot atributi, ter so prav tako definirani v shemi LDAP. Spodnja slika 14 prikazuje definicijo razreda z vrednostjo `organizationalUnit` v shemi LDAP, ki jo najdemo v našem primeru na zgornji sliki 12.



Slika 14: Primer definicije vrednosti razreda `organizationalUnit` v shemi LDAP [4].

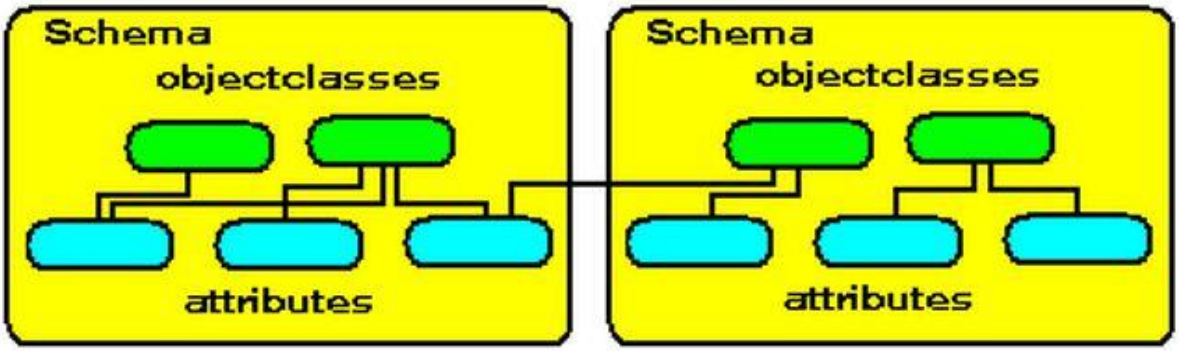
Kot je razvidno z zgornje slike 14, vsebuje definicija razreda tudi identikator objekta, kot je to pri definiciji atributov. Identifikator objekta ima prav tako isti pomen, kot ga ima pri definiciji atributov, saj ta enolično določa vrednost razreda, ki je v našem primeru `organizationalUnit`. Pod ključno besedo `MUST` so navedeni atributi, ki morajo biti prisotni v primeru vrednosti razreda `organizarionalUnit` v določenem predmetu, medtem ko so pod ključno besedo `MAY` navedeni atributi, ki niso obvezni, vendar so vseeno dovoljeni [4]. Zadnji dve ključni besedi, ki se pojavita v definiciji vrednost razreda `organizationalUnit`, sta `SUP` in `STRUCTURAL`. Ključna beseda `SUP` določa objekt, iz katerega izvira objekt `organizationalUnit` oziroma vrednost razreda. Tako objekt `organizationalUnit` podeduje vse zahteve atributov svojega starša, v našem primeru je to objekt `top`. Prav tako lahko določen atribut izvira iz drugega atributa in v tem primeru

podeduje vse njegove lastnosti oziroma zahteve. Tako objekti LDAP omogočajo enkratno dedovanje kot tudi objekti v programskem jeziku JAVA. Preostala ključna beseda `STRUCTURAL` pa predstavlja tip razreda. Na tem mestu je treba še omeniti, da lahko dve različni vrednosti razreda vsebujeta skupne atribute. Atribut `telephoneNumber` namreč pripada vrednosti razreda `organizationalUnit` ter prav tako vrednosti razreda `person`.

Poleg omenjenega tipa razreda `STRUCTURAL` lahko nastopi tudi tip razreda `AUXILARY` ali `ABSTRACT`. Lastnosti vseh tipov razreda so naslednje [4]:

- razred tipa `STRUCTURAL` predstavlja objekt oziroma vrednost razreda v realnem svetu. Tako v našem primeru z zgornje slike 14, vrednost razreda `organizationalUnit` predstavlja objekt v realnem svetu. Vsak predmet v podatkovni bazi imeniške storitve LDAP mora vsebovati natanko en razred tipa `STRUCTURAL`,
- razred tipa `AUXILARY` se lahko uporablja le v primeru, ko je razred tipa `STRUCTURAL` že vsebovan v samem predmetu. S pomočjo razreda tipa `AUXILARY` lahko predmetu dodamo določene neobvezne atribute, definirane v shemi LDAP,
- razred tipa `ABSTRACT` deluje enako kot pri objektno orientiranih programskih jezikih, saj se ta ne more uporabiti neposredno. Primer razreda tipa `ABSTRACT` je `top`, ki je bil uporabljen tudi v našem primeru z zgornje slike 14. Ta razred namreč predstavlja starša vseh razredov LDAP.

Shema LDAP, ki vsebuje definicije atributov in razredov, se nanaša na zbirko shem, ki so med seboj povezane. Tako so definicije razredov in atributov porazdeljene med več shemami [57]. Sheme oziroma njihov zapis je tipično vsebovan v tekstovnih datotekah ter shranjen v določenem imeniku. Kot so predmeti v podatkovni bazi imeniške storitve LDAP organizirani hierarhično, podobno so organizirane sheme, kot to prikazuje spodnja slika 15.

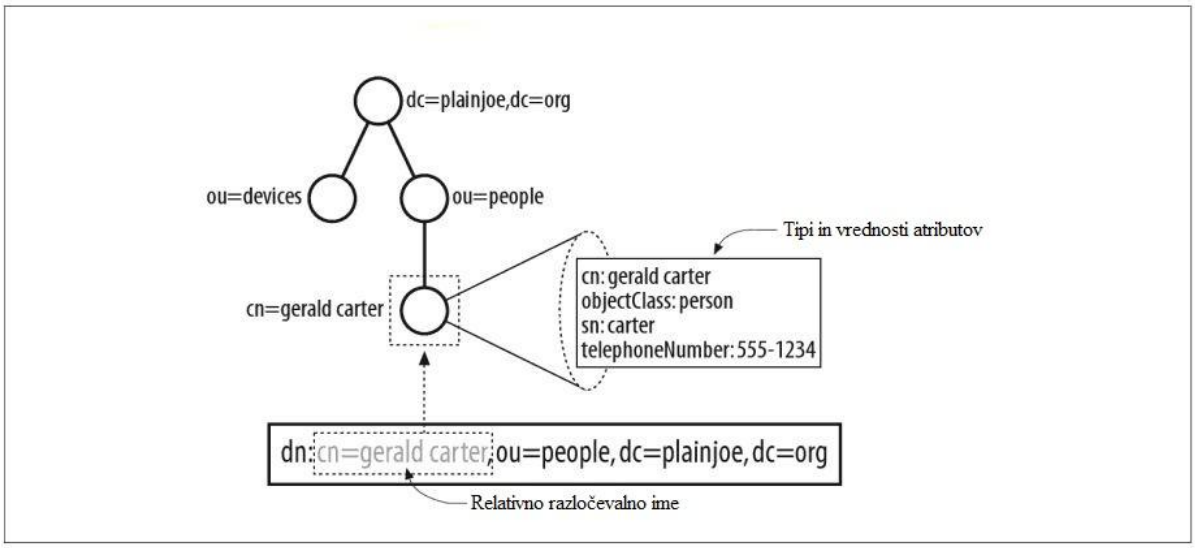


Slika 15: Prikaz vsebovanosti atributov in razredov v shemi LDAP [57].

Z zgornje slike 15 je razvidno, da sta leva in desna shema med seboj povezani. Atribut, ki je definiran v prvi shemi, lahko uporablja tudi razred, definiran v drugi shemi, kar naznanja povezave med shemami. V okviru sheme je treba omeniti, da mora biti vsak atribut oziroma razred uporabljen v posameznem predmetu, definiran v shemi LDAP. Poleg tega mora biti shema LDAP poznana strežniku LDAP, kar zagotovimo s pomočjo navedbe poti za določeno shemo. Kadar uporabljen atribut oziroma razred ni definiran v shemi LDAP oziroma ta ni poznana strežniku LDAP, se dodajanje predmeta v podatkovno bazo imeniške storitve LDAP zavrne [57].

2.4.3 Struktura imenika

Imeniška storitev LDAP vsebuje predmete, ki so urejeni hierarhično v obliki drevesne strukture (angl. *directory information tree*). Drevesna struktura predstavlja posebno obliko podatkovne baze, kjer se na vrhu hierarhije nahaja korenski predmet, iz katerega so nato izpeljani ostali predmeti. Tako vsako vozlišče v drevesni strukturi imeniške storitve LDAP predstavlja en predmet, ki vsebuje razločevalno ime (angl. *distinguished name*) in določene atribute. Primer hierarhične organizacije podatkov v obliki drevesne strukture prikazuje spodnja slika 16.



Slika 16: Organizacija predmetov v podatkovni bazi imeniške storitve LDAP [4].

V našem primeru se (z zgornje slike 16) drevesna struktura podatkov prične s korenskim predmetom `dc=plainjoe,dc=org`. S tem predmetom je imenovana podatkovna baza imeniške storitve LDAP ter se nanjo sklicuje odjemalec v primeru poslane zahteve na strežnik LDAP. Predmet `dc=plainjoe,dc=org` je zasnovan na podlagi imena domene, saj so imena domen enolično določena po celotnem medmrežju (angl. *internet*) [4]. S tem se namreč zagotovi globalna edinstvenost imeniške storitve LDAP. Nato sledita predmeta `ou=devices` in `ou=people`, ki predstavljata oddelek oziroma organizacijsko enoto. Pod oddelkom `ou=people` se nahaja predmet `cn=gerald carter`, ki definira osebo. Vsak predmet oziroma vozlišče v podatkovni bazi imeniške storitve LDAP je predstavljeno z razločevalnim imenom in določenimi atributi, kot je to pri predmetu `cn=gerald carter` (razvidno z zgornje slike 16), ki označuje tudi relativno razločevalno ime (angl. *relative distinguished name*). Treba je omeniti, da so posamezni predmeti v podatkovni bazi imeniške storitve LDAP shranjeni v hierarhičnem razmerju, kar tudi prikazuje zgornja slika 16. Oseba z relativnim razločevalnim imenom `cn=gerald carter` pripada oddelku `ou=people`, ki pripada podatkovni bazi `dc=plainjoe,dc=org` v imeniški storitvi LDAP.

Pomemben element v predmetu je, kot že omenjeno, razločevalno ime, ki ni občutljivo na velike oziroma majhne črke (angl. *case insensitive*). Razločevalno ime vsebuje vsak predmet v podatkovni bazi imeniške storitve LDAP, s katerim je enolično določen. Tako razlikovanje v primeru dveh podobnih predmetov ne predstavlja ovire [2, 4]. Razločevalno ime je sestavljeno iz določenih relativnih razločevalnih imen, ki označujejo ime posameznega vozlišča oziroma predmeta v podatkovni bazi imeniške storitve LDAP. V našem primeru je prikazano na zgornji sliki 16 razločevalno ime `cn=gerald`

carter,ou=people,dc=plainjoe,dc=org, ki enolično določa predmet z relativnim razločevalnim imenom cn=gerald carter. Ta predmet je, kot je razvidno iz razločevalnega imena predmeta, sestavljen z relativnim razločevalnim imenom predmeta ou=people in dc=plainjoe,dc=org. Razločevalna imena imajo določeno nedvoumno metodo, definirano v dokumentu RFC 2253, s pomočjo katere je določena sama oblika razločevalnega imena.

Sama drevesna struktura imeniške storitve LDAP je primerljiva z datotečnim sistemom, saj so predmeti v datotečnem sistemu prav tako predstavljeni hierarhično. Vsak imenik ima lahko več podimenikov, kjer ima vsak imenik svoj korenski imenik. Na ta način si lahko drevesno strukturo imeniške storitev LDAP lažje predstavljamo.

2.4.4 LDIF

LDIF predstavlja tekstovni format datoteke [4], ki ga uporablja imeniška storitev LDAP za shranjevanje svoje vsebine oziroma predmetov ter je definiran v dokumentu RFC 2849. Datoteke LDIF se zelo pogosto uporabljajo v primeru dodajanja novih predmetov oziroma spreminjanja obstoječih v podatkovni bazi imeniške storitve LDAP. Ob dodajanju predmeta oziroma spreminjanju obstoječih predmetov se preveri pravilnost zapisa v datoteki LDIF s pomočjo sheme LDAP na strežniku LDAP. Zapis mora namreč ustrezati pravilom v shemi LDAP. Datoteka LDIF lahko vsebuje več predmetov, v primeru dodajanja večjega števila predmetov hkrati, kjer so med seboj ločeni z uporabo prazne vrstice, kjer je vsak predmet predstavljen v več vrsticah s pomočjo atributov, ki ga opisujejo. Seveda je poleg atributov v datoteki LDIF podano tudi razločevalno ime (angl. *distinguished name*) za posamezen predmet.

Primer predmeta v datoteki LDIF prikazuje zgornja slika 12. Kot je razvidno s slike, je prvi atribut, ki se pojavi v datoteki LDIF, atribut dn ter kot vrednost vsebuje razločevalno ime. Ta atribut oziroma njegova vrednost se v vsakem primeru predmeta nahaja pred vsemi atributi ter s tem naznanja začetek novega predmeta in seveda enolično določa predmet. Razločevalnemu imenu sledijo atributi, kjer razred (objectClass) predstavlja pomembno vlogo, saj njegova vrednost določa obvezno prisotnost določenih atributov. Preostala dva atributa telephoneNumber in description pa sta neobvezna atributa, vendar dovoljena s strani vrednosti razreda organizationalUnit. Kot je razvidno iz imena atributa, atribut telephoneNumber vsebuje telefonsko številko, medtem ko atribut description kot vrednost vsebuje opis predmeta in je zgolj informativne narave. V tem primeru predmeta z zgornje slike 12 je razvidno, da se vrednost atributa description nahaja v dveh vrsticah,

saj se druga vrstica prične z določenim zamikom [4]. Nadaljevanje vrednosti atributa v novi vrstici je, v formatu LDIF, definirano z uporabo presledka pred samim začetkom naslednje vrstice. Tako ni treba dodajati posebnih znakov za prikaz nadaljevanja vrednosti določenega atributa.

Sam format LDIF definira določeno sintakso, ki jo je treba upoštevati. Sintaksa LDIF sestoji iz več pravil, med katerimi so najpomembnejši naslednji [4]:

- atribut `dn` enolično določa predmet, saj kot vrednost vsebuje razločevalno ime,
- za ločitev atributov in njihovih vrednosti je med njima vsebovano dvopičje. Tako se atribut nahaja na levi strani dvopičja, medtem ko se njegova vrednost nahaja na desni. Med dvopičjem in vrednostjo atributa je potreben presledek,
- komentar je predstavljen s pomočjo znaka šahovnice in opravlja svojo vlogo le v vrstici, ki se začne z znakom šahovnice.

2.4.5 Primer komunikacije

Komunikacija pri protokolu LDAP se prične z odjemalcem. Tako odjemalec s poslanim ukazom `bind` na strežnik LDAP izrazi željo po avtentikaciji ter posreduje uporabniško ime v obliki razločevalnega imena (angl. *distinguished name*) ter geslo v čistopisu (angl. *cleartext*). Strežnik LDAP nato preveri pravilnost pridobljenih podatkov odjemalca ter v primeru pravilnih podatkov odjemalcu dodeli dostop do imeniške storitve LDAP z določenimi pravicami oziroma omejitvami (avtorizacija). To metodo avtentikacije imenujemo preprosta avtentikacija (angl. *simple authentication*). Metoda ima veliko pomanjkljivost, saj se geslo od odjemalca do strežnika LDAP posreduje v čistopisu [4]. Poleg te metode avtentikacije sta na voljo tudi dve bolj varni, in sicer preprosta avtentikacija prek protokola SSL/TLS (angl. *simple authentication over SSL/TLS*) ter preprosta avtentikacija in varnostna plast (angl. *simple authentication and security layer*), ki sta prisotni v inačici protokola LDAPv3. V obeh metodah avtentikacije se namreč geslo ne prenaša v čistopisu ter sta posledično varnejši. Odjemalec se lahko avtenticira tudi z uporabo anonimne avtentikacije (angl. *anonymous authentication*), kjer mu ni treba posredovati razločevalnega imena in pripadajočega gesla. To metodo avtentikacije pa mora omogočati sam strežnik LDAP ter jo omogočiti le določenim aplikacijam. Ko je odjemalec avtenticiran oziroma mu strežnik LDAP odobri dostop, lahko odjemalec izvaja določene ukaze nad njegovo podatkovno bazo, kot so ukazi: `Add`, `Modify` in `Search`.

V primeru porazdeljene imeniške storitve LDAP [4], je lahko zahteva odjemalca posredovana na drugi strežnik LDAP, če glavni strežnik ne vsebuje potrebnih podatkov za odjemalca. Tako strežnik, ki pridobi zahtevo odjemalca, le posreduje zahtevo na strežnik, kjer so potrebni podatki.

Kot odjemalec imeniške storitve LDAP lahko deluje v primeru brezžičnega omrežja EDUROAM tudi strežnik RADIUS. Le-ta ima namreč gesla uporabnikov shranjena v imeniški storitvi LDAP ter mora dostopati do nje v okviru procesa avtentikacije.

2.5 Protokol KERBEROS

Protokol KERBEROS je primer avtentikacijskega protokola, ki deluje na osnovi tako imenovanih vstopnic (angl. *tickets*). Protokol je varen, saj nikoli ne prenaša gesel v čistopisu (angl. *cleartext*) po medmrežju (angl. *internet*) in tako zagotavlja varen proces avtentikacije. Vstopnice so kriptografska sporočila, ki izkažejo uporabniško identiteto strežniku brez pošiljanja uporabniškega gesla. Prav tako uporabniku ni treba hraniti gesla na svojem računalniku. Poleg omenjene varnosti zagotavlja protokol tudi naslednje lastnosti [8]:

- **enkratna prijava** (angl. *single-sign-on*) – uporabnik se mora predstaviti oziroma prijaviti v sistemu Kerberos le enkrat ter tako pridobi dostop do vseh storitev, ki podpirajo protokol KERBEROS,
- **zaupanja vredna tretja stran** (angl. *trusted third-party*) – Kerberos deluje s pomočjo avtentikacijskega strežnika, kateremu zaupajo vsi sistemi znotraj področja (angl. *realm*). Tako so vse avtentikacijske zahteve preusmerjene na omenjen avtentikacijski strežnik,
- **vzajemna avtentikacija** (angl. *mutual authentication*) – omogoča, da uporabnik preveri identiteto strežnika in hkrati tudi strežnik preveri identiteto uporabnika. Tako se strežnik in uporabnik prepričata, da komunicirata s pravim udeležencem.

Protokol KERBEROS onemogoča prisluškovanje, saj zagotavlja šifriranje poslanih sporočil. Šifriranje sporočil poleg preprečevanja prisluškovanja preprečuje tudi ponarejanje sporočil oziroma spreminjanje njegove vsebine (ohranja integriteto sporočila). Sam protokol temelji na simetrični kriptografiji in tako podpira številne simetrične algoritme. Široko podprt simetrični algoritem je algoritem DES oziroma 3DES, vendar omenjena algoritma nista povsem varna. Tako protokol omogoča tudi uporabo naprednega šifrirnega algoritma AES [8]. Kot je bilo omenjeno, šifriranje sporočil obenem ohranja tudi integriteto sporočila, saj je za šifriranje oziroma dešifriranje sporočil potreben ključ, ki je znan le dvema končnima točkama. Vendar

so prisotne tudi zgoščevalne funkcije za preverjanje integritete sporočila, s katerimi se lahko prepričamo, da sporočilo med prenosom res ni bilo spremenjeno. Tako protokol KERBEROS uporablja številne zgoščevalne funkcije v ta namen, med katerimi sta najbolj poznani MD5 in SHA1. Poleg tega protokol KERBEROS preprečuje napade s ponovitvijo (angl. *replay attacks*), kjer bi se lahko napadalec izdajal za določenega uporabnika.

Sam sistem Kerberos deluje na podlagi enega ali več centrov za distribucijo ključev (angl. *key distribution center*), ki so sestavni deli sistema Kerberos. Center za distribucijo ključev je sestavljen iz treh logičnih komponent: podatkovne baze, ki vsebuje vse uporabnike določenega področja skupaj z njihovimi gesli, avtentikacijski strežnik in strežnik za dodeljevanje vstopnic (angl. *ticket granting server*) [8]. Čeprav so vse tri logične komponente med seboj ločene oziroma ima vsaka svoj namen, te delujejo skupaj in znotraj enega programa oziroma procesa. Vsako področje Kerberos, kjer se nahajajo uporabniki in je ločeno od drugih področij Kerberos, mora vsebovati vsaj en center za distibucijo ključev. V primeru več centrov za distribucijo ključev znotraj določenega področja, so občutljivi podatki, kot so gesla uporabnikov na določenem področju Kerberos, shranjeni v vsakem centru za distribucijo ključev, kar ni povsem varno. Za uspešno avtentikacijo uporabnika za uporabo določene storitve Kerberos, je potrebno delovanje le enega centra za distribucijo ključev, s čimer se poveča varnost. V primeru več centrov za distribucijo ključev se moramo zavedati, da mora biti podatkovna baza na vsakem centru za distribucijo ključev sinhronizirana. Če namreč podatkovna baza ni sinhronizirana, lahko privede do tega, da strežnik ne bo mogel avtenticirati uporabnika, vendar sam protokol KERBEROS ne podpira metode za sinhronizacijo, tako je implementacija te metode prepuščena posameznikom (angl. *vendors*). Prednost centra za distribucijo ključev je seveda na strani administratorjev, saj olajša njihovo delo. Administratorji tako vzdržujejo le eno podatkovno bazo znotraj centra za distribucijo ključev.

Zadnje obstoječa inačica protokola KERBEROS je inačica 5, ki je definirana v dokumentu RFC 1510, vendar je dostopna tudi nova izdaja specifikacije protokola KERBEROS, in sicer, v dokumentu RFC 4120. Inačica 5 dodaja določene lastnosti in varnostne izboljšave, ki niso bile vključene v predhodni inačici 4. Inačica 4 je prva inačica protokola KERBEROS, ki je bila izdana uporabnikom, vendar danes ni več veliko implementacij te inačice. Začetne inačice 1, 2 in 3 so bile uporabljene le za testne namene znotraj inštituta MIT, ki je protokol KERBEROS razvil [8]. Čeprav je bil protokol KERBEROS razvit na inštitutu MIT, so poleg njihove implementacije dostopne tudi nekatere druge, med katerimi sta Heimdal in Microsoft. Uporaba protokola KERBEROS je zelo priljubljena, kar nakazuje njegova prisotnost v operacijskih sistemih podjetja Microsoft; priljubljen je v manjših in srednje velikih omrežjih.

2.5.1 Vstopnice

Kot je bilo omenjeno, zagotavlja protokol KERBEROS varen proces avtentikacije s pomočjo vstopnic (angl. *tickets*), ki izkažejo identiteto uporabnika. Te vstopnice dodeli oziroma generira center za distribucijo ključev (angl. *key distribution center*) na zahtevo uporabnika, ki izrazi željo po avtentikaciji. Vstopnice predstavljajo ključni koncept v samem protokolu KERBEROS ter so šifrirane, da bi preprečili napadalcu prevzem veljavne vstopnice. Pri vstopnicah gre za šifrirano podatkovno strukturo, ki vključuje enolični sejni ključ (angl. *session key*) za vsako sejo ter zastavice (angl. *ticket flags*) [8]. Ta sejni ključ uporabita odjemalec kot tudi strežnik zaradi potrebe po varni komunikaciji. Zaradi vstopnic uporabniku ni treba pošiljati uporabniškega imena in gesla storitvi Kerberos v okviru procesa avtentikacije, saj vstopnica identificira uporabnika. Tako uporabniško ime in geslo nikoli ni poslano po medmrežju (angl. *internet*). Vstopnice lahko opredelimo tudi kot dovoljenja za uporabo določene storitve Kerberos.

Vsaka vstopnica vsebuje številne podatke in je običajno shranjena v datoteki poleg vseh ostalih vstopnic, ki jih pridobi določen uporabnik. Datoteka se namreč nanaša le na enega uporabnika ter prav tako pripada le enemu uporabniku v določenem času. Glavni podatki, ki so prisotni v vsaki vstopnici, so [8]: uporabniško ime (angl. *the user's principal name*), ime storitve (angl. *the service's principal name*), veljavnost vstopnice in že omenjeni sejni ključ. Podatka, kot sta veljavnost vstopnice in sejni ključ, doda center za distribucijo ključev, medtem ko ostale podatke doda sam odjemalec oziroma uporabnik.

Vsak uporabnik, ki želi dostopati do določene storitve KERBEROS, mora prejeti dve vstopnici različnega tipa. Poznamo dva tipa vstopnic [8, 31]:

- **začetna vstopnica** (angl. *ticket granting ticket*) – vstopnica deluje kot glavni identifikator (angl. *identifier*), ki identificira uporabnika pri uporabi katerekoli storitve KERBEROS. Vstopnico pridobi uporabnik ob prijavi v sistem KERBEROS, in sicer od avtentikacijskega strežnika znotraj centra za distribucijo ključev,
- **vstopnica za določeno storitev** (angl. *service ticket*) – vstopnica je potrebna za dostop do določene storitve Kerberos in je uporabljena namesto posredovanja uporabniškega imena in gesla določenega uporabnika. To vstopnico pridobi odjemalec s poslano zahtevo na strežnik za dodeljevanje vstopnic (angl. *ticket granting server*) znotraj centra za distribucijo ključev. Sama zahteva mora vsebovati ime storitve, do katere želimo dostopati in omenjeno začetno vstopnico.

Dobra lastnost vstopnic je seveda njihova določena veljavnost, saj ob morebitnem prevzemu vstopnice s strani napadalca, ta ne more uporabljati določene storitve neskončno dolgo. Prav tako je z omejeno veljavnostjo vstopnice zmanjšana možnost škode, ki jo lahko povzroči napadalec v primeru kraje vstopnice. Veljavnost začetne vstopnice in vstopnice za določeno storitev je navadno različna, vendar se giblje od 10 do 24 ur [8]. V primeru prenehanja uporabe določene storitve Kerberos pa je možno omenjeni vstopnici tudi uničiti ter tako zagotoviti večjo varnost.

2.5.2 Primer komunikacije

Tipičen primer komunikacije pri protokolu KERBEROS seveda nastopa med odjemalcem in centrom za distribucijo ključev (angl. *key distribution center*) ter je tako zasnovan na podlagi modela odjemalec-strežnik (angl. *client-server model*). Komunikacija se prične z odjemalcem, ki se želi prijaviti v določen sistem oziroma področje Kerberos (angl. *Kerberos realm*) [8]. V ta namen mu avtentikacijski strežnik znotraj centra za distribucijo ključev izda začetno vstopnico (angl. *ticket granting ticket*), ki je šifrirana s pomočjo odjemalčevega gesla. Center za distribucijo ključev vsebuje gesla odjemalcev v svoji podatkovni bazi in je tako geslo znano le odjemalcu in centru za distribucijo ključev. Tako lahko ob prijavi odjemalca v sistem Kerberos avtentikacijski strežnik ugotovi, ali gre za pravega uporabnika na drugi strani oziroma ga avtenticira, saj bo le pravo geslo uporabnika dešifriralo začetno vstopnico. Kadar odjemalec ne uporabi pravega gesla, začetna vstopnica ni uporabna in odjemalec mora poskusiti ponovno. Ob uporabi pravega gesla nad začetno vstopnico, se ta dešifrira in je tako uporabna za pridobitev naslednje vstopnice. Odjemalec je na tem mestu pridružen v določeno področje Kerberos, vendar ta potrebuje še vstopnico za določeno storitev (angl. *service ticket*), ki mu bo omogočala dostop do določene storitve Kerberos. Odjemalec tako pošlje zahtevo za vstopnico za določeno storitev na strežnik za dodeljevanje vstopnic (angl. *ticket granting server*) znotraj centra za distribucijo ključev. Strežnik od odjemalca zahteva ime storitve, do katere želi odjemalec dostopati, ter začetno vstopnico, ki mu jo je izdal avtentikacijski strežnik. Strežnik za dodeljevanje vstopnic nato preveri veljavnost začetne vstopnice ter nato izda vstopnico za določeno storitev. Odjemalec lahko nato z uporabo vstopnice za določeno storitev dostopa do določene storitve Kerberos (dostopa lahko do storitve, za katero je zahteval vstopnico.

2.6 Šifriranje in integriteta sporočil

Ko govorimo o varnosti v računalniških omrežjih, je poleg samega procesa avtentikacije pomembno tudi šifriranje in ohranjanje integritete sporočil, kar nam skupaj zagotavlja varno komunikacijo. Poleg tega varna komunikacija vključuje tudi naslednje:

- **preprečevanje zanikanja** (angl. *nonrepudiation*) – sistem ima podpis, da je uporabnik nekaj prejel oziroma poslal. Tako ta ne more zanikati, da tega ni storil,
- **razpoložljivost in nadzor dostopa** – gre za proces avtorizacije, saj se ugotavlja, kaj določen uporabnik sme početi oziroma kakšne so njegove pravice,
- **beleženje dogodkov oziroma dostopov** – tu nastopi proces beleženja za določenega uporabnika.

Šifriranje je proces, ki omogoča pretvorbo sporočil v obliko, ki ni berljiva in s tem onemogoča prisluškovanje s strani napadalcev. Proces šifriranja omogočajo številne kriptografske metode oziroma šifrirni algoritmi, med katerimi so določeni bolj enostavni in drugi kompleksnejši. Pretvorbo šifrirnega sporočila nazaj v prvotno obliko pa omogoča proces dešifriranja [35]. Za proces dešifriranja je potreben pravi ključ, saj le s pomočjo tega lahko na enostaven način pridemo do prvotnega sporočila. Sporočilo je šifrirano z ustreznim ključem ter tako onemogoča napadalcem dešifrirati sporočilo, saj mora ta poznati ustrezen ključ. Uporaba enostavnih šifrirnih algoritmov ni povsem varna, saj jih je možno dokaj hitro razbiti oziroma priti do berljive vsebine brez poznanega ključa. Tako je z uporabo kompleksnejšega šifrirnega algoritma bistveno težje razbiti algoritem oziroma prisluškovati komunikaciji. Za današnje kompleksnejše šifrirne algoritme lahko rečemo, da jih je praktično nemogoče razbiti brez ustreznega ključa.

Proces šifriranja je zelo pomemben v primeru brezžičnega omrežja, saj je to omrežje lažje zlorabiti kot žično [35], vendar je uporaba šifriranja ustrezna tudi v žičnih omrežjih, še posebej v primeru izvajanja vseh vrst občutljivih transakcij, kot je nakupovanje na spletu s pomočjo kreditnih kartic. Prav tako se šifriranje lahko uporablja za varovanje podatkov na računalnikih in različnih napravah za shranjevanje podatkov. Tako so podatki v primeru kraje ali izgube varnejši.

Kriptografske metode oziroma šifrirne algoritme delimo po načinu in lastnostih ključa. Po načinu se ti delijo na substitucijske, kjer nadomestimo posamezne črke ali dele besedila z drugimi, in transpozicijske, kjer spreminjamo vrstni red znakov oziroma besed, po lastnostih ključa pa se te delijo na simetrične in asimetrične kriptografske metode.

Poleg preprečevanje prisluškovanja s pomočjo šifriranja, se je treba prepričati, ali je sporočilo veljavno oziroma ni bilo spremenjeno med pošiljanjem s strani morebitnega napadalca. Kadar sporočilo ni bilo spremenjeno, pravimo, da je ohranilo celovitost oziroma integriteto. O integriteti sporočila se lahko prepričamo s pomočjo zgoščevalnih funkcij, ki so izvedene na podlagi poslanega oziroma prejetega sporočila. Zgoščevalne funkcije smo podrobneje predstavili v podpoglavju 2.7.

2.6.1 Klasične kriptografske metode

Klasične kriptografske metode so preproste metode šifriranja, ki vključujejo zamenjavo znaka oziroma besed z drugimi ter so tako znakovne usmerjene. Te metode se danes ne uporabljajo več, saj jih je možno z računalniki enostavno razbiti in tako priti do skrite vsebine. Ker uporabljajo en sam ključ za šifriranje in dešifriranje, spadajo med simetrične kriptografske metode. Nekatere izmed klasičnih kriptografskih metod so naslednje:

- **Cezarjev kriptogram** – primer substitucijske kriptografske metode, kjer se vsak znak besedila, ki ga želimo šifrirati, nadomesti z drugim. Vsak znak se zamenjava z znakom, ki je za določeno število mest oddaljen od trenutnega znaka po abecedi. Ključ je tako predstavljen z abecedo, kjer je vsak znak abecede pomaknjen za določeno število mest naprej v primerjavi z abecedo [38]. Tako v primeru ključa, kjer je vsak znak pomaknjen za štiri mesta naprej, se znak A preslika v znak U in znak B v znak V. Vsako sporočilo lahko šifriramo s pomočjo 25 različnih ključev, kolikor je tudi znakov v slovenski abecedi. Prav tako je Cezarjev kriptogram možno razbiti v največ 25 poskusih,
- **Vigenerjev kriptogram** – kriptogram temelji na Vigenerjevi matriki, ki vsebuje vse Cezarjeve abecede. Tako je matrika sestavljena iz 25 vrstic in 25 stolpcev, kjer se prva vrstica prične z znakom A, nato pa si sledijo s pričetkom naslednjega zaporednega znaka abecede [39]. Pričetek vsake vrstice ali stolpca določa ustrezen znak, ki sledi abecedi. Dolžina sporočila, ki ga želimo šifrirati, mora biti enaka dolžini izbranega ključa. Določen znak sporočila šifriramo tako, da ga nadomestimo z znakom, ki se nahaja v določeni vrstici oziroma stolpcu Vigenerjeve matrike. Tako je s trenutnim znakom ključa in trenutnim znakom sporočila določena vrstica oziroma stolpec, Vigenerjeve matrike. Vrstico oziroma stolpec izberemo s pomočjo začetnih znakov vsake vrstice oziroma stolpca, ki sledijo abecedi, saj morata biti trenutni znak ključa in sporočila enaka začetnim znakom vrstice oziroma stolpca,

- **Porterjev kriptogram** – temelji na matriki simbolov, kjer pričetek vsake vrstice oziroma stolpca naznanja znak abecede. Kriptogram šifrira po dva znaka hkrati, kar pomeni, da en simbol iz matrike simbolov nadomesti dva znaka sporočila, ki ga želimo šifrirati. Znaka sporočila, ki ju želimo šifrirati, določata vrstico oziroma stolpec simbola, s katerim bosta šifrirana,
- **transpozicijski kriptogram** – kriptogram znake ali dela besedila premesti na podlagi tabele z vsebovanim ključem in sporočilom, ki ga šifriramo. Ključ mora biti sestavljen iz različnih znakov, saj ključ oštevilčimo po abecedi, kjer je vsak stolpec znakov predstavljen z določeno številko. Ključ vsebuje prva vrstica tabele, medtem ko ostale vsebujejo sporočilo, ki ga želimo šifrirati, vendar se ta razteza le do dolžine ključa v posamezni vrstici. Šifriranje izvedemo tako, da sledimo zaporednim številkam stolpcev in prepisujemo vsebino trenutnega stolpca.

V nadaljevanju smo prikazali kriptografske metode, ki se uporabljajo danes. Sem sodijo seveda simetrične kakor tudi asimetrične kriptografske metode, kjer imata obe vrsti določene prednosti oziroma slabosti. Sodobne kriptografske metode so, za razliko od klasičnih, bitno usmerjene in delujejo s pomočjo računalniških algoritmov.

2.6.2 Simetrične kriptografske metode

Glavna značilnost simetričnih kriptografskih metod oziroma algoritmov je uporaba enakega ključa pri procesu šifriranja kakor tudi pri procesu dešifriranja določenega sporočila. Torej morajo udeleženci, ki želijo med seboj komunicirati zaupno oziroma šifrirati sporočila, poznati ta ključ. Tu nastopi značilna slabost simetričnih kriptografskih metod v primerjavi z asimetričnimi, in sicer težava z distribucijo ključev, saj morajo udeleženci pred pričetkom seje ta ključ pridobiti na varen način [40]. Vendar pa imajo simetrične kriptografske metode tudi prednost v primerjavi z asimetričnimi, saj je sam proces šifriranja oziroma dešifriranja bistveno hitrejši zaradi strojne implementacije. Simetrične kriptografske metode so blokovno orientirane, kjer posamezen blok vsebuje več bitov, te bite pa šifriramo kot celoto. Osnovni operaciji simetričnih kriptografskih metod sta:

- **P−škatla** (transpozicija) – škatla vsebuje določeno število vhodov in tudi izhodov. V primeru permutacije je število vhodov in izhodov enako, medtem ko je v primeru redukcije ali ekspanzije izhodov manj ali več in se tako biti izgubljajo oziroma podvajajo. Pri tej škatli je vsak vhod predstavljen z indeksom, medtem ko je vsak izhod predstavljen s številom, ki tvori ključ. Tako tu spreminjamo zaporedje indeksov vhodov glede na izhod s pomočjo ključa,

- **S−škatla** (substitucija) – je predstavljena s pomočjo dekoderja, P−škatle in koderja, kjer dekoder predstavlja vhod in koder izhod škatle, medtem ko se P–škatla nahaja med njima. Dekoder in koder delujeta na podlagi njune tabele, saj določen vhod pretvorita v določen izhod. Med njima nastopi še vloga omenjene P–škatle. Tako v primeru dekoderja 3/8 ta pretvori 3-bitni vhod v 8-bitni izhod, kar nato pride kot vhod v P–škatlo. Izhod iz P–škatle je nato posredovan kot vhod v koder 8/3, ki nato pretvori 8-bitni vhod nazaj v 3-bitnega.

Danes se simetrične kriptografske metode pogosto uporabljajo v kombinaciji z asimetričnimi, saj se na tak način izognemo težavam z distribucijo ključa, ki jih prinašajo simetrične kriptografske metode. Prav tako lahko s tem odpravimo slabost asimetričnih kriptografskih metod, saj se za samo šifriranje komunikacije uporabijo simetrične kriptografske metode, kar privede do hitrejšega procesa šifriranja. Najbolje poznana oziroma uporabljena simetrična algoritma sta naslednja:

- **DES** – algoritem DES je bil včasih najpogosteje uporabljen simetrični algoritem, vendar danes ni več varen za uporabo. Nad algoritmom DES je bil pred leti izveden uspešen napad z grobo silo (angl. *brute force attack*), zato danes ne spada med varne simetrične algoritme. Prav tako je dolžina ključa 56-bitov majhna, kar potrjuje slabšo varnost [41]. Algoritem DES temelji na tako imenovanem procesu `Feistel scheme`, ki je sestavljen iz določenih stopenj, med katerimi sta tudi substitucija in permutacija. Proces deluje na blokih dolžine 32-bitov, kar je polovica dolžine celotnega bloka, ter se ponovi 16-krat za določen 64-bitni blok. Zaradi slabše varnosti algoritma DES je bil razvit njegov naslednik 3DES, ki velja za varnejši algoritem, saj je bolj odporen na napad z grobo silo, a je algoritem 3DES 3-krat počasnejši v primerjavi z algoritmom DES zaradi trojnega šifriranja. Algoritem 3DES je sedaj nadomestil algoritem AES, ki odpravlja slabosti,
- **AES** – danes najbolj uporabljen simetrični algoritem, ker je hitrejši in varnejši v primerjavi z algoritmom DES. Dolžina ključa znaša različno, in sicer 128, 192 ali 256 bitov, kar je precej več kot pri algoritmu DES, kjer znaša dolžina ključa 56 bitov in se ne spreminja [42]. Tako je tudi na podlagi dolžine ključa razvidno, da gre za varnejši algoritem. Prav tako algoritem deluje nad bloki dolžine 128 bitov, za razliko od algoritma DES, kjer algoritem deluje nad bloki dolžine 64 bitov. Bloki so pri algoritmu AES razporejeni v matrike, pogosto dimenzije 4x4, nad katerimi se izvajajo različne operacije, ki se večkrat ponovijo v posameznih iteracijah. Med osnovne operacije algoritma sodijo, poleg substitucije in mešanja vrstic (angl. *shift row*), tudi mešanje stolpcev (angl. *mix column*) in drugačen primer substitucije (angl. *add round key*).

2.6.3 Asimetrične kriptografske metode

Za razliko od simetričnih kriptografskih metod zahtevajo asimetrične kriptografske metode uporabo par različnih ključev, kjer je eden javni ter tako dostopen vsem ter drugi zasebni, ki je znan le lastniku. Vsak udeleženec v komunikaciji mora imeti par zasebnega in javnega ključa, s katerima lahko šifrira oziroma dešifrira sporočila [36, 45]. Ko pošiljatelj pošlje sporočilo prejemniku, ga ta šifrira z javnim ključem prejemnika in tako zagotovi sporočilo zaupno, saj lahko le prejemnik dešifrira sporočilo z uporabo svojega zasebnega ključa. Tako mora vsak pošiljatelj, ki želi ohraniti sporočilo zaupno, šifrirati sporočilo z javnim ključem prejemnika. Čeprav sta ključa med seboj različna, velja med njima določena povezava, saj lahko le zasebni ključ, ki pripada določenemu javnemu ključu, dešifrira sporočilo. Kljub znanemu javnemu ključu je nemogoče ugotoviti pravi par zasebnega ključa. Prav tako je na podlagi šifriranega sporočila z javnim ključem prejemnika in sporočilom v čistopisu (angl. *cleartext*) nemogoče ugotoviti pravi par zasebnega ključa.

Pri uporabi dveh ključev ni težav z distribucijo, kot je to pri simetričnih kriptografskih metodah, saj ni potrebe po izmenjavi ključev in so tako asimetrične kriptografske metode varnejše. Slaba stran asimetričnih kriptografskih metod nastopi pri hitrosti šifriranja oziroma dešifriranja, saj so počasnejše.

Asimetrični algoritmi, kot je RSA, poleg zaupnosti omogočajo pošiljatelju podpis njegovega sporočila. Tu gre za digitalni podpis, ki dokazuje avtentičnost sporočila. Digitalni podpis mora omogočati naslednje:

- preverjanje veljavnosti s strani prejemnika,
- onemogočanje ponarejanja,
- onemogočanje zanikanja podpisa s strani pošiljatelja.

V primeru algoritma RSA pošiljatelj najprej podpiše sporočilo s svojim zasebnim ključem ter ga nato šifrira še z javnim ključem prejemnika in pošlje prejemniku. Prejemnik nato dešifrira prejeto sporočilo s svojim zasebnim ključem in nato rezultat ponovno dešifrira z uporabo javnega ključa pošiljatelja ter na ta način preveri podpis.

V praksi poteka celoten postopek komunikacije med pošiljateljem in prejemnikom z uporabo simetričnega algoritma RSA na naslednji način: pošiljatelj naprej izračuna izvleček sporočila (angl. *hash value*) s pomočjo zgoščevalne funkcije in šifrira sporočilo z javnim ključem prejemnika. Nato pošiljatelj podpiše oziroma šifrira izvleček s svojim zasebnim ključem (pridobi digitalni podpis sporočila) ter ga nato še šifrira z javnim ključem prejemnika. Celotno

sporočilo skupaj s podpisanim in šifriranim izvlečkom pošlje pošiljatelj prejemniku, ki dešifrira sporočilo s svojim zasebnim ključem in prav tako dešifrira izvleček s svojim zasebnim ključem ter tako pridobi digitalni podpis sporočila. Nato prejemnik preveri veljavnost podpisa z dešifriranjem podpisa, z javnim ključem pošiljatelja ter v primeru veljavnega podpisa pridobi izvleček in tako preveri avtentičnost sporočila (prepriča se o pravem pošiljatelju). Nato prejemnik izračuna izvleček nad pridobljenim sporočilom z uporabo zgoščevalne funkcije ter primerja rezultat z izvlečkom, ki ga je pridobil od pošiljatelja. Prejemnik se prepriča, da sporočilo med prenosom ni bilo spremenjeno, če gre za enake vrednosti obeh izvlečkov.

2.7 Zgoščevalne funkcije

Zgoščevalne funkcije so funkcije oblike `f=h(m)`, ki prejmejo kot argument poljubno dolgo sporočilo (m) in kot rezultat vrnejo izvleček (angl. *hash value*) omejene dolžine (f). Rezultat zgoščevalne funkcije oziroma izvleček je tako krajši v primerjavi z originalnim sporočilom, ki ga funkcija prejme in je fiksne dolžine. Sam algoritem zgoščevalnih funkcij temelji na preprostih bitnih operacijah brez ključa, kjer se vhodno sporočilo deli v bloke, ki se procesirajo v več ciklih (sam postopek je hitrejši v primerjavi z šifrirani algoritmi). Zgoščevalne funkcije, ki so dobre oziroma varne, vsebujejo naslednje lastnosti [46]:

- izvleček oziroma rezultat zgoščevalne funkcije, ki ga ni težko pridobiti iz poljubnega vhodnega sporočila,
- so enosmerne (angl. *one-way*), kar onemogoča na podlagi določenega izvlečka pridobiti njegovo vhodno sporočilo, iz katerega je izvleček nastal,
- majhna sprememba sporočila privede do povsem drugega izvlečka, oziroma rezultata zgoščevalne funkcije,
- preprečujejo nastanek kolizije. Kolizija nastane, kadar imata dve sporočili ali več različnih enak izvleček oziroma rezultat zgoščevalne funkcije in predstavlja enega izmed glavnih težav, ki se lahko pojavijo (do nastanka kolizije pride v redkih primerih, ko gre za dobre zgoščevalne funkcije).

Zgoščevalne funkcije se pogosto uporabljajo pri sporočilih, da bi se preverila integriteta sporočila. Tako pošiljatelj skupaj s sporočilom pošlje tudi izvleček sporočila, ki ga nato uporabi prejemnik za preverjanje integritete sporočila. Ta namreč izračuna svojo vrednost izvlečka z uporabo zgoščevalne funkcije pri prejetem sporočilu in preveri enakost obeh izvlečkov. V primeru enakosti je sporočilo ohranilo celovitost oziroma integriteto in tako med prenosom ni bilo spremenjeno s strani morebitnega napadalca. Prav tako se zgoščevalne

funkcije uporabijo v primeru preverjanja avtentičnosti sporočila, saj se rezultat zgoščevalne funkcije podpiše pred samim pošiljanjem z zasebnim ključem pošiljatelja. Podpis se tako izvede le nad izvlečkom, saj je podpisovanje celotnega sporočila zamudno. Tako v tem primeru prejemnik preveri veljavnost podpisa in tudi integriteto sporočila (celoten potek komunikacije smo opisali v prejšnjem podpoglavju pri primeru asimetričnega algoritma RSA).

Uporaba zgoščevalnih funkcij je možna tudi pri uporabniških geslih, shranjenih v podatkovnih bazah, imenikih ali tekstovnih datotekah. Tako z uporabo zgoščevalne funkcije onemogočimo vidnost gesla v čistopisu (angl. *cleartext*), saj je geslo shranjeno v obliki izvlečka oziroma rezultata zgoščevalne funkcije ter je posledično varnejše. Izvleček enolično določa uporabniško geslo.

Poleg običajnih zgoščevalnih funkcij brez ključa obstajajo tudi posebne zgoščevalne funkcije s ključem, imenovane HMAC. Izraz HMAC izhaja iz algoritma MAC, ki temelji na zgoščevalni funkciji ter tako nastopi skupaj z uporabo zgoščevalne funkcije. Algoritem MAC prejme poleg poljubno dolgega sporočila tudi tajni ključ in vrne vrednost MAC [47, 48]. Rezultat zgoščevalne funkcije HMAC nam omogoča preveriti integriteto sporočila in tudi avtentičnost. Tajni ključ, ki ga algoritem MAC prejme kot parameter, si morata pošiljatelj in prejemnik izmenjati pred pričetkom komunikacije, saj lahko le tako izračunata vrednost algoritma MAC. Pred pričetkom pošiljanja sporočila pošiljatelj izvede algoritem MAC nad sporočilom s pomočjo tajnega ključa in pridobi vrednost MAC, ki jo nato pošlje skupaj s sporočilom prejemniku. Prejemnik nato ponovno izvede algoritem MAC nad prejetim sporočilom s pomočjo tajnega ključa ter preveri enakost svojega rezultata MAC s tistim, ki ga je prejel od pošiljatelja. V primeru enakosti obeh vrednosti MAC je sporočilo ohranilo celovitost oziroma integriteto in je avtentično oziroma izvira od pravega pošiljatelja.

Algoritem MAC se lahko uporabi s številnimi zgoščevalnimi funkcijami, vendar je v tem primeru še vedno pomembna uporaba dobre zgoščevalne funkcije [48], saj je na podlagi dolžine izvlečka in ključa določena moč algoritma MAC. Algoritem se velikokrat uporabi z zgoščevalno funkcijo MD5 in SHA-1, ki ju bomo predstavili v nadaljevanju.

V primeru uporabe zgoščevalne funkcije HMAC ni poskrbljeno za preprečevanje zanikanja (angl. *non-repudiation*), kot je to pri digitalnih podpisih [47]. Pri zgoščevalni funkciji HMAC namreč uporabljata pošiljatelj in prejemnik isti tajni ključ za izračun vrednosti MAC, kar spominja na simetrično kriptografijo. S tem tajni ključ ne pripada točno določenemu uporabniku, kot je to pri asimetrični kriptografiji.

2.7.1 MD5

Zgoščevalna funkcija MD5 prejme kot vhod poljubno dolgo sporočilo ter vrne oziroma izračuna izvleček (angl. *hash value*) dolžine 128 bitov. Algoritem MD5 je od samega začetka določen v dokumentu RFC 1321, vendar je njegova dokumentacija posodobljena v dokumentu RFC 6151 z varnostnega stališča. Uporaba zgoščevalne funkcije MD5 danes ni več primerna v primerih, kjer je odpornost na kolizije nujno potrebna, kot je to pri digitalnih podpisih. Zgoščevalna funkcija MD5 ne preprečuje nastanka kolizij, saj so bili nad njo izvedeni številni uspešni napadi, ki so pokazali ranljivost zgoščevalne funkcije [49].

Zgoščevalna funkcija MD5 s ključem, v tem primeru HMAC-MD5, je varnejša. Algoritem MAC je bistveno odpornejši na kolizije kakor sama zgoščevalna funkcija MD5. Tako je uporaba zgoščevalne funkcije HMAC-MD5 primernejša in se njena uporaba ne izključuje v obstoječih protokolih, vendar novejše implementacije protokolov ne vključujejo uporabe zgoščevalne funkcije HMAC-MD5, odkar uporaba zgoščevalne funkcije MD5 ni primerna, ko gre za digitalne podpise (primer izvlečka zgoščevalne funkcije MD5 in HMAC-MD5, prikazuje spodnja slika 17).

```
MD5 ("Testni niz") = 5b6c30e1ecde0a206664b446282b94cc
HMAC-MD5 ("kljuc", "Testni niz") = 3c229ca53ecec9c63326a090e9320557
```

Slika 17: Primer izvlečka zgoščevalne funkcije MD5 in HMAC-MD5, z istim vhodnim nizom.

Danes je priporočljiva uporaba zgoščevalnih funkcij iz družine SHA-2, kot je HMAC-SHA256 [14]. Družino zgoščevalnih funkcij SHA-2 smo predstavili v naslednjem podpoglavju.

Kadar želimo z uporabo zgoščevalne funkcije skriti uporabniško gesla v čistopisu (angl. *cleartext*), lahko poleg MD5 uporabimo zgoščevalno funkcijo SMD5. Ta namreč prejme tako imenovano sol (angl. *salt*), ki je predstavljena z naključno generiranimi biti [15]. Sol se uporabi v kombinaciji z uporabniškim geslom kot vhod v zgoščevalno funkcijo, ter se tako poda kot atribut zgoščevalni funkciji. Izvleček oziroma uporabniško geslo je tako varnejše in bolj odporno na napad s slovarjem (angl. *dictionary attack*). Ker je dodana sol lahko poljubna, ne moremo vedeti kakšen izvleček bo generirala zgoščevalna funkcija, saj sol vpliva na rezultat zgoščevalne funkcije. Tako je v tem primeru onemogočen tudi napad z mavričnimi tabelami (angl. *rainbow attack*).

2.7.2 SHA

Izraz SHA zajema tri primere zgoščevalnih funkcij, in sicer SHA-0, SHA-1 in SHA-2, medtem ko je zgoščevalna funkcija SHA-3 še v razvoju. Zgoščevalna funkcija SHA-0 je bila umaknjena iz uporabe zaradi določenih napak v samem algoritmu, ki so privedle do slabše varnosti zgoščevalne funkcije ter do nastanka kolizij. Tako je bila zgoščevalna funkcija SHA-0 nadomeščena z zgoščevalno funkcijo SHA-1, ki je podobna zgoščevalni funkciji SHA-0, vendar odpravlja njene slabosti [50].

Zgoščevalna funkcija SHA-1, določena v dokumentu RFC 3174, kot rezultat vrne 160 bitni izvleček (angl. *hash value*) in je varnejša v primerjavi z zgoščevalno funkcijo MD5, saj preprečuje nastanek kolizij oziroma lahko do kolizije pride le teoretično [50]. Uporaba zgoščevalne funkcije SHA-1 je zelo razširjena, saj jo uporabljajo številne aplikacije in protokoli. Vendar so bile najdene določene napake z varnostnega stališča tudi v zgoščevalni funkciji SHA-1, kar je privedlo do potrebe po novi zgoščevalni funkciji oziroma družini zgoščevalnih funkcij SHA-2. Družina SHA-2, določena v dokumentu RFC 5754, zajema štiri zgoščevalne funkcije, ki vsebujejo številne spremembe v primerjavi z zgoščevalno funkcijo SHA-1 [51]. V to družino spadajo zgoščevalne funkcije: SHA-224, SHA-256, SHA-384 in SHA-512, kjer sta SHA-224 in SHA-384 okrnjeni verziji preostalih dveh. Dolžina izvlečka oziroma rezultat zgoščevalnih funkcij iz družine SHA-2 se giblje med 224 in 512 biti, kar je razvidno iz samega imena. Vse zgoščevalne funkcije iz družine SHA-2 so varne, saj je računsko neizvedljivo najti izvleček, ki bi ustrezal določenemu sporočilu oziroma je neizvedljivo najti dva različna sporočila, ki bi lahko imela isti izvleček. Uporaba zgoščevalnih funkcij SHA-2 ni tako razširjena kot uporaba zgoščevalnih funkcij SHA-1. Pri zgoščevalni funkciji SHA-1 še ni bilo najdenih kolizij, zato je uporaba teh še vedno večja.

Podobno kot v primeru zgoščevalne funkcije MD5 lahko tudi tu nastopi algoritem MAC z uporabo zgoščevalnih funkcij SHA-1 in SHA-2, kar privede do zgoščevalnih funkcij s ključem (npr. HMAC-SHA256). Na spodnji sliki 18 je razvidno, da zgoščevalna funkcija SHA-1 in SHA-2 ob istem vhodnem nizu generirata različen izvleček, če ju uporabimo s ključem in brez ključa.

```
SHA1 ("Testni niz") = ca2ffbc123ef1212251e0d5fa3c74a6b57778f96
HMAC-SHA1 ("kljuc", "Testni niz") = b0fb477e5946cf491d6c5476d994bcf641bf1fd1

SHA256 ("Testni niz") =
bfdc6728b44e6a31b13ecb578b6704a0f3b245a58496988683815a880c306062
HMAC-SHA256 ("kljuc", "Testni niz") =
ecbca3e2398302dde8b51bc1351487d93a48ccdb44d3de4a88c1d5aa24298071
```

Slika 18: Primer izvlečka zgoščevalne funkcije SHA-1, HMAC-SHA1, SHA-256 in HMAC-SHA256 z istim vhodnim nizom.

Kadar želimo onemogočiti vidnost uporabniškega gesla v čistopisu (angl. *cleartext*), lahko uporabimo zgoščevalno funkcijo SSHA, ki je varnejša kakor SHA, saj ta prejme dodano sol kot atribut zgoščevalne funkcije in tako onemogoča napad z mavričnimi tabelami (angl. *rainbow attack*).

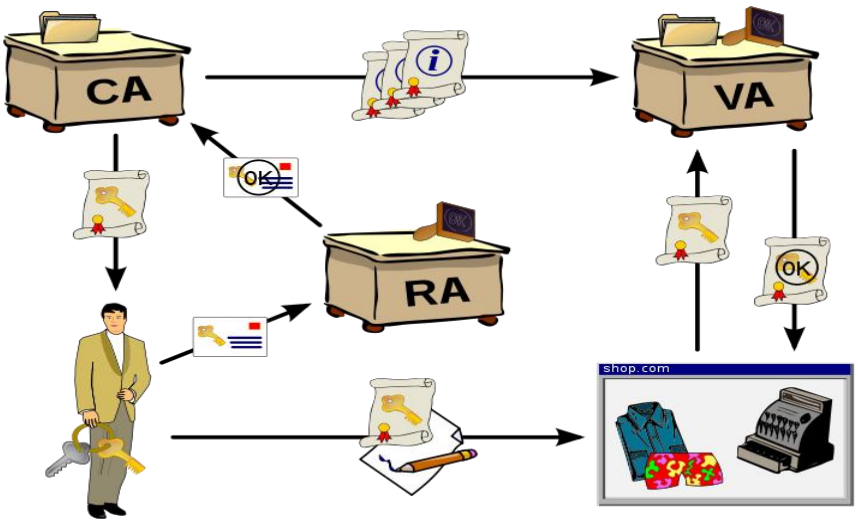
2.8 PKI

PKI predstavlja varno infrastrukturo, ki omogoča uporabnikom varno in zaupno izmenjavo podatkov prek medmrežja (angl. *internet*). Ta varna infrastruktura uporablja asimetrično kriptografijo oziroma kriptografijo z javnim ključem (angl. *public key cryptography*), ki je najbolj razširjena metoda za zagotavljanje avtentičnosti in zaupnosti sporočila. Infrastruktura PKI je sestavljena iz več delov, kjer ima vsak izmed njih določeno vlogo v sami infrastrukturi. Ti deli so naslednji [17, 34]:

- **CA** – izdaja certifikate določenim uporabnikom. Podrobnejši pregled vloge CA smo prikazali v podpoglavju, ki sledi naslednjemu,
- **RA** – preverja identiteto uporabnika, ki želi pridobiti certifikat, preden mu ga CA izda. (RA tako le olajša delo za CA),
- **imenik oziroma več imenikov** – v njem so varno shranjeni certifikati, ki vsebujejo javni ključ določenega uporabnika. Ta imenik mora biti dostopen vsem uporabnikom določene infrastrukture PKI. Gre namreč za kriptografijo z javnim ključem, kjer mora določen pošiljatelj pridobiti javni ključ prejemnika, kateremu želi poslati sporočilo z namenom šifriranja sporočila. Tako lahko pošiljatelj pridobi javni ključ iz imenika, ki vsebuje certifikate. V primeru določene organizacije oziroma podjetja je lahko ta imenik v lasti podjetja, ki vsebuje le certifikate svojih zaposlenih ter do njega lahko dostopajo le oni,

- **Sistem za upravljanje s certifikati** (angl. *certificate management system*) – s pomočjo tega sistema so certifikati objavljeni, podaljšani, preklicani in začasno ali trajno prekinjeni. CA požene ta sistem z namenom izvajanja svojih obveznosti.

Sama infrastruktura PKI se lahko uporablja v primeru zasebnih omrežij (angl. *private networks*), ki vključujejo tudi notranja omrežja podjetij, kjer si lahko udeleženci izmenjujejo sporočila na varen način. Prav tako se infrastruktura PKI uporablja za pridobitev kriptografskih ključev na varen način, ko želi določen uporabnik pridobiti par zasebnega in javnega ključa [17]. Par zasebnega in javnega ključa lahko uporabnik generira tudi sam. Primer infrastrukture PKI prikazuje spodnja slika 19, kjer uporabnik že ima generiran par zasebnega in javnega ključa ter pridobi certifikat. S pomočjo tega nato dostopa do spletne trgovine, ki preveri veljavnost certifikata, ki ga predloži uporabnik.



Slika 19: Primer pridobitve certifikata in nakup v spletni trgovini [54].

2.8.1 Certifikati

Certifikati so ključnega pomena v infrastrukturi PKI, saj s pomočjo teh posamezni uporabniki izkažejo svojo identiteto ter ga tako uporabljamo za avtentikacijo. Certifikat je elektronski dokument, ki vsebuje številne podatke, med katerimi so osnovni naslednji:

- naziv izdajatelja,
- osebni podatki, kot so: ime osebe oziroma organizacije, ime domene in naslov,
- javni ključ lastnika,

- digitalni podpis, ki ga izvede izdajatelj certifikata s svojim zasebnim ključem,
- čas veljavnosti certifikata.

Certifikat potrjuje, da je točno določena oseba, za katero vsebuje podatke, lastnik vsebovanega javnega ključa oziroma da ta pripada njemu. To je potrjeno z vsebovanim digitalnim podpisom izdajatelja certifikata, ki je CA. Ta digitalni podpis lahko prejemnik preveri ter ugotovi, ali gre za pravi digitalni podpis. S pomočjo digitalnega podpisa lahko prejemnik tudi preveri, ali gre za CA, ki je vredna zaupanja. Prav tako lahko prejemnik preveri javni ključ izdajatelja certifikata, identiteto lastnika in integriteto certifikata.

Kot je razvidno iz zgornjih naštetih podatkov, vsebujejo certifikati čas veljavnosti, ki se določi ob izdaji certifikata s strani CA. Določeni primeri lahko privedejo do preklica veljavnosti certifikata, čeprav se temu še ni iztekla veljavnost. To se lahko pojavi v primeru sprememb podatkov, pri kraji certifikata ali sumu, da lastnik certifikata ni tisti, za katerega se izdaja oziroma ta nima ustreznega para zasebnega ključa, ki pripada vsebovanemu javnemu ključu v certifikatu. CA občasno izdaja seznam preklicanih certifikatov (angl. *certificate revocation list*), ki je podpisan s strani CA ter je javno dostopen v javnem imeniku. Ko se uporabnik predstavi določenemu strežniku s certifikatom, ta poleg veljavnosti certifikata in podpisa preveri tudi, če se ta ne pojavi na seznamu preklicanih certifikatov s pomočjo serijskega števila, ki identificira določen certifikat znotraj seznama preklicanih certifikatov [5]. Na ta način se strežnik lahko prepriča, da je uporabnikov certifikat veljaven. Za razliko od preklicanih oziroma neveljavnih certifikatov so veljavni certifikati objavljeni oziroma shranjeni znotraj imenika ali več imenikov, ki so del infrastrukture PKI.

Poleg certifikatov, ki pripadajo končnim osebam (angl. *end entity certificates*), so tu še certifikati, namenjeni CA. Ti se delijo v tri razrede, in sicer [5]:

- **križni certifikati** (angl. *cross certificates*) − ti opisujejo zaupno razmerje med dvema CA,
- **lastno izdani certifikati** (angl. *self-issued certificates*) − so certifikati, kjer je izdajatelj in lastnik certifikata ista oseba,
- **lastno podpisani certifikati** (angl. *self-signed certificates*) − so primer lastno izdanega certifikata, kjer se lahko preveri digitalni podpis s pomočjo javnega ključa.

Sam format certifikatov oziroma njegovi vsebovani podatki so določeni s standardom X.509, ki definira format certifikatov. Zadnji standard X.509 inačice 3 (X.509v3) omogoča poleg standardnih polj še dodatna razširljiva polja, s katerimi lahko dodamo več podatkov. Razširitvena polja se pogosto nanašajo na dodatne informacije o lastniku certifikata ali o

vsebovanem javnem ključu. Določena razširitvena polja mora podpirati CA in jih izpolniti ob izdaji certifikata, medtem ko je podpora za nekatera razširitvena polja neobvezna. Primer razširitvenega polja, ki mora biti podprto oziroma vključeno s strani CA, je polje `key usage` [5]. Ta definira namen uporabe vsebovanega javnega ključa, saj je ta lahko uporaben za več operacij. Predhodnik inačice 3 (X.509v2) teh dodatnih razširljivih polj ne omogoča, vendar v primerjavi s prvo inačico (X.509v1), vsebuje dva dodana polja. Format certifikatov X.509v3 je določen v dokumentu RFC 5280, kjer je poleg tega določen tudi format X.509v2 za seznam preklicanih certifikatov.

2.8.2 CA

Pomemben del infrastrukture PKI predstavlja CA, saj ta izdaja certifikate. CA pred izdajo certifikata preveri identiteto uporabnika, ki želi pridobiti certifikat, saj mora ta izkazati svojo identiteto. Uporabnik izkaže svojo identiteto z ustreznimi podatki, ki potrjujejo njegovo identiteto, ter s tem dokaže, da je tisti, za katerega se izdaja. CA podpiše izdan certifikat ter s tem prepreči spreminjanje vsebovanih podatkov v certifikatu. Uporabnik pridobi certifikat, ki spada v določen razred, s pomočjo katerega je določena stopnja preverjanja identitete. Ti razredi certifikatov imajo naslednje določene stopnje preverjanja identitete [17]:

- **prvi razred certifikatov** – certifikat iz tega razreda lahko pridobimo le s posredovanjem svojega elektronskega naslova,
- **drugi razred certifikatov** – certifikat iz tega razreda lahko pridobimo s posredovanjem dodanih osebnih podatkov,
- **tretji razred certifikatov** – certifikat iz tega razreda lahko pridobimo le v primeru, ko je bila identiteta uporabnika preverjena oziroma jo je ta izkazal,
- **četrti razred certifikatov** – te certifikate uporabljajo vlade in organizacije, ki potrebujejo visoko stopnjo preverjanja.

CA nastopi kot zaupanja vredna tretja stran, saj ji zaupa tako uporabnik, ki pridobi certifikat, kakor tudi določen ponudnik storitve (angl. *network access server*), ki pridobi certifikat določenega uporabnika, saj mu ga ta predloži pri samem dostopu. CA zagotavlja pravilnost podatkov v certifikatu s svojim digitalnim podpisom.

Par javnega in zasebnega ključa lahko, namesto CA, generira tudi sam uporabnik, ki želi pridobiti certifikat. Ta v tem primeru pošlje zahtevo za pridobitev certifikata s priloženim javnim ključem, s podatki o svoji identiteti in podatki o CA. Gre za varnejši primer pridobitve

certifikata s stališča uporabnika, saj se ta prepriča, da je zasebni ključ znan le njemu (primer z zgornje slike 19).

CA lahko seveda nastopi tudi večkrat, kjer lahko med njimi veljajo določena zaupna razmerja. To privede do tako imenovane verige zaupanja, kjer lahko zaupamo tudi komurkoli nižje v verigi. Razmerje vzpostavi določena CA z izdajo certifikata drugi CA. Poznamo dva naslednja razmerja:

- **certifikacijska hierarhija** (angl. *certification hierarchy*) – tu nastopita dva tipa CA, in sicer korenska avtoriteta (angl. *superior CA*) in organizacijska CA (angl. *subordinate CA*). Korenska avtoriteta je izdajatelj certifikata organizacijski CA. Pod organizacijsko CA se nahaja uporabnik, ki tako zaupa obema nad seboj,
- **križno certificiranje** (angl. *cross-certification*) – ta lahko nastopi kot enostransko ali dvostransko. Pri enostranskem križnem certificiranju gre za križno certificiranje ene CA z drugo, vendar ne v obratni smeri. V primeru dvostranskega certificiranja gre za križno certificiranje med dvema CA v obe smeri (v tem primeru si obe medsebojno izdata certifikat). Tako nastopi zaupanje med več CA iz drugih hierarhij.

2.8.3 Protokol SSL/TLS

Protokol SSL oziroma TLS je primer varnostnega protokola, ki leži med transportno in aplikacijsko plastjo. Na ta način protokol omogoča varen vtič ter s tem varno komunikacijo na transportni plasti. Njegov namen je vzpostavitev varne povezave med odjemalcem in strežnikom. Protokol SSL je bil preimenovan v TLS, vendar inačice protokola TLS temeljijo na zadnji inačici protokola SSL 3.0 [44]. Tako se bomo v nadaljevanju posvetili nasledniku protokola SSL, ki je TLS.

Glavni cilj protokola TLS je zagotovitev zasebnosti in zanesljivosti med udeležencema v komunikaciji, ki si izmenjujeta sporočila. Protokol TLS je neodvisen od protokola na aplikacijski plasti, kar je njegova velika prednost. Sam protokol zagotavlja varno povezavo, in sicer z naslednjimi lastnostmi [6, 7]:

- zasebna povezava med odjemalcem in strežnikom s pomočjo simetričnih kriptografskih metod za šifriranje sporočil: med najpogosteje uporabljene simetrične algoritme sodijo AES, 3DES in RC4. Šifriranje se uporablja po začetni fazi rokovanja (angl. *handshake*), kjer se prične izmenjava podatkov med odjemalcem in strežnikom,

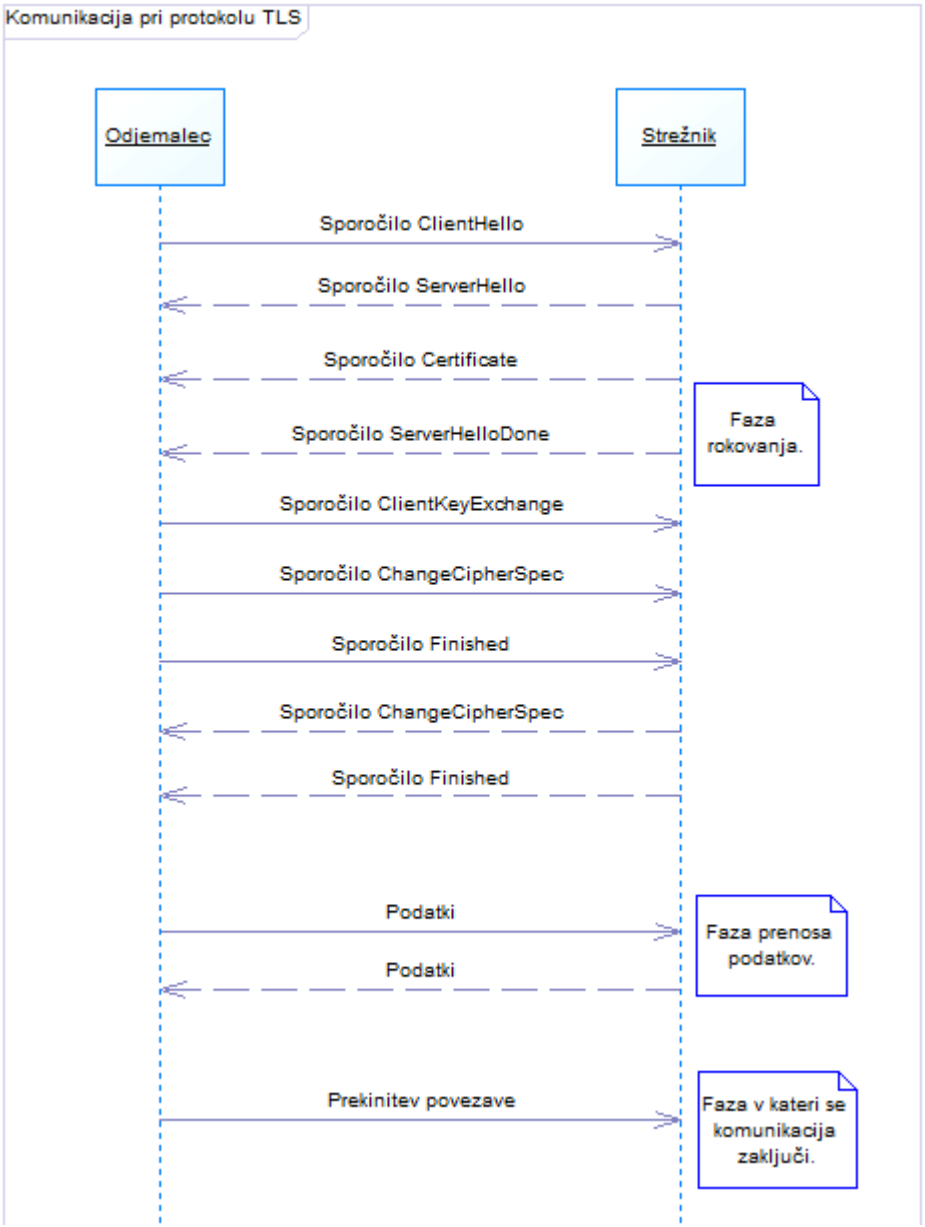
- zanesljiva povezava med odjemalcem in strežnikom, z uporabo zgoščevalnih funkcij HMAC: za vsako preneseno sporočilo se na prejemnikovi strani preveri njegova integriteta, kjer se najpogosteje uporabi zgoščevalna funkcija SHA-1,
- identiteta odjemalca oziroma strežnika se lahko preveri z uporabo asimetrične kriptografije oziroma kriptografije z javnim ključem (angl. *public key cryptography*), kjer je najpogosteje uporabljen algoritem RSA. Preverjanje identitete odjemalca ni obvezno, medtem ko je zahtevano preverjanje identitete strežnika. Identiteto izkažeta s pomočjo predloženega certifikata, ki vsebuje javni ključ lastnika. Poleg tega se kriptografija z javnim ključem pri protokolu TLS uporablja za izmenjavo ključev (angl. *key exchange*). Poleg algoritma RSA lahko v tem primeru nastopu tudi Diffie-Hellman izmenjava ključev.

Kadar želi odjemalec vzpostaviti varno povezavo z uporabo protokola TLS, mora o tem obvestiti strežnik. To lahko izvede z uporabo drugih vrat (angl. *port*), ki je značilna za protokol TLS, ali s posebnim ukazom, ki sporoča strežniku preklop na protokol TLS. Celotno komunikacijo med odjemalcem in strežnikom v primeru uporabe protokola TLS lahko razdelimo v tri faze.

Komunikacija se prične s fazo rokovanja, ki nastopi pred pričetkom izmenjave podatkov med odjemalcem in strežnikom, kjer se izvede postopek, potreben za vzpostavitev varne povezave med njima. Postopek se prične na strani odjemalca, ki pošlje sporočilo `ClientHello`, ki vsebuje najvišjo podprto inačico protokola TLS, naključno število oziroma žeton (angl. *token*) in seznam podprtih algoritmov [44]. Strežnik se na to sporočilo odzove s sporočilom `ServerHello`, ki vsebuje izbrano inačico protokola TLS, naključno število oziroma žeton in izbran algoritem s seznama podprtih algoritmov. Takoj po sporočilu `ServerHello` strežnik pošlje svoj certifikat z namenom avtentikacije in nato še sporočilo `ServerHelloDone`, s katerim označuje konec rokovanja z njegove strani. Nato odjemalec v primeru izbire algoritma RSA za avtentikacijo strežnika in določanje ključev pošlje strežniku sporočilo `ClientKeyExchange`. Ta vsebuje generiran ključ (angl. *premaster secret*) s strani odjemalca, ki ga šifrira z javnim ključem strežnika, saj ga pridobi iz njegovega certifikata. Strežnik nato uporabi svoj zasebni ključ ter dešifrira pridobljeni generirani ključ. Na tem mestu strežnik in odjemalec pretvorita generirani ključ v glavni ključ (angl. *master secret*) s pomočjo generiranega ključa in njunih žetonov. Z uporabo glavnega ključa in njunih žetonov oba izračunata še ključe, potrebne za naslednjo fazo (prenos podatkov). Oba izpeljeta štiri ključe, in sicer: šifrirni ključ odjemalca in strežnika in MAC ključ odjemalca in strežnika. Na koncu si odjemalec in strežnik izmenjata sporočilo `ChangeCipherSpec` in za tem še sporočilo `Finished`, ki vsebuje vrednost zgoščevalne funkcije HMAC, ki je izvedena nad vsemi sporočili v fazi rokovanja z izjemo sporočila `Finished`. V primeru odjemalca se

izvleček (angl. *hash value*) izračuna na podlagi odjemalčevih poslanih sporočil, medtem ko se v primeru strežnika izvleček izračuna na podlagi strežnikovih poslanih sporočil. Tako lahko odjemalec oziroma strežnik ugotovita, ali je v tej fazi prišlo do posega napadalca, saj bi ta lahko izbrisal močnejši šifrirni algoritem s seznama podprtih algoritmov, ki ga pošlje odjemalec strežniku. Na ta način se onemogoči tovrstni napad in zagotovi integriteto vseh prenesenih sporočil v fazi rokovanja.

Po fazi rokovanja sledi faza prenosa podatkov med odjemalcem in strežnikom, in sicer v obliki zapisov. Sporočila oziroma podatki so združeni v zapise, kjer vsak zapis vsebuje določen obseg podatkov. Vsakemu zapisu je navadno priložena vrednost zgoščevalne funkcije HMAC, izvedene nad posameznim zapisom z uporabo ključa MAC (na ta način lahko prejemnik preveri integriteto posameznega zapisa). Vrednost zgoščevalne funkcije je šifrirana skupaj z zapisom, z uporabo šifrirnega ključa (oba uporabljena ključa sta izračunana v začetni fazi rokovanja). Zapisu je še pripeta glava zapisa, ki vsebuje informacijo o vrsti vsebine, inačici protokola TLS in dolžini [44]. Po končanem prenosu podatkov med odjemalcem in strežnikom se mora najprej zaključiti varna povezava med njima. To se zgodi v zadnji fazi, kjer se povezava med njima zaključi (celoten opisan potek komunikacije, z vsemi omenjenimi sporočili, prikazuje spodnja slika 20).



Slika 20: Primer komunikacije pri protokolu TLS.

V fazi rokovanja lahko nastopi tudi avtentikacija odjemalca, ki je neobvezna in se izvede le na zahtevo strežnika (tu tako nastopi vzajemna avtentikacija (angl. *mutual authentication*)). V tem primeru strežnik pošlje sporočilo `CertificateRequest`, po tem ko pošlje svoj certifikat ter s tem zahteva certifikat odjemalca, ki mu ga ta pošlje v sporočilu `ClientCertificate` po sporočilu strežnika `ServerHelloDone`. Odjemalec po poslanem sporočilu `ClientKeyExchange` pošlje še sporočilo `CertificateVerify`, s katerim zagotovi preverjanje svojega certifikata s strani strežnika [44]. Sporočilo vsebuje podpis vseh sporočil od začetka rokovanja do te faze z izjemo sporočila `CertificateVerify`. Podpis odjemalec opravi s svojim zasebnim ključem, tako lahko

strežnik preveri podpis z odjemalčevim javnim ključem, ki ga pridobi iz odjemalčevega certifikata.

Poleg tega se lahko odjemalec in strežnik odločita za nadaljevanje prejšnje seje, da bi se izognili računsko zahtevnim operacijam asimetrične kriptografije [44]. V tem primeru odjemalec pošlje sporočilo `ClientHello` z dodatnim atributom, imenovanim `sessionID`, ki enolično označuje sejo. Strežnik nato preveri obstoj atributa `sessionID` v pomnilniku ter v primeru obstoja pošlje odjemalcu sporočilo `ServerHello`, ki vsebuje atribut `sessionID` (s tem strežnik potrdi odjemalcu, da se bo izvedlo nadaljevanje seje). Na tem mestu imata odjemalec in strežnik že glavni ključ in žetone za generiranje naslednjih ključev. Nato si odjemalec in strežnik izmenjata le še sporočilo `ChangeCipherCpec` in takoj za tem še sporočilo `Finished`. Na ta način se tako skrajša faza rokovanja, s čimer tudi prihranimo čas.

2.9 Varnostni vidiki protokola RADIUS in LDAP ter ranljivosti protokola KERBEROS

V tem podpoglavju smo navedli nekaj zanimivih citatov, ki se nanašajo na naše pregledano področje. Na začetku smo navedli citat, ki se nanaša na protokol RADIUS in govori o možni uporabi protokola RADIUS preko protokola TLS, kar seveda poveča stopnjo varnosti protokola RADIUS. Drugi citat se nanaša na protokol LDAP in govori o mehanizmu SASL, ki je vključen v tretjo inačico protokola LDAP in o njem nismo dosti govorili. Podpoglavje smo zaključili s citati, ki se nanašajo na protokol KERBEROS in opisujejo nekatere možne napade nanj ter nato navedli rešitve za določen napad. Te napade namreč nismo omenili, ko smo govorili o protokolu KERBEROS.

Protokol RADIUS v osnovi ne omogoča zanesljivega prenosa podatkov, saj temelji na transportnem protokolu UDP. Prav tako ne omogoča visoke stopnje varnosti, saj njegova varnost temelji na algoritmu MD5, ki danes ne velja za varno. Protokol RADIUS se lahko uporabi tudi preko protokola TLS in se tako oblaži morebitne napade na algoritem MD5. Prav tako tu nastopi uporaba transportnega protokola TCP, kar omogoča zanesljiv prenos podatkov brez izgube paketov, kot je to možno pri transportnem protokolu UDP. Poleg tega tu nastopijo bolj sodobne metode za preverjanje identitete odjemalca in strežnika RADIUS, v določeni komunikaciji (namesto naslova IP in skupne skrivnosti, se lahko uporabi certifikat). Vse to potrjuje citat, podan v nadaljevanju (pridobljen iz prvega poglavja dokumenta RFC 6614), ki govori o možni uporabi protokola RADIUS prek protokola TLS, kar zagotavlja varno in zanesljivo komunikacijo med odjemalcem in strežnikom RADIUS:

"The RADIUS protocol [RFC2865] is a widely deployed authentication and authorization protocol. The supplementary RADIUS Accounting specification [RFC2866] provides accounting mechanisms, thus delivering a full Authentication, Authorization, and Accounting (AAA) solution. However, RADIUS is experiencing several shortcomings, such as its dependency on the unreliable transport protocol UDP and the lack of security for large parts of its packet payload. RADIUS security is based on the MD5 algorithm, which has been proven to be insecure.

The main focus of RADIUS over TLS is to provide a means to secure the communication between RADIUS/TCP peers using TLS. The most important use of this specification lies in roaming environments where RADIUS packets need to be transferred through different administrative domains and untrusted, potentially hostile networks. An example for a worldwide roaming environment that uses RADIUS over TLS to secure communication is "eduroam", see [eduroam].

There are multiple known attacks on the MD5 algorithm that is used in RADIUS to provide integrity protection and a limited confidentiality protection (see [MD5-attacks]). RADIUS over TLS wraps the entire RADIUS packet payload into a TLS stream and thus mitigates the risk of attacks on MD5.

Because of the static trust establishment between RADIUS peers (IP address and shared secret), the only scalable way of creating a massive deployment of RADIUS servers under the control of different administrative entities is to introduce some form of a proxy chain to route the access requests to their home server. This creates a lot of overhead in terms of possible points of failure, longer transmission times, as well as middleboxes through which authentication traffic flows. These middleboxes may learn privacy-relevant data while forwarding requests. The new features in RADIUS over TLS obsolete the use of IP addresses and shared MD5 secrets to identify other peers and thus allow the use of more contemporary trust models, e.g., checking a certificate by inspecting the issuer and other certificate properties [16]*."*

Ko smo govorili o protokolu LDAPv3, smo navedli številne možnosti avtentikacije odjemalca, ki jih podpira protokol LDAPv3, med katerimi je tudi splošna avtentikacija in varnostna plast (angl. *simple authentication and security layer*). Protokol LDAPv3 podpira uporabo tako imenovanega mehanizma SASL, ki predstavlja razširljivo varnostno shemo. Ta mehanizem omogoča dodajanje novih avtentikacijskih mehanizmov ter odjemalcu in strežniku omogoča predhodno pogajanje o izbiri avtentikacijskega mehanizma. Zato je pri protokolu LDAPv3 z uporabo avtentikacijskega mehanizma SASL omogočeno dodajanje novih avtentikacijskih mehanizmov brez preoblikovanja protokola LDAPv3. Mehanizem SASL prav tako omogoča varnostno plast (angl. *security layer*), s katero je zagotovljeno šifriranje poslanih podatkov v sami komunikaciji. Varnostna plast lahko tu nastopi z uporabo

protokola SSL/TLS. Vse to je razvidno tudi iz naslednjega citata (pridobljenega iz prvega poglavja dokumenta RFC 4422), ki govori o mehanizmu SASL:

"The Simple Authentication and Security Layer (SASL) is a framework for providing authentication and data security services in connection-oriented protocols via replaceable mechanisms. SASL provides a structured interface between protocols and mechanisms. SASL also provides a protocol for securing subsequent protocol exchanges within a data security layer. The data security layer can provide data integrity, data confidentiality, and other services.

SASL's design is intended to allow new protocols to reuse existing mechanisms without requiring redesign of the mechanisms and allows existing protocols to make use of new mechanisms without redesign of protocols.

SASL is conceptually a framework that provides an abstraction layer between protocols and mechanisms as illustrated in the following diagram.

It is through the interfaces of this abstraction layer that the framework allows any protocol to utilize any mechanism [10]."

Čeprav sistem Kerberos velja teoretično za varen sistem, so v praksi možni nekateri napadi. V nadaljevanju smo prikazali nekaj možnih napadov v okviru protokola KERBEROS. Navajani spodnji citat (vzet iz desetega poglavja dokumenta RFC 4120) opisuje neodpornost strežnika KERBEROS na številne napade z onemogočanjem storitve (angl. *denial of service*), kar lahko privede do onemogočanja delovanja določenih storitev, ki uporabljajo protokol KERBEROS za proces avtentikacije.

"Denial of service attacks are not solved with Kerberos. There are places in the protocols where an intruder can prevent an application from participating in the proper authentication steps. Because authentication is a required step for the use of many services, successful denial of service attacks on a Kerberos server might result in the denial of other network services that rely on Kerberos for authentication. Kerberos is vulnerable to many kinds of denial of service attacks: those on the network, which would prevent clients from contacting the KDC; those on the domain name system, which could prevent a client from finding the IP address of the Kerberos server; and those by overloading the Kerberos KDC itself with repeated requests [11]."

Ker sistem Kerberos ne preprečuje tovrstnega napada, je priporočljiva uporabo požarnega zidu (angl. *firewall*), da bi s tem zavarovali naše omrežje skupaj s centrom za distribucijo ključev (angl. *key distribution center*). Prav tako lahko vključitev dodatnih centrov za distribucijo ključev ublaži tovrstni napad.

Prav tako napad s slovarjem (angl. *dictionary attack*) in napad z grobo silo (angl. *brute force attack*) nista razrešena v okviru protokola KERBEROS. V primeru slabega gesla uporabnika lahko napadalec poskuša z različnimi vnosi, ki bi lahko uspešno dešifrirali začetno vstopnico (angl. *ticket granting ticket*) za določenega uporabnika. Zato je priporočeno, da uporabimo predhodno avtentikacijo (angl. *pre-authentication*), kot to sugerira naslednji citat (vzet iz desetega poglavja dokumenta RFC 4120):

"Unless pre-authentication options are required by the policy of a realm, the KDC will not know whether a request for authentication succeeds. An attacker can request a reply with credentials for any principal. These credentials will likely not be of much use to the attacker unless it knows the client's secret key, but the availability of the response encrypted in the client's secret key provides the attacker with ciphertext that may be used to mount brute force or dictionary attacks to decrypt the credentials, by guessing the user's password. For this reason it is strongly encouraged that Kerberos realms require the use of pre-authentication. Even with pre-authentication, attackers may try brute force or dictionary attacks against credentials that are observed by eavesdropping on the network [11]."

Poleg omenjenih napadov na sistem Kerberos je možen tudi napad s ponovitvijo (angl. *replay attack*), vendar je za napad na voljo nekaj zaščit, ki so že vgrajene v zadnjo implementacijo protokola KERBEROS. Napadalec lahko namreč prisluškuje določeni komunikaciji ter pridobi kopije sporočil, ki jih lahko kasneje ponovno pošlje. Ko želi odjemalec dostopati do Kerberos storitve, pošlje svojo začetno vstopnico do te storitve z namenom odobritve dostopa. Napadalec lahko tu prestreže sporočilo ter pridobi šifrirano začetno vstopnico, ki jo lahko pošlje kasneje ponovno in se tako izdaja za odjemalca. Za preprečevanje napada s ponovitvijo so v protokolu KERBEROS implementirane naslednje zaščite (navedene v treh navedenih citatih), ki jih predstavlja avtor Garman v šestem poglavju svojega dela:

"Address field in tickets

When a client requests a ticket from the KDC, it lists the network addresses from which the ticket is valid. For example, if the IP address of the workstation is 192.168.1.1, the workstation would fill that address in the address field in the ticket request, and the KDC would copy that into the ticket it returns to the workstation. This protection solves the problem of an attacker who attempts to replay a valid ticket on a workstation not listed in the ticket's address field. However, this protection is not enough to completely thwart replay attacks; the address field my be left blank, and the ticket would be valid for all addresses. Or, if the attacker had access to a machine listed in the addresses field, he could login and replay the ticket from there [8]."

˝Time-based authenticators

Since address-based security is imperfect, Kerberos employs another scheme to thwart replay attacks. Every time a client wishes to use a Kerberized service, she generates an authenticator, which is submitted with the ticket to the service for authentication. The authenticator contains nothing but a timestamp, encrypted with the session key generated by the KDC for this particular ticket exchange. When the service receives this authenticator, it checks the decrypted timestamp against its system clock. If the two times differ by more than five minutes in either direction, the service will reject the ticket and refuse to authenticate the user. The 5-minute time period is designed to allow for some variation between different clocks on the network. However, even five minutes may be more than enough time for an attacker to replay valid tickets, especially if he employs an automated program to capture and replay tickets [8].˝

˝Replay caches

The last line of defense that Kerberos has against replay attacks is the replay cache. Both the Kerberos v4 and v5 protocols include the time-based authenticator. Kerberos v5 introduces the replay cache to avoid attackers reusing tickets in the short time period that authenticators are valid. Every Kerberized service maintains a cache of the authenticators it has recently received. When the service receives an authenticator, it checks the replay cache. If the service finds a copy of the authenticator already in the replay cache, it rejects the request. Otherwise, the service accepts the request and adds the authenticator to the replay cache to validate further requests [8].˝

Kot je razvidno iz zgornjih treh citatov, vključuje protokol KERBEROS inačice štiri dve rešitvi za preprečevanje napadov s ponavljanjem, vendar ti dve rešitvi ne zagotavljata odpravo tovrstnega napada. Tako protokol KERBEROS inačice pet vključuje uporabo posebnega predpomnilnika (angl. *replay cache*), s pomočjo katerega se onemogoči napad s ponavljanjem.

3 Pregled izbranih avtentikacijskih strežnikov

V tem poglavju smo prikazali primere avtentikacijskih strežnikov, ki predstavljajo konkretne produkte ter se danes pogosto uporabljajo. Na koncu poglavja smo prikazali tudi primerjavo dveh med seboj podobnih produktov, in sicer OpenLDAP in Active Directory. Kot smo opazili, produkt Active Directory zajema veliko več kot le imeniško storitev.

3.1 FreeRADIUS

FreeRADIUS predstavlja konkretni produkt oziroma projekt, ki vključuje strežnik RADIUS, knjižnico odjemalcev (angl. *client library*), knjižnico PAM in modul Apache [22]. Projekt FreeRADIUS se danes nanaša na strežnik RADIUS, ki je glavna komponenta celotnega projekta. FreeRADIUS tako implementira strežnik RADIUS skupaj z vsemi njegovimi razširitvami, določenimi v številnih dokumentih RFC, ter tako predstavlja strežnik tipa AAA. Strežnik FreeRADIUS velja za najbolj priljubljeno implementacijo strežnika RADIUS, saj je njegova uporaba široko razporejena po vsem svetu. Primer njegove uporabe je v brezžičnem omrežju EDUROAM, kjer je potreben tovrstni strežnik tipa AAA. Poleg tega se strežnik FreeRADIUS uporablja v številnih podjetjih in tudi pri internetnih ponudnikih storitev (angl. *internet service providers*). Strežnik FreeRADIUS ima številne lastnosti, s pomočjo katerih velja danes za najbolj priljubljen oziroma najbolj uporabljan strežnik RADIUS [15]:

- **je odprtokoden** (angl. *open source*) – kar omogoča spreminjanje, razširitev in popravek česarkoli v skladi s potrebami. Poleg tega je produkt FreeRADIUS dostopen brezplačno,
- **je zmogljiv** – je eden izmed najhitrejših tovrstnih produktov, saj je njegova konfiguracija shranjena v glavnem pomnilniku. Zmogljivost strežnika FreeRADIUS je odvisna od dejavnikov, kot so: izbira podatkovne baze, zmogljivost strojne opreme, uporabe avtentikacijskega protokola EAP in omrežne hitrosti,
- **je modularen** (angl. *modular*) – strežnik FreeRADIUS vsebuje številne module, ki prinašajo določene lastnosti samemu produktu. Poleg modulov, ki so že vključeni v produkt, lahko vključimo številne druge obstoječe module, med katerimi so lahko tudi lastno kreirani. Na ta način je omogočeno enostavno dodajanje novih lastnosti ter tudi brisanje nepotrebnih, saj lahko določen modul odstranimo. Primer obstoječega modula, ki ga lahko vključimo, je modul FreeRADIUS-MySQL, s katerim je strežniku omogočeno shranjevanje uporabnikov v podatkovno bazo MySQL,

- **številni podprti avtentikacijski protokoli** – strežnik lahko avtenticira odjemalca s pomočjo enostavnih avtentikacijskih protokolov, kot so: PAP, CHAP, MS-CHAP in MS-CHAPv2. Poleg teh lahko strežnik avtenticira odjemalca z uporabo avtentikacijskega protokola EAP. Podprte so številne metode EAP, kot so: MD5, TLS, PEAP in TTLS,
- **podpora virtualnih strežnikov** – po privzetih nastavitvah strežnik FreeRADIUS omogoča dva virtualna strežnika, in sicer `default` ter `inner-tunnel`. Virtualni strežnik `default` obdeluje vse zahteve, za katere ni določen virtualni strežnik, medtem ko virtualni strežnik `inner-tunnel` obdeluje EAP zahteve, kot je EAP z metodo TTLS (EAP-TTLS),
- **razpoložljivost** (angl. *availability***)** – strežnik FreeRADIUS je dostopen za različne operacijske sisteme, med katerimi je tudi Windows. V primeru operacijskega sistema Linux je ta vključen v seznam dostopnih paketov,
- **podpora več podatkovnih baz** – strežnik FreeRADIUS namreč podpira številne podatkovne baze, med katere sodijo: MySQL, LDAP in tekstovna datoteka. Tako lahko strežnik FreeRADIUS namesto tekstovne datoteke uporabi zunanji vir, kot je imeniška storitev LDAP z namenom avtentikacije uporabnika.

Poleg vseh naštetih dobrih lastnosti strežnika FreeRADIUS,vsebuje ta tudi dve slabi lastnosti, in sicer kompleksnost ter ranljivost [15]. Kompleksnost predstavlja edino resno slabost, saj nudi FreeRADIUS številne opcije konfiguracij, ki lahko ob nepazljivosti privedejo do večjih težav. Druga omenjana slabost ni tako kritična, saj gre za ranljivosti, ki so prisotne v določenih inačicah strežnika FreeRADIUS. Te ranljivosti so bile pri novejših inačicah odstranjene.

3.2 OpenLDAP

Produkt OpenLDAP implementira strežnik LDAP ter velja za enega izmed hitrejših in priljubljenih tovrstnih produktov. Strežnik OpenLDAP je prav tako kot FreeRADIUS odprtokoden (angl. *open source*) in brezplačen, zato je dostopen za številne operacijske sisteme, med katerimi je tudi Windows. Številne distribucije operacijskega sistema Linux vključujejo uporabo produkta OpenLDAP, da bi podprle strežnik LDAP. Celoten produkt vključuje naslednje komponente [2, 4, 43]:

- **strežnik LDAP** − imenovan SLAPD, ki odjemalcem omogoča dostop do podatkov. SLAPD tako posluša na določenih vratih (angl. *port*) ter sprejema zahteve odjemalcev in na njih ustrezno reagira,

- **orodja za neposredni dostop do imeniške storitve LDAP** (angl. *utilities*) – ta orodja ne uporabljajo protokola LDAP za dostop do strežnika LDAP, saj do njega dostopajo neposredno. Med ta orodja sodijo ukazi, kot so: `slapadd`, `slappasswd` in `slaptest` ter se uporabljajo s strani administratorja za opravljanje njegovih nalog nad strežnikom LDAP,
- **knjižnice** (angl. *client libraries*) – produkt vključuje več knjižnic, ki odjemalcem oziroma aplikacijam LDAP zagotavljajo uporabo protokola LDAP,
- **orodja, namenjena odjemalcem** (angl. *tools*) − za razliko od orodij za neposreden dostop do imeniške storitve LDAP, uporabljajo ta orodja protokol LDAP za dostop do strežnika LDAP. Orodja so namenjena odjemalcem, s pomočjo katerih izvajajo določene operacije nad strežnikom LDAP. Med ta orodja sodijo ukazi, kot so: `bind`, `ldapadd` in `ldapmodify`.

Strežnik OpenLDAP izhaja v več inačicah, kjer starejše ne podpirajo protokola LDAPv3. Inačica OpenLDAP 2.0 in naslednje podpirajo protokol LDAPv3, medtem ko starejše inačice ne podpirajo protokola LDAPv3, temveč protokol LDAPv2. Inačice OpenLDAP 2.0 in naslednje podpirajo tako odjemalce s protokolom LDAPv2 in tudi s protokolom LDAPv3 [4].

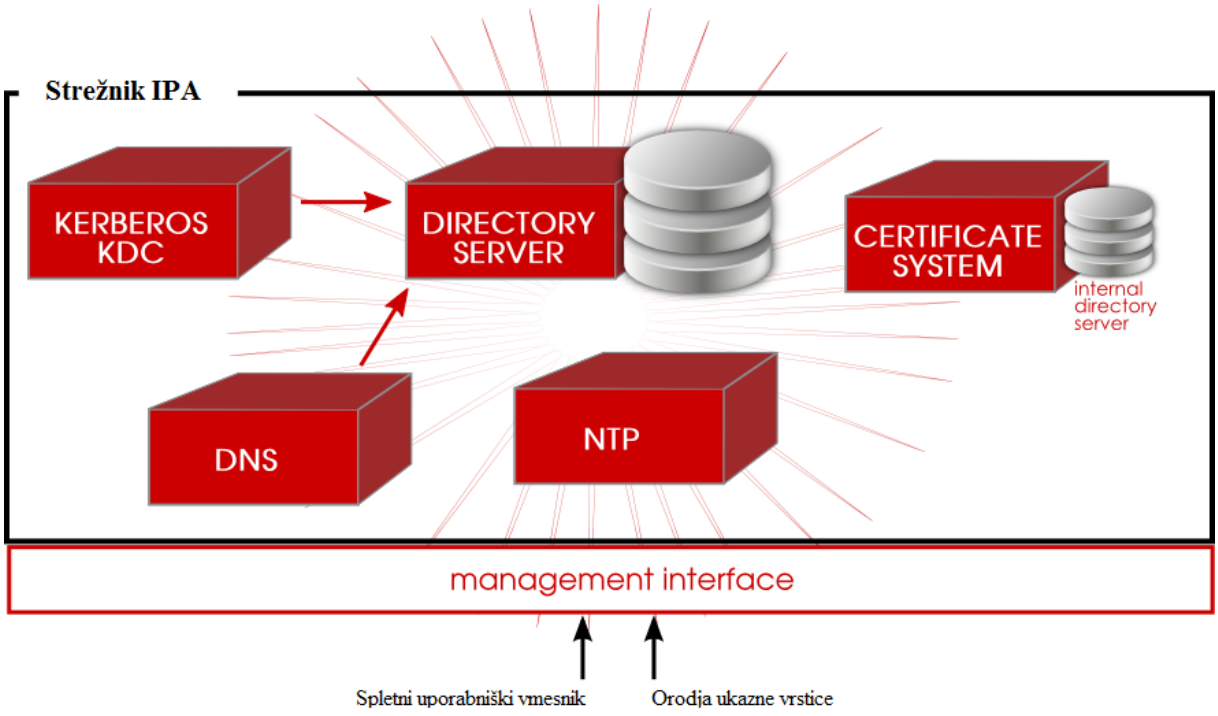
3.3 FreeIPA

FreeIPA je odprtokodni (angl. *open source*) produkt, ki definira domeno, v katero so vključeni strežniki in odjemalci ter tako na osnovni ravni nadzira domeno [21]. Udeleženci v domeni strežniki in odjemalci) si delijo storitve, ki se upravljajo centralno s strani administratorja domene FreeIPA. Sam namen produkta FreeIPA je zagotavljanje centralnega upravljanja z identitetami znotraj domene FreeIPA na preprost način. Produkt se nanaša tudi na strežnik FreeIPA, ki združuje več različnih storitev ter v glavnem služi kot avtentikacijski strežnik. Te storitve delujejo kot celota pod nadzorom strežnika FreeIPA.

Glavni cilj produkta FreeIPA je poenostavitev oziroma olajšanje dela administratorja v domeni FreeIPA. Ta lahko upravlja z vsemi storitvami in uporabniki znotraj domene na enem mestu (centralno). Ta konfiguracija se lahko enakomerno porazdeli med več računalniki v domeni FreeIPA ter tako administratorju ni treba upravljati vsake izmed storitev v domeni ločeno oziroma lokalno. Produkt FreeIPA združuje več različnih storitev, ki so integrirane v celoto, nad njimi se uporablja le ena množica orodij z namenom upravljanja s strani administratorja. Storitve, ki jih združuje strežnik FreeIPA, so naslednje [21]:

- **imeniška storitev 389** (angl. *389 directory server*) – predstavlja primer generične imeniške storitve, ki lahko hrani vse vrste podatkov ter temelji na imeniški storitvi LDAP. Zato je glavni namen imeniške storitve 389 hranjenje podatkov. V okviru strežnika FreeIPA ta imeniška storitev hrani vse podatke v povezavi s protokolom KERBEROS kot tudi vse ostale podatke v domeni FreeIPA (med te podatke sodijo podatki o uporabniških računih, skupinah, storitvah, itd.),
- **Kerberos center za distribucijo ključev** (angl. *Kerberos key distribution center*) – ta omogoča avtentikacijo s pomočjo protokola KERBEROS, kjer nastopi uporaba vstopnic (angl. *tickets*). Uporabnik, ki želi dostopati do določene storitve znotraj domene FreeIPA, mora imeti ustrezno in veljavno vstopnico, saj storitev, ki uporablja protokol KERBEROS za avtentikacijo (angl. *Kerberos-aware service*), preveri veljavnost te vstopnice,
- **NTP** – storitev NTP se v domeni FreeIPA uporablja z namenom sinhronizacije sistemskega datuma in ure med strežniki in odjemalci. S strežnikom NTP odjemalci komunicirajo s pomočjo protokola NTP ter tako opravijo sinhronizacijo sistemskega datuma in ure, saj strežnik NTP nudi točen datum in čas odjemalcem. Sinhronizacija datuma in ure je pomembna zlasti, ko nastopijo vstopnice protokola KERBEROS, saj je njihova veljavnost določena časovno. Kadar se čas med odjemalcem in strežnikom razlikuje za več, kot je dovoljeno, se vstopnica razveljavi. V domeni FreeIPA se pred opravljanjem katerekoli operacije izvrši sinhronizacija datuma in ure, kar preprečuje omenjeno nevšečnost,
- **certifikacijski sistem Dogtag** (angl. *Dogtag certificate system*) – s pomočjo tega sistema je vključena CA, ki izdaja certifikate strežnikom, storitvam in odjemalcem v domeni FreeIPA. CA lahko nastopi kot korenska avtoriteta (angl. *superior CA*) ali kot organizacijska CA (angl. *subordinate CA*). Izdani certifikati se lahko uporabljajo skupaj z vstopnicami s strani protokola KERBEROS. Ta lahko za postopek avtentikacije uporabi tudi certifikate poleg vstopnic, saj določene storitve zahtevajo prisotnost certifikatov z namenom povečanja varnosti,
- **sistem DNS** – v domeni FreeIPA, kot je bilo rečeno, nastopa več računalnikov z različnimi uporabniki oziroma storitvami, ki dostopajo do skupnih sredstev in si delijo ustrezne datoteke s konfiguracijo. Tako morajo odjemalci komunicirati med seboj, kar je zagotovljeno s pomočjo sistem DNS, saj ta povezuje določeno ime gostitelja (angl. *hostname*) z njegovim naslovom IP in obratno. Odjemalec uporabi sistem DNS, ko želi najti določenega gostitelja (angl. *host*) oziroma njegov naslov IP, ko želi dostopa do njega. Prav tako takrat, ko se določen odjemalec pridruži domeni FreeIPA, uporablja sistem DNS, saj mora ta najti strežnik FreeIPA in ostale odjemalce ter storitve v domeni FreeIPA.

Produkt FreeIPA izhaja v dveh inačicah, in sicer FreeIPAv1 in FreeIPAv2. Opazna razlika med prvo in drugo inačico je v številu storitev, ki jih produkt združuje v celoto. Prva inačica namreč ne vsebuje certifikacijskega sistema Dogtag in sistema DNS. Produkt oziroma strežnik FreeIPAv2 skupaj z vsemi storitvami, ki smo si jih ogledali v zgornjih alinejah prikazuje spodnja slika 21.



Slika 21: Strežnik FreeIPA [21].

S pomočjo tako imenovane replike IPA (angl. *IPA replica*) lahko določena domena FreeIPA vsebuje več strežnikov FreeIPA. Konfiguracija strežnika FreeIPA se lahko uporabi kot osnova za drugi strežnik FreeIPA znotraj domene FreeIPA, ta se v tem primeru imenuje replika IPA [21]. Razlika med strežnikom FreeIPA in repliko FreeIPA je le v tem, da replika FreeIPA temelji na obstoječem strežniku FreeIPA, saj uporabi njegovo konfiguracijo. Prednost uporabe več strežnikov oziroma replik FreeIPA je v tem, da je možna avtomatska porazdelitev obremenitve med njimi.

3.4 Active Directory

Active Directory je produkt podjetja Microsoft, ki je primer imeniške storitve, katera je vključena v številne strežniške operacijske sisteme Windows, kot so: Windows 2000 server, Windows server 2003 in Windows server 2008. Active Directory je zasnovan predvsem za

porazdeljena omrežja (angl. *distributed networks*) ter predstavlja temelj v porazdeljenih omrežjih, ki zajemajo več domen. Zagotavlja namreč hierarhično in varno shranjevanje podatkov o posameznih predmetih v omrežju ter omogoča centralno upravljanje s predmeti oziroma z njihovimi podatki s strani administratorja. Med predmete v omrežju sodijo številni uporabniki, računalniki, storitve in tiskalniki. Active Directory zagotavlja iskanje teh predmetov v omrežju kakor tudi delo z njimi [27]. Poleg tega lahko Active Directory opravlja avtentikacijo in avtorizacijo nad vsemi uporabniki oziroma računalniki, ki so zajeti znotraj porazdeljenega omrežja.

Active Directory nudi številne napredne lastnosti, med katerimi so naslednje [23, 27]:

- **integrirana in prilagodljiva varnost** – omogočena je uporaba protokola SSL ali protokola KERBEROS (lahko nastopita tudi v kombinaciji). Protokol SSL se lahko uporabi za vzpostavitev varnega kanala (angl. *security channel*) z namenom šifriranja, medtem ko se protokol KERBEROS lahko uporabi za sam proces avtentikacije,
- **več možnosti dostopa** – do imeniške storitve Active Directory se lahko dostopa programsko ter tako opravlja administratorske naloge. To je omogočeno na več načinov, med katerima sta: uporaba programskega vmesnika Active Directory (angl. *active directory service interface*) in programskega vmesnika LDAP (angl. *LDAP aplication program interface*),
- **bogata in razširljiva shema** – shema Active Directory definira vse možne atribute oziroma lastnosti, ki jih lahko vsebuje določen predmet. Prav tako definira posebne atribute, imenovane razredi (`objectClasses`), ki se pojavijo v vsakem predmetu ter določajo obveznost oziroma neobveznost preostalih atributov, kot je to pri imeniški storitvi LDAP. Shema je bogata, saj definira številne atribute ter je obenem razširljiva za morebitne dodatne atribute,
- **podpora protokola LDAP** – s tem se omogoči delovanje med več imeniškimi storitvami,
- **omrežna razširljivost** – produkt omogoča dodajanje oziroma brisanje ene ali več domen,
- **prilagodljivo iskanje** – z uporabo tako imenovanega globalnega kataloga (angl. *global catalog*) je možno poiskati lokacijo vseh predmetov v določeni domeni. To lastnost največkrat uporabijo aplikacije in uporabniki, ko ne poznajo razločevalnega imena (angl. *distinguished name*) predmeta oziroma ne vedo, kje v imeniški storitvi Active Directory se ta nahaja. Globalni katalog omogoča iskanje lokacije predmetov na podlagi podanih atributov, ki jih vsebuje iskani predmet.

Podatkovni model imeniške storitve Active Directory izvira iz podatkovnega modela X.500, kar pomeni, da gre za enak podatkovni model kot pri imeniški storitvi LDAP [27]. Imeniška storitev LDAP prav tako izvira iz podatkovnega modela X.500, zato veljajo med produktom OpenLDAP in Active Directory številne podobnosti. Primerjavo med njima bomo prikazali v naslednjem podpoglavju.

3.5 Primerjava med produktom OpenLDAP in Active Directory

Ko govorimo o protokolu LDAP in produktu Active Directory, gre za bistveno razliko med njima. LDAP je protokol, s pomočjo katerega lahko dostopamo do imeniških storitev, da bi pridobili podatke, medtem ko je Active Directory implementacija imeniške storitve [19, 20]. Poleg imeniške storitve Active Directory so na voljo tudi številne druge, med katerimi je OpenLDAP, ki uporablja protokol LDAP, kot tudi imeniška storitev Active Directory. Vendar pa nudi imeniška storitev podjetja Microsoft več kot produkt OpenLDAP, kar je tudi razvidno z možno uporabo avtentikacijskega protokola KERBEROS. Produkt Active Directory je bil namreč razvit z namenom, da zajema veliko več kot produkt OpenLDAP. Poleg tega produkt Active Directory v veliki meri temelji na protokolu LDAP, da bi ga podpiral in bil skladen z njim.

Čeprav produkta OpenLDAP in Active Directory implementirata imeniško storitev, veljajo med njima številne razlike, med katerimi so naslednje [37]:

- Active Directory je lastniški produkt podjetja Microsoft ter se v glavnem uporablja na strežnikih, kjer tečejo strežniški operacijski sistemi Windows. Produkt OpenLDAP lahko teče za razliko od produkta Active Directory na strežnikih, kjer ni nujno potreben strežniški operacijski sistem Windows, saj ta teče na številnih drugih operacijskih sistemih, medtem ko produkt Active Directory redko najdemo v uporabi izven operacijskega sistema Windows,
- produkt OpenLDAP omogoča zasnovo drevesne strukture (angl. *directory information tree*) na več načinov, medtem ko produkt Active Directory tega ne omogoča. Po namestitvi produkta OpenLDAP je njegova drevesna struktura prazna ter se tako pričakuje kreiranje drevesne strukture s strani administratorja. V primeru produkta Active Directory je, ob namestitvi, že pripravljena osnovna drevesna struktura ter prav tako že ponuja grafični uporabniški vmesnik z namenom takojšnega pričetka z delom. Tako je produkt Active Directory lažje pripraviti za takojšnje delo, vendar je produkt OpenLDAP bolj prilagodljiv, saj omogoča zasnovo drevesne strukture na več načinov,

- možnost avtentikacije uporabnika je pri produktu Active Directory omogočena z uporabo uporabniškega elektronskega naslova ali z uporabo uporabniškega razločevalnega imena (angl. *distinguished name*). Pri produktu OpenLDAP je namreč avtentikacija uporabnika izvedena običajno s pomočjo predloženega uporabniškega razločevalnega imena,
- produkt Active Directory podpira avtentikacijo, ki temelji na protokolu KERBEROS, medtem ko produkt OpenLDAP ne omogoča avtentikacije s pomočjo protokola KERBEROS (razen v okviru mehanizma SASL).

Uporaba produkta Active Directory je primerna v manjših in tudi v večjih omrežjih z več računalniki oziroma uporabniki. Tako se uporablja z namenom centralnega upravljanja omrežne infrastrukture, kjer nastopa več predmetov, med katerimi so lahko: računalniki, strežniki in tiskalniki. Prav tako produkt zagotavlja upravljanje z uporabniškimi profili, dovoljenji in delitvijo datotek znotraj omrežne infrastrukture, kjer lahko nastopa več domen [23]. Na drugi strani se produkt OpenLDAP uporablja kot organizacijski strežnik, kjer služi kot imenik za shranjevanje določenih podatkov. Primerni vsebovani podatki so tisti, kateri se redko spreminjajo, saj je imeniška storitev optimizirana za hitro iskanje, ne pa za spreminjanje obstoječih podatkov.

3.6 Izbira produktov

Izmed predstavljenih produktov smo izbrali tri ter jih medsebojno primerjali v okviru avtentikacije. Izbrali smo naslednje produkte:

- FreeRADIUS,
- OpenLDAP,
- FreeIPA.

V naslednjem poglavju smo prikazali primerjavo, ki smo jo opravili. Tako bo sledila avtentikacija s pomočjo produkta FreeRADIUS, OpenLDAP in FreeIPA. Na ta način smo zajeli avtentikacijo s pomočjo treh različnih protokolov (RADIUS, LDAP in KERBEROS).

4 Primerjava avtentikacije s pomočjo izbranih produktov

Da bi primerjali izvedbo avtentikacije s pomočjo izbranih produktov, smo uporabili virtualni strežnik z operacijskim sistemom CentOS 6.2, ki je bil dostopen na javnem naslovu IP. Poleg virtualnega strežnika smo uporabili tudi računalnik z operacijskim sistemom Ubuntu 11.10 TLS, ki je služil kot odjemalec. Na virtualni strežnik smo namestili vse produkte, medtem ko smo odjemalca uporabili za testne namene.

Odjemalca smo prav tako uporabljali z namenom oddaljenega dostopa do virtualnega strežnika, saj smo vso konfiguracijo na virtualnem strežniku opravljali oddaljeno s pomočjo terminala in protokola SSH. Za dostop do virtualnega strežnika smo uporabili kreiranega uporabnika na strani virtualnega strežnika ter številko vrat (angl. *port*) 2222 in ne privzeto 22 za protokol SSH, za kar je poskrbel požarni zid na strežniku, imenovan `iptables` (ki je omogočal dostop preko vrat 2222 z uporabo protokola SSH). Poleg tega je požarni zid preprečeval oddaljen dostop s pomočjo protokola SSH v primeru napačnga odjemalca (uspešno smo se lahko prijavili le z našim odjemalcem).

4.1 FreeRADIUS avtentikacija

Na oddaljeni virtualni strežnik smo namestili produkt FreeRADIUS, s pomočjo katerega smo pridobili strežnik RADIUS. Vse konfiguracijske datoteke za strežnik FreeRADIUS so se namestile v imenik `/etc/raddb`, kjer smo opravljali konfiguracijo strežnika FreeRADIUS. Strežnik FreeRADIUS smo uporabili za izvajanje avtentikacije, in sicer s pomočjo podatkovne baze MySQL in strežnika OpenLDAP, kar smo prikazali v naslednjih dveh podpoglavjih. Poleg tega smo omogočili tudi FreeRADIUS avtentikacijo, ki ne uporablja zunanjega vira.

FreeRADIUS avtentikacijo, ki ne uporablja zunanjega vira, smo omogočili tako, da smo v datoteko `/etc/raddb/users` dodali uporabnika `bob` z njegovim geslom, ki smo ga zakrili z uporabo zgoščevalne funkcije SSHA (z uporabo ukaza `slappasswd`). Dodanega uporabnika v datoteki `/etc/raddb/users` prikazuje spodnja slika 22.

```
# On no match, the user is denied access.

bob SSHA-Password := "PhWiPIkP/t7HJAMn9vzuzLMkWlfidF3H"
        Reply-Message = "Pozdravljen Bob!"
```

Slika 22: Konfiguracijska datoteka `users`.

Nato smo z uporabo programa `radtest`, ki deluje kot odjamalec za strežnik FreeRADIUS, poslali zahtevo `Access-Request` z uporabniškim imenom `bob` in njegovim geslom na strežnik FreeRADIUS. Program `radtest` smo najprej izvedli lokalno na našem virtualnem strežniku, kot prikazuje spodnja slika 23.

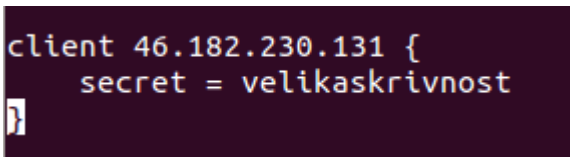
```
[matej@radius ~]$ radtest bob bobek localhost 1812 skrivnost
Sending Access-Request of id 161 to 127.0.0.1 port 1812
        User-Name = "bob"
        User-Password = "bobek"
        NAS-IP-Address = 88.200.24.242
        NAS-Port = 1812
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=161, length=38
        Reply-Message = "Pozdravljen Bob!"
[matej@radius ~]$
```

Slika 23: Lokalno testiranje avtentikacije s pomočjo programa `radtest`.

Poleg uporabniškega imena `bob` in njegovega gesla (bobek), smo programu `radtest` podali tudi naslov IP strežnika FreeRADIUS (v našem primeru `localhost`), številko vrat (angl. *port*) in skupno skrivnost (angl. *shared secret*) med odjemalcem (`radtest`) in strežnikom FreeRADIUS, kot je to razvidno z zgornje slike 23. Skupno skrivnost smo pridobili iz datoteke `/etc/raddb/clients.conf` in je v našem primeru `skrivnost`, medtem ko smo za številko vrat podali 1812, saj strežnik FreeRADIUS posluša na teh vratih v okviru procesa avtentikacije. Po strežnikovem odgovoru s sporočilom `Access-Accept`, smo se lahko prepričali o pravem posredovanem uporabniškim imenom in geslom uporabnika, saj je strežnik FreeRADIUS preveril enakost v datoteki `/etc/raddb/users` ter tako uspešno avtenticiral uporabnika.

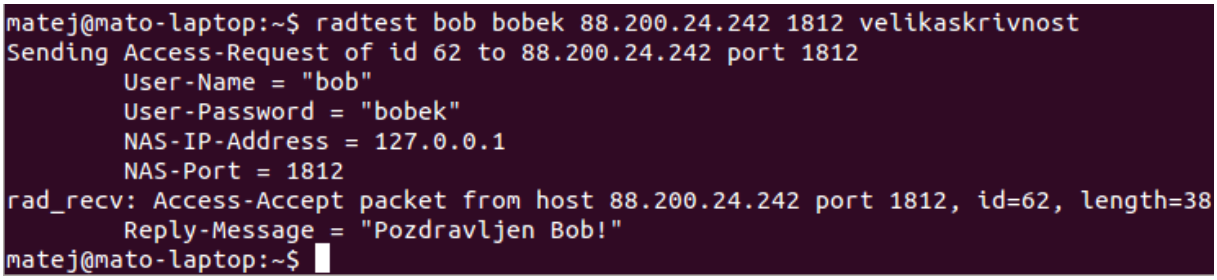
Program `radtest` smo nato izvedli še na strani našega odjemalca, pri čemer se je razlikovala vrednost skupne skrivnosti in naslov IP strežnika FreeRADIUS. Preden smo lahko izvedli program `radtest` na strani našega odjemalca, smo morali dodati zunanji naslov IP našega odjemalca (46.182.230.131), v datoteko `/etc/raddb/clients.conf` skupaj z

novo skupno skrivnostjo, ki je v našem primeru `velikaskrivnost` (prikazuje spodnja slika 24).



Slika 24: Del konfiguracijske datoteke `clients.conf`.

Z dodanim zunanjim IP naslovom našega odjemalca v datoteko `/etc/raddb/clients.conf` je strežnik FreeRADIUS omogočil dostop našemu odjemalcu, saj bi ta v nasprotnem primeru ignoriral poslano zahtevo. Izvedbo programa `radtest` na našem odjemalcu prikazuje spodnja slika 25. S slike je razvidno, da smo uporabili naslov IP našega virtualnega strežnika (`88.200.24.242`) in skupno skrivnost `velikaskrivnost`.



Slika 25: Oddaljeno testiranje avtentikacije s pomočjo programa `radtest`.

4.1.1 S pomočjo podatkovne baze MySQL

Da smo lahko omogočili FreeRADIUS avtentikacijo s pomočjo podatkovne baze MySQL, smo morali na virtualni strežnik naložiti strežnik MySQL in MySQL modul, namenjen produktu FreeRADIUS. Namestili smo paket `mysql-server`, ki predstavlja strežnik MySQL ter paket `freeradius-mysql`, s katerim smo lahko omogočili dostop strežnika FreeRADIUS do strežnika MySQL.

Na začetku smo se prijavili v strežnik MySQL z uporabo uporabniškega vmesnika za upravljanje s podatkovno bazo MySQL (angl. *MySQL monitor*). Tu smo kreirali podatkovno bazo z imenom `freeradius`, ki jo je uporabljal strežnik FreeRADIUS. Nato smo uvozili tabele produkta FreeRADIUS v kreirano podatkovno bazo `freeradius` ter tako pripravili

podatkovno bazo `freeradius` za uporabo (ustrezne kreirane tabele smo pridobili iz imenika `/usr/share/doc/freeradius-2.1.10`, kjer zadnje številke označujejo inačico strežnika FreeRADIUS). Nato smo z uporabo ustrezne poizvedbe MySQL nastavili geslo uporabniku `root`, in sicer v tabeli `user` podatkovne baze `mysql`. Rezultat je viden na spodnji sliki 26, ki prikazuje izpis tabele `user` v podatkovni bazi `mysql`.

```
mysql> select user,host,password from mysql.user;
+------+-----------+-------------------------------------------+
| user | host      | password                                  |
+------+-----------+-------------------------------------------+
| root | localhost | *2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19 |
| root | radius    | *2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19 |
| root | 127.0.0.1 | *2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19 |
+------+-----------+-------------------------------------------+
3 rows in set (0.00 sec)

mysql>
```

Slika 26: Izpis tabele `user` v podatkovni bazi `mysql`.

Uporabniku `root` smo z namenom določili geslo, saj smo z njim dostopali do uporabniškega vmesnika za upravljanje s podatkovno bazo MySQL. Pred tem je bila prijava vanj uspešna le s posredovanim imenom uporabnika (`root`). Uporabnika `root` smo prav tako uporabili za dostop strežnika FreeRADIUS do strežnika MySQL oziroma podatkovne baze `freeradius` v njem. V datoteko `/etc /raddb/sql.conf`, ki predstavlja konfiguracijsko datoteko za modul MySQL, smo dodali uporabnika `root` in njegovo geslo. Poleg tega smo morali nastaviti ime podatkovne baze, do katere bo strežnik FreeRADIUS dostopal. Prepričati smo se morali tudi o pravem podanem naslovu IP strežnika MySQL in izbiri podatkovne baze. Vsebino datoteke `/etc/raddb/sql.conf` prikazuje spodnja slika 27.

```
sql {
        #
        #  Set the database to one of:
        #
        #       mysql, mssql, oracle, postgresql
        #
        database = "mysql"

        #
        #  Which FreeRADIUS driver to use.
        #
        driver = "rlm_sql_${database}"

        # Connection info:
        server = "localhost"
        #port = 3306
        login = "root"
        password = "password"

        # Database table configuration for everything except Oracle
        radius_db = "freeradius"
```

Slika 27: Del konfiguracijske datoteke za modul MySQL.

Kot je razvidno z zgornje slike 27, smo za naslov strežnika uporabili vrednost `localhost`, saj se strežnik MySQL in FreeRADIUS nahajata na istem virtualnem strežniku. Izbiro podatkovne baze vsebuje atribut `database`, ki je v našem primeru seveda `mysql`. Atribut `login` vsebuje ime uporabnika za dostop do strežnika MySQL (v našem primeru `root`), medtem ko je njegovo geslo podano z atributom `password`. Preostali atribut `radius_db` pa označuje ime podatkovne baze, do katere želi strežnik FreeRADIUS dostopati, ki je v našem primeru seveda `freeradius`. S tem smo omogočili strežniku FreeRADIUS dostop do strežnika MySQL oziroma do podatkovne baze `freeradius`.

Da je lahko strežnik FreeRADIUS v okviru procesa avtentikacije dostopal do strežnika MySQL oziroma podatkovne baze `freeradius`, smo morali v datoteki `/etc/raddb/sites-enabled/default` odkomentirati niz `mysql`, in sicer znotraj tako dela `authorize`. Konfiguracijska datoteka `/etc/raddb/sites-enabled/default` se namreč nanaša na virtualni strežnik, imenovan `default`, ki je del produkta FreeRADIUS. Ta virtualni strežnik obdeluje običajne avtentikacijske zahteve, medtem ko virtualni strežnik `inner-tunel` obdeluje določene EAP zahteve. Niz `mysql` v datoteki `/etc/raddb/sites-enabled/default` prikazuje spodnja slika 28.

```
        #
        #  Look in an SQL database.  The schema of the database
        #  is meant to mirror the "users" file.
        #
        #  See "Authorization Queries" in sql.conf
        sql
```

Slika 28: Del konfiguracijske datoteke virtualnega strežnika `default`.

Del datoteke `/etc/raddb/sites-enabled/default`, imenovan `authorize`, podaja, kam vse strežnik FreeRADIUS dostopa v okviru procesa avtentikacije. Tako je v našem primeru strežnik FreeRADIUS dostopal do strežnika MySQL.

Za izvedbo testiranja FreeRADIUS avtentikacije s pomočjo podatkovne baze MySQL smo v podatkovno bazo `freradius` znotraj strežnika MySQL dodali dva uporabnika z določenim geslom. Ta uporabnika smo dodali v tabelo `radcheck`, ki je ena izmed uvoženih tabel. To tabelo privzeto uporablja strežnik FreeRADIUS v okviru procesa avtentikacije, saj ta v njej preveri vsebovanost oziroma pravilnost uporabniškega imena in gesla. Dodana uporabnika v tabeli `radcheck` prikazuje spodnja slika 29.

```
mysql> select * from freeradius.radcheck;
+----+----------+---------------+----+------------------------------------------+
| id | username | attribute     | op | value                                    |
+----+----------+---------------+----+------------------------------------------+
|  6 | test     | SHA1-Password | := | 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8 |
|  7 | upo      | SHA1-Password | := | e8cc564a5e9320d6c22647c5e6dab55005bf1e68 |
+----+----------+---------------+----+------------------------------------------+
2 rows in set (0.00 sec)

mysql>
```

Slika 29: Prikaz vsebine tabele `radcheck`.

Z zgornje slike 29 lahko razberemo, da je geslo uporabnika `test` in `upo` šifrirano, saj smo to opravili z namenom povečanja varnosti. Pri tem smo uporabili zgoščevalno funkcijo SHA-1, kar je razvidno v stolpcu `attribute` na zgornji slike 29.

Testiranje smo nato naprej opravili lokalno na našem virtualnem strežniku, s pomočjo programa `radtest`, kot to prikazuje zgornja slika 23. Vendar smo tu, namesto uporabniškega imena `bob` in njegovega gesla (bobek), uporabili uporabniško ime `test` oziroma `upo` ter njegovo geslo. Tako smo, v okviru sporočila `Access-Request`, poslali

na strežnik FreeRADIUS uporabniško ime `test` oziroma `upo` in njegovo geslo. Strežnik FreeRADIUS je nato preveril vsebovanost oziroma pravilnost uporabniškega imena `test` oziroma `upo` in njegovega gesla v tabeli `radcheck` znotraj podatkovne baze `freeradius`. Strežnik FreeRADIUS je nato seveda avtenticiral uporabnika ter poslal sporočilo `Access-Accept`, kar ponazarja uspešno avtentikacijo.

Nato smo program `radtest` izvedli še na našem odjemalcu, kot prikazuje zgornja slika 25. Vendar smo tu prav tako namesto uporabniškega imena `bob` in njegovega gesla (bobek), uporabili uporabniško ime `test` oziroma `upo` in njegovo geslo. V tem primeru smo prav tako pridobili sporočilo `Access-Accept`.

4.1.2 S pomočjo strežnika OpenLDAP

Najprej smo na virtualni strežnik namestili potrebne pakete, in sicer `openldap-servers`, `freeradius-ldap` in paket `openldap-clients`. Z nameščenim paketom `openldap-servers` smo pridobili strežnik OpenLDAP, imenovan `slapd`, ter orodja za zagon in konfiguracijo strežnika OpenLDAP. Za dostop strežnika FreeRADIUS do strežnika OpenLDAP smo potrebovali ustrezen LDAP modul, ki ga vključuje paket `freeradius-ldap`. S preostalim paketom `open-ldap clients` smo pridobili ustrezna orodja za izvajanje v terminalu, s pomočjo katerih je omogočeno upravljanje s strežnikom OpenLDAP.

Strežnik OpenLDAP je bilo treba najprej ustrezno skonfigurirati, v kar je vključeno tudi kreiranje drevesne strukture (angl. *directory information tree*), saj je bila ta po namestitvi prazna. Globalna konfiguracija za strežnik OpenLDAP se nahaja v datoteki `/etc/openldap/slapd.d/cn=config.ldif`. Datoteka vsebuje privzete nastavitve, ki so nam zadoščale, zato jih tako nismo spreminjali. Poleg datoteke `cn=config.ldif` smo posegali tudi v datoteko `olcDatabase={2}bdb.ldif`, ki se nahaja v imeniku `/etc/openldap/slapd.d/cn=config/`. Le-ta omogoča konfiguracijo podatkovne baze, imenovane Berkeley (angl. *Berkeley database*), ki jo strežnik OpenLDAP uporablja pod privzetimi nastavitvami za shranjevanje predmetov. V tej datoteki so omembe vredne naslednje nastavitve:

- uporaba podatkovne baze Berkeley le za bralni dostop, kar omogočimo s pomočjo atributa `olcReadOnly` ter njegovo prednostjo `true`. Uporabljena vrednost je tu seveda `false`, saj podatkovne baze Berkeley nismo potrebovali le za branje,

- določitev uporabnika, ki ima administratorske pravice ter lahko izvaja vse operacije nad podatkovno bazo Berkeley. Določitev omogoča atribut `olcRootDN`, ki prejme kot vrednost uporabnika v obliki razločevalnega imena (angl. *distinguished name*). Vrednost je v našem primeru `cn=Manager,dn=my-domain,dn=com`,
- določitev gesla za uporabnika, ki je podano kot vrednost atributa `olcRootDN`. Geslo smo določili s pomočjo atributa `olcRootPW` ter mu kot vrednost podali izbrano šifrirano geslo. Geslo smo šifrirali z uporabo ukaza `slappasswd`, ki prejme geslo v čistopisu (angl. *cleartext*) ter ga šifrira z določeno zgoščevalno funkcijo (v našem primeru je geslo šifrirano z uporabo zgoščevalne funkcije SSHA),
- določitev imena podatkovne baze Berkeley v obliki domene, kar je omočeno z uporabo atributa `olcSuffix`. Vrednost atributa je v našem primeru `dc=my-domain,dc=com`,
- določitev imenika, kjer se dejansko nahaja podatkovna baza Berkeley. To je omogočeno z atributom `olcDbDirectory`, ki ima v našem primeru vrednost `/var/lib/ldap`. V ta imenik smo morali prenesti tudi datoteko, imenovano `DB_CONFIG`, ki je potrebna za boljšo zmogljivost, kakor tudi za pravilno delovanje podatkovne baze Berkeley in strežnika OpenLDAP (`slapd`).

Vse te nastavitve ter tudi nekatere druge prikazuje spodnja slika 30. Ta namreč prikazuje del konfiguracijske datoteke `olcDatabase={2}bdb.ldif`.

```
dn: olcDatabase={2}bdb
objectClass: olcDatabaseConfig
objectClass: olcBdbConfig
olcDatabase: {2}bdb
olcSuffix: dc=my-domain,dc=com
olcAddContentAcl: FALSE
olcLastMod: TRUE
olcMaxDerefDepth: 15
olcReadOnly: FALSE
olcRootDN: cn=Manager,dc=my-domain,dc=com
olcRootPW: {SSHA}UBdlP+BJOIVLO4WIkPisMhTFy4oe+nQp
olcSyncUseSubentry: FALSE
olcMonitoring: TRUE
olcDbDirectory: /var/lib/ldap
```

Slika 30: Del konfiguracijske datoteke podatkovne baze Berkeley.

Kreirali smo drevesno strukturo strežnika OpenLDAP, saj smo potrebovali podatke za samo testiranje avtentikacije. Tako smo pričeli z vnosom prvega predmeta z razločevalnim imenom `dc=my-domain,dc=com`, ki predstavlja korenski predmet, iz katerega izvirajo vsi

preostali predmeti. Ta korenski predmet predstavlja ime podatkovne baze Berkeley, na katerega smo se sklicevali v primeru iskanja podatkov na strežniku OpenLDAP. Po vnosu korenskega predmeta smo vnesli dva predmeta z razločevalnim imenom `ou=Marketing,dc=my-domain,dc=com` in `ou=Sales,dc=my-domain,dc=com`, ki tako izhajata iz korenskega predmeta. Nato smo še dodali dva predmeta, kjer prvi izhaja iz predmeta `ou=Marketing,dc=my-domain,dc=com` ter drugi iz predmeta `ou=Sales,dc=my-domain,dc=com`. Ta dva predmeta predstavljata uporabnika, ki smo ju potrebovali za testne namene. Prvi uporabnik je določen z razločevalnim imenom `cn=Matek Mo,ou=Marketing,dc=my-domain,dc=com`, medtem ko je drugi določen z `cn=John Doe,ou=Sales,dc=my-domain,dc=com`. Vse predmete in njihove atribute smo zapisali v datoteko LDIF ter nato s pomočjo ukaza `ldapadd`, ki je kot argument prejel to datoteko, dodali predmete v podatkovni bazo Berkeley. Na ta način smo pridobili manjšo drevesno strukturo. Vsi predmeti skupaj z njegovimi atributi so seveda morali ustrezati pravilom sheme LDAP, ki definira atribute in razrede (`objectClasses`) ter se nahaja v imeniku `/etc/openldap/schema/`. Spodnja slika 31 prikazuje vneseni predmet z razločevalnim imenom `cn=John Doe,ou=Sales,dc=my-domain,dc=com` skupaj z njegovimi atributi, kar smo uporabili za testne namene.

```
dn: cn=John Doe,ou=Sales,dc=my-domain,dc=com
objectClass: person
objectClass: inetOrgPerson
cn: John
sn: Doe
uid: jdoe
userPassword: {SSHA}QcIeBLlu7k65rVFQxrRC9oV/bpk/7qgq
```

Slika 31: Datoteka LDIF z vsebovanim predmetom, ki predstavlja uporabnika.

Kot je razvidno z zgornje slike 31, smo geslo uporabnika šifrirali z uporabo zgoščevalne funkcije SSHA, kar smo opravili z ukazom `slappasswd`. Poleg tega uporabnik vsebuje atribut `uid` z vrednostjo `jdoe`, ki predstavlja enolični identifikator predmeta oziroma uporabnika. Ta enolični identifikator smo potrebovali pri testiranju FreeRADIUS avtentikacije s pomočjo strežnika LDAP, kjer smo se s pomočjo tega sklicevali na uporabnika. Drugi uporabnik, ki smo dodali v podatkovno bazo Berkeley, vsebuje iste atribute kot uporabnik na zgornji sliki 31, le da ta izhaja iz predmeta z razločevalnim imenom `ou=Marketing,dc=my-domain,dc=com`.

Ko smo pripravili strežnik OpenLDAP, smo morali omogočiti dostop strežnika FreeRADIUS do strežnika OpenLDAP, in sicer smo to storili z uporabo modula LDAP. V konfiguracijski

datoteki, za modul LDAP (`/etc/raddb/modules/ldap`), smo nastavili: naslov IP strežnika OpenLDAP, uporabnika za dostop do strežnika OpenLDAP ter njegovo geslo, ime podatkovne baze Berkeley in številko vrat (angl. *port*). Te nastavitve prikazuje spodnja slika 32.

```
ldap {
        #
        #  Note that this needs to match the name in the LDAP
        #  server certificate, if you're using ldaps.
        server = "localhost"
        identity = "cn=Manager,dc=my-domain,dc=com"
        password = geslo
        basedn = "dc=my-domain,dc=com"
        port = 389
```

Slika 32: Del konfiguracijske datoteke za modul LDAP.

Vrednost atributa `server` smo nastavili na `localhost`, saj se strežnik OpenLDAP nahaja na istem virtualnem strežniku kot strežnik FreeRADIUS, kar vidimo na zgornji sliki 32. Prav tako je razvidno, da smo za dostop strežnika OpenLDAP nastavili razločevalno ime (angl. *distinguished name*) uporabnika, ki ima administratorske pravice ter je določen v konfiguracijski datoteki podatkovne baze Berkeley. Uporabili smo tudi njegovo nastavljeno geslo v tej datoteki, kjer je podan kot vrednost atributa `password`. Tako smo s temi nastavitvami omogočili strežniku FreeRADIUS dostop do strežnika OpenLDAP.

Preden smo pričeli s testiranjem FreeRADIUS avtentikacije s pomočjo strežnika OpenLDAP, smo morali odkomentirati niz `ldap` v konfiguracijski datoteki `/etc/raddb/sites-enabled/default`, znotraj dela `authorize`. S tem smo strežniku FreeRADIUS določili, naj dostopa do strežnika OpenLDAP v okviru procesa avtentikacije. Niz `ldap` v tej datoteki prikazuje spodnja slika 33.

```
        #
        #  The ldap module will set Auth-Type to LDAP if it has not
        #  already been set
        ldap
```

Slika 33: Del konfiguracijske datoteke `default`.

Po tej nastavitvi smo lahko pričeli s testiranjem FreeRADIUS avtentikacije s pomočjo strežnika OpenLDAP. Testiranje smo izvedli z uporabo programa `radtest` ter ga najprej

izvedli na strani našega virtualnega strežnika, kot to prikazuje zgornja slika 23. Namesto uporabniškega imena `bob` in njegovega gesla (bobek) smo uporabili uporabniško ime `jdoe` in njegovo geslo. Tako smo z odjemalcem `radtest` poslali strežniku FreeRADIUS sporočilo `Access-Request` z uporabniškim imenom `jdoe` in njegovim geslom. Strežnik FreeRADIUS je nato dostopal do strežnika OpenLDAP z uporabo razločevalnega imena uporabnika, ki ima administratorske pravice (`cn=Manager,dc=my-domain,dc=com`) ter njegovega gesla. Nato je strežnik OpenLDAP na zahtevo strežnika FreeRADIUS preveril vsebovanost uporabnika `jdoe` in njegovega gesla v podatkovni bazi Berkeley. Tako je FreeRADIUS avtentikacija s pomočjo strežnika OpenLDAP uspela, saj se je uporabnik `jdoe` in njegovo geslo nahajalo v podatkovni bazi Berkeley. Prav tako je uspešno avtentikacijo potrdilo prejeto sporočilo `Access-Accept` s strani strežnika FreeRADIUS.

Program `radtest` smo nato izvedli še na strani našega odjemalca, kot to prikazuje zgornja slika 25. Tu smo prav tako namesto uporabnika `bob` in njegovo geslo (bobek) uporabili uporabniško ime `jdoe` in pripadajoče geslo ter pridobili sporočilo `Access-Accept` od strežnika FreeRADIUS, s katerim smo se prepričali, da je avtentikacija uspela.

4.2 FreeRADIUS avtentikacija Linux uporabnikov

FreeRADIUS avtentikacijo Linux uporabnikov smo izvedli s pomočjo mehanizma PAM. Ta predstavlja centralni avtentikacijski mehanizem, ki ga uporabljajo številne aplikacije v operacijskem sistemu Linux ter omogoča proces avtentikacije tem aplikacijam s pomočjo PAM modulov. Prav tako operacijski sistem Linux uporablja mehanizem PAM v okviru procesa avtentikacije, zato ni potrebna prilagoditev aplikacij v primeru zahteve po novem avtentikacijskem mehanizmu, kot je avtentikacija s protokolom RADIUS, LDAP ali KERBEROS, saj je v tem primeru treba le zapisati primeren PAM modul v konfiguracijsko datoteko, ki je namenjena tej aplikaciji [30]. Vse aplikacije, ki se zanašajo na mehanizem PAM oziroma ga uporabljajo, vsebujejo svojo lastno konfiguracijsko datoteko v imeniku `/etc/pam.d`. Pred prihodom mehanizma PAM je bila potrebna prilagoditev aplikacije z namenom, da se vpelje nov zahtevan avtentikacijski mehanizem, kar pa pomeni daljši proces ter večjo verjetnost pojava napak. Vsaka aplikacija je imela svoj lasten mehanizem za zagotavljanje procesa avtentikacije, kar se je s prihodom mehanizma PAM spremenilo. Uporaba mehanizma PAM prinaša naslednje prednosti [32]:

- uporaba skupnega avtentikacijskega mehanizma (PAM) s strani več različnih aplikacij,

- sistemski administratorji lahko izkoristijo prilagodljivost pri nastavitvah avtentikacijskih mehanizmov za določeno aplikacijo,
- razvijalcem aplikacij ni treba kreirati lastnega avtentikacijskega mehanizma za določeno aplikacijo.

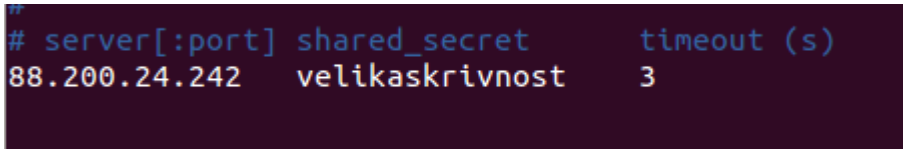
V našem primeru smo skonfigurirali FreeRADIUS avtentikacijo Linux uporabnikov za uporabo storitve SSH in SUDO. V primeru storitve SSH smo se iz virtualnega strežnika prijavili na našega odjemalca s pomočjo protokola SSH. Oddaljeni odjemalec je v tem trenutku dostopal do strežnika FreeRADIUS ter tam preveril pravilnost podanega uporabniškega imena in gesla. Tako je bila uspešnost naše oddaljene prijave odvisna od odgovora strežnika FreeRADIUS. V primeru pravilnega uporabniškega imena in gesla smo se seveda uspešno prijavili.

V primeru storitve SUDO smo testiranje izvedli s prijavo v našega odjemalca. Tu je seveda pravilnost uporabniškega imena in gesla zopet preveril strežnik FreeRADIUS, saj je naš odjemalec dostopal do njega ter pridobil ustrezen odgovor. Tako smo se tudi v tem primeru, ob pravilno posredovanem uporabniškem imenu in geslu uspešno prijavili. Konfiguracijo in podrobnosti smo prikazali v naslednjem podpoglavju, in sicer za uporabo storitve SSH in tudi za SUDO.

4.2.1 Pam_Radius modul in varnost

Za izvedbo FreeRADIUS avtentikcije Linux uporabnikov smo naprej morali namestiti paket `libpam-radius-auth`, ki predstavlja modul `Pam_Radius`, na našega odjemalca. S pomočjo tega PAM modula je namreč lahko naš odjemalec dostopal do strežnika FreeRADIUS v okviru procesa avtentikacije. Prav tako smo na našem odjemalcu kreirali novega uporabnika, imenovan `bob`, saj smo ga uporabili za testne namene v primeru oddaljene prijave s pomočjo protokola SSH. Poleg tega smo še namestili paket `openssh-server`, ki predstavlja storitev SSHD, ki omogoča oddaljeno prijavo s pomočjo protokola SSH.

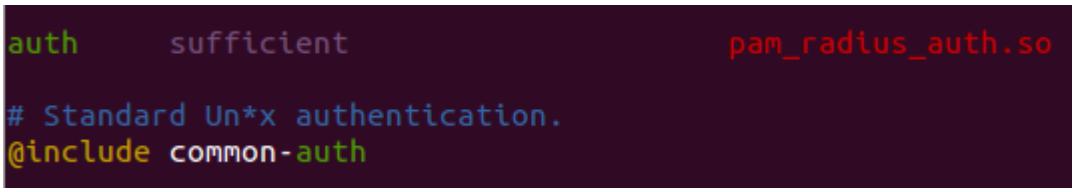
Našemu odjemalcu smo najprej določili podatke, s katerimi bo lahko dostopal do strežnika FreeRADIUS. Podatke za dostop smo mu določili v konfiguracijski datoteki `/etc/pam_radius_auth.conf`. Ta datoteka namreč predstavlja konfiguracijsko datoteko za modul `Pam_Radius`. Podatke, ki smo jih zapisali v to konfiguracijsko datoteko, prikazuje spodnja slika 34.



Slika 34: Konfiguracijska datoteka za modul `Pam_Radius`.

Kot vidimo na zgornji sliki 34, smo nastavili naslov IP našega virtualnega strežnika, kjer se nahaja strežnik FreeRADIUS. Poleg tega smo morali navesti skupno skrivnost (angl. *shared secret*) med našim odjemalcem in virtualnim strežnikom, ki smo jo že definirali v konfiguracijski datoteki `/etc/raddb/clients.conf`, kar je razvidno iz zgornje slike 24. Poleg tega smo še nastavili število sekund, ki določajo, koliko časa naj modul `Pam_Radius` čaka na odgovor strežnika FreeRADIUS (v našem primeru tri sekunde). Če modul `Pam_Radius` ne pridobi odgovora v določenem času, bo ta sprejel, da se strežnik FreeRADIUS ne odziva.

Nato smo morali še na našem odjemalcu nastaviti PAM modul, imenovan `pam_radius_auth.so` v konfiguracijski datoteki `/etc/pam.d/sshd`. Ta datoteka je namenjena storitvi SSHD, kjer smo s pomočjo modula `pam_radius_auth.so` dodali nov avtentikacijski mehanizem za to storitev. Tako bo storitev SSHD uporabljala avtentikacijo s pomočjo protokola RADIUS ter dostopala do strežnika FreeRADIUS z uporabo podatkov v konfiguracijski datoteki `/etc/pam_radius_auth.conf`. Zapis modula `pam_radius_auth.so` v konfiguracijski datoteki `/etc/pam.d/sshd` prikazuje spodnja slika 35.



Slika 35: Del konfiguracijske datoteke PAM za storitev SSHD.

Kot je razvidno z zgornje slike 35, se v isti vrstici, kjer je modul `pam_radius_auth.so`, nahaja tudi niz `auth` in `sufficient`. Prvi niz, imenovan `auth`, predstavlja tip modula, medtem ko niz `sufficient` predstavlja nadzorno zastavico (angl. *control flag*), ki označuje vedenje modula PAM (`pam_radius_auth.so`). S tipom modula `auth` smo določili, naj modul avtenticira uporabnika ter tako od njega zahteva in preveri geslo. Nadzorna zastavica `sufficient` pa ima naslednjo lastnost: če [30, 32] se modul ne izvede uspešno, se to

enostavno ignorira. Vendar ko se modul z nadzorno zastavico `sufficient` izvede uspešno, ni potrebna izvedba nobenega modula več v tej konfiguracijski datoteki PAM. Tako je uporabnik po uspešni izvedbi takoj avtenticiran. Nato nastopi pogoj, in sicer da se vsi moduli z nadzorno zastavico `required`, ki se nahajajo pred modulom z nadzorno zastavico `sufficient`, izvedejo uspešno. V našem primeru z zgornje slike 35 to pomeni, da se bo ob neuspešni izvedbi modula `pam_radius_auth.so` izvedla še datoteka `common-auth`, ki je vključena v to PAM konfiguracijsko datoteko. Če pa se bo modul `pam_radius_auth.so` izvedel uspešno, se vključena datoteka ne bo izvedla in uporabnik bo avtenticiran za uporabo storitve SSHD. Datoteka `common-auth` predstavlja konfiguracijsko datoteko, ki vsebuje module tipa `auth`, ter je skupna za vse storitve, ki uporablja mehanizem PAM za proces avtentikacije (spremembe v tej datoteki vplivajo na vse storitve, ki uporabljajo mehanizem PAM za proces avtentikacije, saj je vključena v vseh konfiguracijskih datotekah teh storitev). Na ta način je omogočeno na enem mestu spreminjati avtentikacijske mehanizme za vse storitve. Poleg tega datoteka `common-auth` predstavlja standardno avtentikacijo na našem odjemalcu, kar je tudi razvidno z zgornje slike 35.

Po tem smo lahko pričeli s testiranjam FreeRADIUS avtentikacijo Linux uporabnikov za storitev SSH. Uporabili smo uporabnika `bob`, ki smo ga že imeli v datoteki `/etc/raddb/users` na strani strežnika FreeRADIUS, kot prikazuje zgornja slika 22. Tako je strežnik FreeRADIUS pridobil potrebne podatke o uporabniku ter je lahko izvedel avtentikacijo, testiranje pa smo izvedli na strani virtualnega strežnika, kot prikazuje spodnja slika 36. Preden smo lahko izvedli testiranje, smo morali še namestiti paket `openssh-clients`, ki predstavlja odjemalca za storitev SSHD.

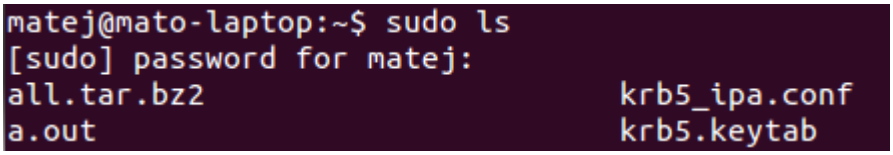
```
[matej@radius ~]$ ssh bob@46.182.230.131
bob@46.182.230.131's password:
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-20-generic-pae i686)
```

Slika 36: Oddaljena prijava na našega odjemalca s pomočjo protokola SSH.

Kot vidimo na zgornji sliki 36, smo ukazu `ssh` podali dva parametra. Prvi predstavlja uporabnika, s katerim se želimo oddaljeno prijaviti (`bob`), medtem ko drugi predstavlja zunanji naslov IP našega odjemalca. Ker smo podali zunanji naslov IP našega odjemalca, smo morali na usmerjavalniku (angl. *router*), kamor je priključen naš odjemalec, nastaviti posredovanje zahteve, ki pride na številko vrat (angl. *port*) 22, na notranji oziroma lokalni naslov IP našega odjemalca (brez tega posredovanja oddaljena prijava na našega odjemalca ni bila možna). Po izvedbi ukaza smo vnesli geslo za uporabnika `bob`, ki je shranjeno v

konfiguracijski datoteki `/etc/raddb/users`. Nato je naš odjemalec s pomočjo modula `pam_radius_auth.so`, dostopal do strežnika FreeRADIUS s poslano zahtevo `Access-Request` ter mu posredoval uporabniško ime `bob` in njegovo geslo. Strežnik FreeRADIUS je preveril pravilnost uporabniškega imena in gesla ter odgovoril s poročilom `Access-Accept`. Tako se je modul `pam_radius_auth.so` v konfiguracijski datoteki `/etc/pam.d/sshd` izvedel uspešno. Posledično smo se seveda uspešno oddaljeno prijavili.

Pri izvedbi testiranja FreeRADIUS avtentikacije Linux uporabnikov za storitev SUDO smo najprej dodali modul `pam_radius_auth.so` v PAM konfiguracijsko datoteko `/etc/pam.d/sudo` na našemu odjemalcu, ki je namenjena tej storitvi. Modul smo dodali prav tako nad zapisom, ki vključuje datoteko `common-auth`, kot je prikazano na zgornji sliki 35. Poleg tega smo na strani strežnika FreeRADIUS dodali še enega uporabnika (`matej`) v datoteko `/etc/raddb/users`, ki smo ga uporabili za testne namene, ter prav tako šifrirali njegovo geslo z uporabo zgoščevalne funkcije SSHA, kot smo to storili v primeru uporabnika `bob` z zgornje slike 22. Nato smo lahko pričeli s testiranjem avtentikacije, in sicer na naslednji način: na našem odjemalcu smo v novem terminalu izvedli ukaz `sudo ls`. Nato smo podali uporabniško geslo za dodanega uporabnika `matej`, ki smo ga dodali v datoteko `/etc/raddb/users` na strani strežnika FreeRADIUS. Uporabnika z uporabniškim imenom `matej` smo dodali namenoma, saj smo bili na našem odjemalcu prijavljeni s tem uporabniškim imenom ter smo tako lahko izvedli testiranje avtentikacije v terminalu (potreben je bil predhodno lokalno kreiran uporabnik na našem odjemalcu). Tako je naš odjemalec s pomočjo modula `pam_radius_auth.so` dostopal do strežnika FreeRADIUS s sporočilom `Access-Request` ter mu posredoval uporabniško ime `matej` in njegovo geslo, ki smo ga podali. Strežnik FreeRADIUS je preveril pravilnost uporabniškega imena in gesla, ter vrnil sporočilo `AccessAccept`. Tako se je ukaz `sudo ls` v terminalu izvršil uspešno ter prikazal vsebino trenutnega imenika, saj se je modul `pam_radius_auth.so` izvedel uspešno. Prikaz izvedbe ukaza `sudo ls` in izpis trenutnega imenika prikazuje spodnja slika 37.



Slika 37: Testiranje avtentikacije s pomočjo ukaza `sudo ls`.

Komunikacija med našim odjemalcem in virtualnim strežnikom v nobenem izmed obeh primerov uporabe modula `Pam_Radius` ni varna. Modul `Pam_Radius` uporablja avtentikacijski protokol PAP oziroma pošilja sporočilo PAP, ko dostopa do strežnika FreeRADIUS. To pomeni, da naš odjemalec pošilja uporabniško geslo v čistopisu (angl. *cleartext*), ko dostopa do strežnika FreeRADIUS, kar seveda ni varno. Vendar modul `Pam_Radius` ne omogoča uporabe drugega avtentikacijskega protokola, kot je na primer EAP ter tudi ne omogoča pošiljanje EAP sporočil. Če želimo omogočiti varno povezavo pri uporabi `Pam_Radius` modula, je treba vzpostaviti varno omrežje med odjemalcem in strežnikom FreeRADIUS. Tu lahko nastopi tuneliranje povezave med odjemalcem in strežnikom FreeRADIUS z uporabo virtualnega privatnega omrežja (angl. *virtual private network*).

4.3 Beleženje avtentikacij na strežniku FreeRADIUS

V okviru beleženja avtentikacij na strežniku FreeRADIUS smo nastavili beleženje vseh avtentikacij, ki jih opravi strežnik FreeRADIUS. Na ta način smo vedeli, kdo se je avtenticiral oziroma prijavil v sistem ter kdaj. V naslednjem podpoglavju smo prikazali potrebno konfiguracijo in testiranje beleženja avtentikacij na strežniku FreeRADIUS.

4.3.1 S pomočjo podatkovne baze MySQL

Beleženja avtentikacij na strežniku FreeRADIUS smo izvedli s pomočjo tabele `radpostauth`, katero smo pridobili že pri uvozu tabel v podatkovno bazo `freeradius`. Strežnik FreeRADIUS je tako ob vsaki uspešni avtentikaciji dostopal do strežnika MySQL ter zapisal uporabniško ime avtenticiranega uporabnika in čas avtentikacije v tabelo `radpostauth` znotraj podatkovne baze `freeradius`. Za dostop strežnika FreeRADIUS do strežnika MySQL smo uporabili, tako kot v primeru FreeRADIUS avtentikacije s pomočjo podatkovne baze MySQL, modul MySQL. Prav tako smo uporabili iste podatke za dostop strežnika FreeRADIUS do strežnika MySQL v konfiguracijski datoteki `/etc/raddb/sql.conf`, kot prikazuje zgornja slika 27.

Poleg nastavitve modula MySQL smo morali v konfiguracijski datoteki `/etc/raddb/sites-enabled/default` odkomentirati niz `sql`, in sicer znotraj dela `post-auth`, kjer se izvedejo določeni koraki takoj po uspešni avtentikaciji. Tako smo omogočili takojšen zapis podatkov po uspešni avtentikaciji v tabelo `radpostauth` znotraj

podatkovne baze `freeradius`. Niz `sql` znotraj dela `post-auth` prikazuje spodnja slika 38.

```
#  Post-Authentication
#  Once we KNOW that the user has been authenticated, there are
#  additional steps we can take.
post-auth {
        #  Get an address from the IP Pool.
#       main_pool

        #
        #  If you want to have a log of authentication replies,
        #  un-comment the following line, and the 'detail reply_log'
        #  section, above.
#       reply_log

        #
        #  After authenticating the user, do another SQL query.
        #
        #  See "Authentication Logging Queries" in sql.conf
        sql
```

Slika 38: Del konfiguracijske datoteke za virtualni strežnik `default`.

Testiranje beleženja avtentikacij smo izvedli s pomočjo programa `radtest` na strani našega odjemalca, kot to prikazuje zgornja slika 25. Poleg uporabniškega imena `bob` in njegovega gesla smo uporabili tudi uporabnika `upo` in `test` (s pripadajočim geslom), ki smo jih dodali v okviru FreeRADIUS avtentikacije s pomočjo podatkovne baze MySQL. Tako smo s pomočjo programa `radtest` v okviru zahteve `Access-Request` posredovali izbrano uporabniško ime in pripadajoče geslo do strežnika FreeRADIUS, ki je nato preveril vsebovanost podanega uporabnika in njegovega gesla v datoteki `/etc/raddb/users` (v primeru uporabnika `bob` in `test` je preveril vsebovanost uporabnika v tabeli `radcheck` znotraj podatkovne baze `freeradius`). Nato je strežnik FreeRADIUS avtenticiral uporabnika, ga zapisal in zabeležil čas avtentikacije v tabelo `radpostauth` ter nam vrnil sporočilo `Access-Accept`. Tako je bila avtentikacija uspešna, pri čemer smo še pridobili zapis uporabnika in čas njegove avtentikacije v tabeli `radpostauth` znotraj podatkovne baze `freeradius`. Zapis uporabniškega imena in njegova gesla prikazuje spodnja slika 39.

```
mysql> select * from freeradius.radpostauth;
+----+----------+----------+---------------+---------------------+
| id | username | pass     | reply         | authdate            |
+----+----------+----------+---------------+---------------------+
|  1 | upo      | geslo    | Access-Accept | 2012-05-08 14:06:54 |
|  2 | test     | password | Access-Accept | 2012-05-08 14:24:52 |
|  4 | bob      | bobek    | Access-Accept | 2012-08-17 11:12:38 |
+----+----------+----------+---------------+---------------------+
3 rows in set (0.00 sec)
```

Slika 39: Izpis tabele `radpostauth` znotraj podatkovne baze `freeradius`.

Kot vidimo z zgornje slike 39, je strežnik FreeRADIUS za vsakega avtenticiranega uporabnika poleg uporabniškega imena in časa avtentikacije zapisal še uporabniško geslo in njegov odgovor, ki se nahaja v stolpcu `reply`, ter je v našem primeru seveda `Access-Accept`. Na ta način smo lahko uspešno razbrali čas avtentikacije določenega uporabnika.

4.4 OpenLDAP avtentikacija

V okviru OpenLDAP avtentikacije smo omogočili avtentikacijo s pomočjo protokola LDAP. To smo omogočili s pomočjo mehanizma PAM oziroma z uporabo njegovega modula `Pam_Ldap`. Tako smo na našem odjemalcu namestili paket `libpam-ldap`, ki predstavlja modul `Pam_Ldap`, ki smo ga uporabili za neposreden dostop do strežnika OpenLDAP. V tem primeru je strežnik OpenLDAP služil kot avtentikacijski strežnik, saj smo do njega dostopali v okviru procesa avtentikacije. V primeru avtentikacije uporabnika smo z uporabo modula `Pam_Ldap` dostopali do strežnika OpenLDAP, ki je nato preveril pravilnost oziroma vsebovanost uporabniškega imena in njegovega gesla v njegovi podatkovni bazi Berkeley (angl. *Berkeley database*).

Za razliko od modula `Pam_Radius`, ki smo ga uporabili pri FreeRADIUS avtentikaciji Linux uporabnikov, omogoča modul `Pam_Ldap` varno povezavo med odjemalcem in strežnikom LDAP, kar smo v tem primeru omogočili med našim odjemalcem in strežnikom OpenLDAP v okviru procesa avtentikacije s pomočjo protokola TLS. Konfiguracijo in izvedbo testiranja smo prikazali v naslednjem podpoglavju.

4.4.1 Pam_Ldap modul in varnost

Da bi omogočili dostop našega odjemalca do strežnika OpenLDAP oziroma do njegove podatkovne baze Berkeley (angl. *Berkeley database*), smo morali ustrezno skonfigurairati `Pam_Ldap` modul na našem odjemalcu. V njegovi konfiguracijski datoteki `/etc/ldap.conf` smo nastavili naslednje podatke, kot prikazuje spodnja slika 40.

```
# The distinguished name of the search base.
base dc=my-domain,dc=com
# Another way to specify your LDAP server is to provide an
uri ldaps://88.200.24.242:636/
# The distinguished name to bind to the server with
# if the effective user ID is root. Password is
# stored in /etc/ldap.secret (mode 600)
rootbinddn cn=Mager,dc=my-domain,dc=com
# OpenLDAP SSL mechanism
ssl on
# CA certificates for server certificate verification
tls_cacertfile /etc/ssl/certs/ca.cert
# Client certificate and key
# Use these, if your server requires client authentication.
tls_cert /etc/ssl/certs/client.cert
tls_key /etc/ssl/certs/client.key
```

Slika 40: Del konfiguracijske datoteke modula `Pam_Ldap`.

Kot je razvidno z zgornje slike 40, smo nastavili razločevalno ime (angl. *distinguished name*) podatkovne baze Berkeley, naslov IP našega virtualnega strežnika, kjer se nahaja strežnik OpenLDAP, ter razločevalno ime, s katerim se bo modul `Pam_Ldap` prijavil (angl. *bind*) v strežnik OpenLDAP. Razločevalno ime podatkovne baze Berkeley smo podali kot vrednost atributa `base`, ki je v našem primeru `dc=my-domain,dc=com`. Kot vrednost atributa `uri`, smo poleg naslova IP strežnika OpenLDAP, podali še številko vrat (angl. *port*), na katerih posluša strežnik OpenLDAP, ki je v našem primeru 636. Številko vrat 636 smo določili z namenom, saj na teh vratih privzeto posluša strežnik OpenLDAP, kadar gre za varno komunikacijo med njim in odjemalcem. Z vrednostjo atributa `rootbindn` smo določili razločevalno ime (angl. *distinguished name*), s katerim se je modul `Pam_Ldap` prijavil v strežnik OpenLDAP in je v našem primeru `cn=Manager,dc=mydomain,dc=com`. Geslo za to razločevalno ime pa se nahaja v datoteki `etc/ldap.secret`. Poleg tega smo morali zaradi potrebe po varni komunikaciji z uporabo protokola TLS omogočiti mehanizem OpenLDAP SSL (`ssl on`) ter podati pot do potrebnih certifikatov. Z atributom `tls_cacertfile` smo podali pot do certifikata CA, ki

je izdala certifikat strežniku OpenLDAP. S tem smo našemu odjemalcu oziroma modulu `Pam_Ldap` dodali CA, kateri ta zaupa. Tako je naš odjemalec oziroma `Pam_ldap` modul lahko preveril, ali je CA, kateri zaupa, izdala certifikat strežniku OpenLDAP. Poleg certifikata CA smo podali še pot do certifikata našega odjemalca in njegovega zasebnega ključa. Pot do certifikata odjemalca smo podali kot vrednost atributa `tls_cert`, medtem ko smo pot do njegovega zasebnega ključa podali z vrednostjo atributa `tls_key`. S tem je lahko strežnik OpenLDAP opravil avtentikacijo našega odjemalca ter se tudi on prepričal o veljavnosti certifikata našega odjemalca.

Certifikate oziroma pot do njih smo morali navesti tudi na strani strežnika OpenLDAP, kjer smo podali pot do njegovega certifikata in zasebnega ključa ter pot do certifikata CA, ki je izdala certifikat odjemalcu in prav tako strežniku OpenLDAP. To smo nastavili v konfiguracijski datoteki `/etc/openldap/slapd.d/cn=config/olcdatabase={2}bdb.ldif`, ki je namenjena konfiguraciji podatkovne baze Berkeley (angl. *Berkeley database*), katero v našem primeru uporablja strežnik OpenLDAP. Dodane poti do potrebnih certifikatov prikazuje spodnja slika 41.

```
olcTLSCertificateFile: /etc/pki/tls/certs/ldap.cert
olcTLSCertificateKeyFile: /etc/pki/tls/certs/ldap.key
olcTLSCACertificateFile: /etc/pki/tls/certs/ca.cert
olcTLSVerifyClient: demand
```

Slika 41: Del konfiguracijske datoteke podatkovne baze Berkeley.

Kot vrednost atributa `olcTLSCerificateFile` smo podali pot do strežnikovega certifikata ter kot vrednost atributa `olcTLSCertificateKeyFile` tudi lokacijo njegova zasebnega ključa. Pot do zadnjega certifikata z zgornje slike 41, ki predstavlja pot do certifikata CA, smo podali kot vrednost atributa `olcTLSCACertificateFile`. Ta certifikat CA je enak tistemu na strani našega odjemalca ter ga strežnik OpenLDAP uporablja za isti namen, kot ga uporablja naš odjemalec. Tako lahko tudi strežnik OpenLDAP preveri, ali je izdajatelj odjemalčevega certifikata tista CA, kateri strežnik OpenLDAP zaupa. Ker tu nastopa ista CA, ki je izdala certifikat našemu odjemalcu in strežniku ter ji oba zaupata, to privede do vzajemne avtentikacije (angl. *mutual authentication*). Odjemalec in strežnik OpenLDAP avtenticirata drug drugega. Kot je razvidno z zgornje Slike 41, smo na koncu dodali tudi atribut `olcTLSVerifyClient` z vrednostjo `demand`. S tem smo strežniku OpenLDAP določili, naj zahteva certifikat našega odjemalca, ki mora biti veljaven. V

primeru, če naš odjemalec ne bi predložil certifikata ali pa bi bil ta neveljaven, se bi komunikacija med našim odjemalcem in strežnikom OpenLDAP nemudoma prekinila.

Zasebni ključ odjemalca in strežnika OpenLDAP in tudi vse certifikate smo generirali z uporabo ukaza `openssl` in njegovih ustreznih parametrov. Najprej smo kreirali svojo lastno CA, s katero smo lahko izdali oziroma podpisali certifikat za našega odjemalca in strežnik OpenLDAP. Tako smo kreirali zasebni ključ za CA ter s pomočjo zasebnega ključa kreirali njen certifikat, ki je v našem primeru `ca.cert`. Nato smo za strežnik OpenLDAP generirali zasebni ključ, ki je v našem primeru `ldap.key` ter ga uporabili pri kreiranju zahteve za podpis certifikata (angl. *certificate signing request*). Z uporabo zahteve za podpis certifikata, certifikata CA in njenega zasebnega ključa smo podpisali certifikat ter pridobili datoteko `ldap.cert`. Enak postopek smo ponovili tudi za našega odjemalca ter pridobili njegov certifikat `client.cert` in zasebni ključ `client.key`.

Ko smo opravili s konfiguracijo modula `Pam_Ldap` in kreirali potrebne certifikate, smo morali na strani strežnika OpenLDAP omogočiti uporabo protokola TLS za varno povezavo med našim odjemalcem in strežnikom OpenLDAP. To smo omogočili v konfiguracijski datoteki `/etc/sysconfig/ldap`, kjer smo nastavili atribut `SLAPD_LDAPS` na vrednost `yes`, kot prikazuje spodnja slika 42.

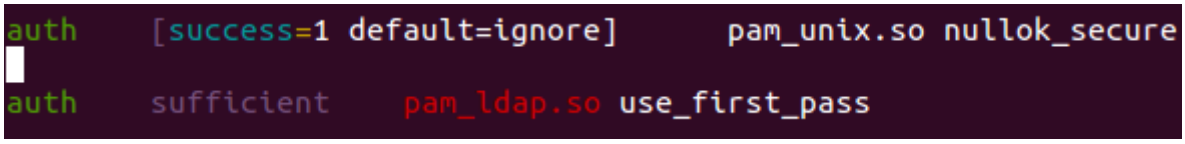
```
# Run slapd with -h "... ldaps:/// ..."
#   yes/no, default: no
SLAPD_LDAPS=yes
```

Slika 42: Del konfiguracijske datoteke `ldap`.

Po tej nastavitvi z zgornje slike 42 je strežnik OpenLDAP, imenovan `slapd`, omogočal varno povezavo s pomočjo protokola LDAPS, ki vključuje uporabo protokola TLS. S protokolom LDAPS smo tudi dostopali do strežnika OpenLDAP, kot je to razvidno iz zgornje slike 40 pod vrednostjo atributa `uri`.

Preden smo lahko pričeli s testiranjem LDAP avtentikacije, smo morali v konfiguracijsko datoteko `/etc/pam.d/common-auth` na strani našega odjemalca dodati modul `pam_ldap.so`. Tega smo dodali za modulom `pam_unix.so`, ki predstavlja standardni avtentikacijski modul v operacijskem sistemu Linux, saj ta avtenticira uporabnika s pomočjo datoteke `/etc/passwd` in `etc/shadow`, kjer pridobi pravilno uporabniško ime in pripadajoče geslo. Ker smo dodali modul `pam_ldap.so` v datoteko

`/etc/pam.d/common-auth`, smo omogočili LDAP avtentikacijo za vse storitve, ki uporabljajo mehanizem PAM, saj je ta konfiguracijska datoteka skupna za vse te storitve v okviru procesa avtentikacije (ta datoteka je namreč vključena v vseh PAM konfiguracijskih datotekah, kjer je določena namenjena eni storitvi). Dodan modul `pam_dap.so` v datoteki `/etc/pam.d/common-auth` prikazuje spodnja slika 43.



Slika 43: Del konfiguracijske datoteke `common-auth`.

Modulu `pam_ldap.so` smo določili tip `auth` in nadzorno zastavico (angl. *control flag*) `sufficient`, kot je razvidno z zgornje slike 43. Tip modula `auth` označuje, da bo modul `pam_ldap.so` izvedel avtentikacijo uporabnika, kar se je v našem primeru zgodilo s posredovanjem uporabniškega gesla na strežnik OpenLDAP. Nadzorna zastavica `sufficient` označuje, da se ob uspešni izvedbi modula `pam_ldap.so` ne izvrši noben izmed naslednjih modulov, vendar le v primeru, če se vsi moduli pred modulom `pam_ldap.so`, ki imajo nadzorno zastavico `required`, izvršijo uspešno [30, 32]. Poleg teh dveh komponent smo dodali tudi argument modula, katerega vrednost je `use_first_pass` (namreč format v konfiguracijskih datotekah PAM zajema tudi možnost podanega argumenta, s katerim lahko posredujemo določeno informacijo modulu med samo avtentikacijo). V našem primeru smo z atributom `use_first_pass` sporočili modulu, naj uporabi prvo podano geslo ter posledično ne zahteva gesla uporabnika z administratorskimi pravicami nad podatkovno bazo Berkeley.

Po dodanem modulu `pam_ldap.so` smo lahko pričeli s testiranjem LDAP avtentikacije, in sicer na naslednji način: na našem odjemalcu smo v novem terminalu izvedli ukaz `sudo ls` ter podali uporabniško geslo za uporabnika `matej` (s tem uporabnikom smo bili prijavljeni na našem odjemalcu). Modul `pam_ldap.so` je v tem trenutku dostopal do strežnika OpenLDAP ter se avtenticiral z razločevalnim imenom `cn=Manager,dc=my-domain,dc=com`, ki smo ga določili v konfiguracijski datoteki `/etc/ldap.conf`. Prav tako je modul `pam_ldap.so` posredoval uporabniško ime `matej` in njegovo geslo strežniku OpenLDAP, ki je nato preveril obstoj uporabnika z njegovim geslom v podatkovni bazi Berkeley. Uporabnik `matej` in njegovo geslo je seveda obstajalo v podatkovni bazi Berkeley, saj smo ga že dodali v podatkovno bazo Berkeley skupaj z njegovim geslom. Tako se je modul `pam_ldap.so` izvedel uspešno in avtentikacija uporabnika je uspela s pomočjo

protokola LDAPS. V terminalu smo nato pridobili izpis trenutnega imenika, kot to prikazuje zgornja slika 37.

Pri primerjanju FreeRADIUS avtentikacije Linux uporabnikov, kjer smo uporabili modul `Pam_Radius`, ter OpenLDAP avtentikacije z uporabo modula `Pam_Ldap` smo opazili pomembno razliko. Modul `Pam_Ldap` omogoča varno povezavo med njim in strežnikom OpenLDAP z uporabo protokola TLS, kar pri modulu `Pam_Radius` ni mogoče. Tako je v primeru OpenLDAP avtentikacija varnejša zaradi modula `pam_ldap`.

4.5 Kerberos avtentikacija

Avtentikacijo s pomočjo protokola KERBEROS smo pridobili v okviru produkta FreeIPA, ki znotraj domene FreeIPA zagotavlja centralno KERBEROS avtentikacijo. Na naš virtualni strežnik smo namestili strežnik FreeIPA in odjemalca FreeIPA ter oba ustrezno skonfigurirali s pomočjo že pripravljenih skript. Nato smo z uporabo spletnega uporabniškega vmesnika FreeIPA (angl. *FreeIPA WEB user interface*) za upravljanje s strežnikom FreeIPA ustvarili novega uporabnika v domeni FreeIPA (s tem uporabnikom smo se nato prijavili v naš virtualni strežnik). Z namestitvijo odjemalca FreeIPA je naš virtualni strežnik postal tudi odjemalec strežnika FreeIPA, kar nam je omogočilo prijavo z uporabnikom, ki smo ga ustvarili z uporabo spletnega uporabniškega vmesnika FreeIPA. Do spletnega uporabniška vmesnika FreeIPA smo dostopali iz našega odjemalca, ki ni bil vključen v domeni FreeIPA. Za uporabo spletnega uporabniškega vmesnika FreeIPA smo se seveda morali uspešno avtenticirati. Celoten potek smo podrobneje zajeli v nadaljevanju.

4.5.1 Strežnik FreeIPA

Pričeli smo z nameščanjem paketa `ipa-server`, ki predstavlja strežnik FreeIPA. Namestitev strežnika FreeIPA vključuje številne storitve, med katerimi je imeniška storitev 389 (angl. 389 *directory server base*) za shranjevanje podatkov, Kerberos strežnik (angl. *krb5-server*), strežnik NTP in strežnik HTTP. Paket `ipa-server` zajema tudi množico orodij (angl. *FreeIPA-tools*), s katerim je omogočeno upravljanje s strežnikom FreeIPA. Namestitev strežnika FreeIPA ni vključevala storitev DNS, saj jo za naše zahteve nismo potrebovali.

Po namestitvi paketa `ipa-server` smo morali strežnik FreeIPA ustrezno skonfigurirati. To smo opravili s pomočjo že pripravljene skripte, ki vključuje konfiguracijo storitev, ki smo jih

namestili v okviru namestitve strežnika FreeIPA. Skripto smo pognali z ukazom `ipa-server-install`, ki požene skripto interaktivno. Tako smo v času izvrševanja skripte podajali zahtevane podatke z namenom konfiguracije strežnika FreeIPA. Podatke, ki smo jih morali posredovati, so naslednji:

- polno ime sistema (angl. *fully qualified domain name*), ki predstavlja ime sistema na katerem smo namestili strežnik FreeIPA. Ta je v našem primeru `radius.example.com`, ki smo ga določili v datoteki `/etc/hosts`. Tu smo podali preslikavo med polnim imenom sistema in njegovim naslovom IP,
- ime domene (angl. *domain name*), ki je v našem primeru `example.com`, saj je ta del polnega imena sistema,
- ime Kerberos področja (angl. *Kerberos realm name*), ki ga zahteva protokol KERBEROS. Ta je običajno ime domene, ki je podano z veliki tiskanimi črkami. Tako je ime Kerberos področja v našem primeru `EXAMPLE.COM`,
- geslo skrbnika (angl. *superuser*) za imeniško storitev, ki ima polni dostop (angl. *full access*) do imeniške storitve,
- geslo za uporabnika `admin`, ki predstavlja uporabnika z administratorskimi pravicami. Uporabnik je namenjen upravljanju s strežnikom FreeIPA.

Izvedena skripta je kreirala ter ustrezno nastavila imeniško storitev in Kerberos center za distribucijo ključev (angl. *Kerberos key distribution center*). Prav tako je skripta ustrezno nastavila strežnik NTP in HTTP, imenovan Apache. Na ta način smo strežnik ustrezno skonfigurirali.

4.5.2 Upravljanje s strežnikom FreeIPA s pomočjo spletne aplikacije

Strežnik FreeIPA nudi upravljanje s pomočjo orodij ukazne vrstice (angl. *command-line tools*) ter prav tako s pomočjo spletnega uporabniškega vmesnika FreeIPA (angl. *FreeIPA WEB user interface*). Obe možnosti upravljanja nudita administratorjem upravljanje z domeno FreeIPA ter s storitvami, ki jih ta vključuje. Tako imata upravljanje s pomočjo orodij ukazne vrstice kakor tudi upravljanje s pomočjo spletnega vmesnika FreeIPA določene prednosti oziroma slabosti [21].

Upravljanje s pomočjo orodij ukazne vrstice ima v primerjavi z upravljanjem s pomočjo spletnega uporabniškega vmesnika FreeIPA, dve naslednji prednosti [21]:

- vsi predmeti so lahko dodani v imeniško storitev z vsemi možnimi atributi v enem samem koraku. V primeru spletnega uporabniškega vmesnika FreeIPA je treba namreč najprej kreirati predmet ter mu nato dodati neobvezne atribute,
- z uporabo upravljanja s pomočjo orodij ukazne vrstice je možno dodati vse neobvezne atribute za določen predmet, medtem ko v primeru spletnega uporabniškega vmesnika FreeIPA nekateri niso vključeni. Tako jih ni možno dodati k samemu predmetu.

Kljub temu da ima upravljanje s pomočjo orodij ukazne vrstice omenjeni prednosti, je spletni uporabniški vmesnik FreeIPA zasnovan za enostavnejše upravljanje z domeno FreeIPA. Tako je spletni uporabniški vmesnik FreeIPA bolj priljubljen, saj omogoča enostavnejše upravljanje z domeno FreeIPA kot orodja s pomočjo ukazne vrstice . Enostavnejše upravljanje je razvidno iz naslednjih lastnosti [21]:

- poznavanje ukazov z namenom upravljanja ni potrebno, kot to velja v primeru upravljanja s pomočjo orodij ukazne vrstice,
- spletni uporabniški vmesnik FreeIPA je dostopen tudi sistemom, ki se ne nahajajo v domeni FreeIPA. Tako je možno upravljanje iz poljubnega sistema. Seveda je za dostop do spletnega uporabniška vmesnika potrebna avtentikacija,
- pregleden nadzor nad vsemi predmeti, ki se nahajajo v imeniški storitvi omogoča, da so vsi predmeti vidni brez potrebe po izvršitvi ukaza za iskanje predmetov. Poleg tega spletni uporabniški vmesnik FreeIPA nudi polje za iskanje predmetov.

V obeh primerih upravljanja s strežnikom FreeIPA je seveda potrebna predhodna avtentikacija, ki jo opravi Kerberos center za distribucijo ključev (angl. *Kerberos key distribution center*) znotraj domene FreeIPA. Tako tu nastopi avtentikacija s pomočjo protokola KERBEROS, ki za ta proces uporablja vstopnice (angl. *tickets*). V našem primeru smo za upravljanje s strežnikom FreeIPA uporabili spletni uporabniški vmesnik FreeIPA . Preden smo ga lahko uporabili na našem odjemalcu, smo se morali uspešno avtenticirati oziroma vpisati (angl. *login*) v domeno FreeIPA.

Na našega odjemalca smo morali najprej namestiti paket `heimdal-clients`, ki predstavlja implementacijo protokola KERBEROS ter vključuje potrebna orodja za podporo KERBEROS avtentikacije. Nato smo iz strežnika FreeIPA prekopirali konfiguracijsko datoteko `/etc/krb5.conf` na našega odjemalca ter omogočili njeno uporabo namesto že obstoječe, ki jo je vključil nameščeni paket `heimdal-clients`. Datoteka `/etc/krb5.conf` predstavlja konfiguracijsko datoteko za Kerberos področje (angl. *Kerberos realm*) `EXAMPLE.COM` znotraj domene FreeIPA. Ta vsebuje številne podatke, med katerimi so naslednji:

- ime Kerberos področja in privzeto Kerberos področje. V našem primeru sta oba atributa nastavljena na vrednost `EXAMPLE.COM`,
- čas veljavnosti vstopnice, ki je v našem primeru 24 ur,
- naslov IP Kerberos centra za distribucijo ključev ter številka vrat (angl. *port*); prvi je v našem primeru nastavljen na naslov IP našega virtualnega strežnika, medtem ko je številka vrat nastavljena na 88,
- privzeto ime domene (angl. *domain name*), ki je v našem primeru `example.com`. To ime smo nastavili pri konfiguraciji strežnika FreeIPA.

Spodnja slika 44 prikazuje del konfiguracijske datoteke `/etc/krb5.conf`, ki prikazuje nastavitev omenjenih podatkov.

```
[libdefaults]
 default_realm = EXAMPLE.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false
 rdns = false
 ticket_lifetime = 24h
 forwardable = yes

[realms]
 EXAMPLE.COM = {
  kdc = 88.200.24.242:88
  admin_server = 88.200.24.242:749
  default_domain = example.com
  pkinit_anchors = FILE:/etc/ipa/ca.crt
}
```

Slika 44: Del konfiguracijske datoteke `krb5.conf`.

Konfiguracijsko datoteko `/etc/krb5.conf` smo morali prekopirati na našega odjemalca, saj ta ni član domene FreeIPA. S tem smo na našem odjemalcu omogočili avtentikacijo s pomočjo protokola KERBEROS s strani strežnika FreeIPA, saj je lahko naš odjamalec dostopal do Kerberos centra za distribucijo ključev znotraj domene FreeIPA.

Nato smo se lahko uspešno avtenticirali oziroma vpisali (angl. *login*) v domeno FreeIPA, pri čemer smo uporabili uporabnika `admin`, za katerega smo nastavili geslo pri konfiguraciji strežnika FreeIPA. Na našem odjemalcu smo v terminalu izvedli ukaz `kinit` ter mu dodali kot parameter uporabnika `admin`. S tem smo pridobili začetno vstopnico (angl. *ticket granting ticket*) s strani kerberos centra za distribucijo ključev ter nato posredovali geslo uporabnika `admin`, da bi uspešno dešifrirali začetne vstopnice in posledično uspešno izvedli avtentikacijo. V primeru napačnega gesla se začetna vstopnica ne bi dešifrirala, kar bi

povzročilo, da se tudi ne bi mogli avtenticirati oziroma vpisati v domeno FreeIPA z uporabnikom `admin`.

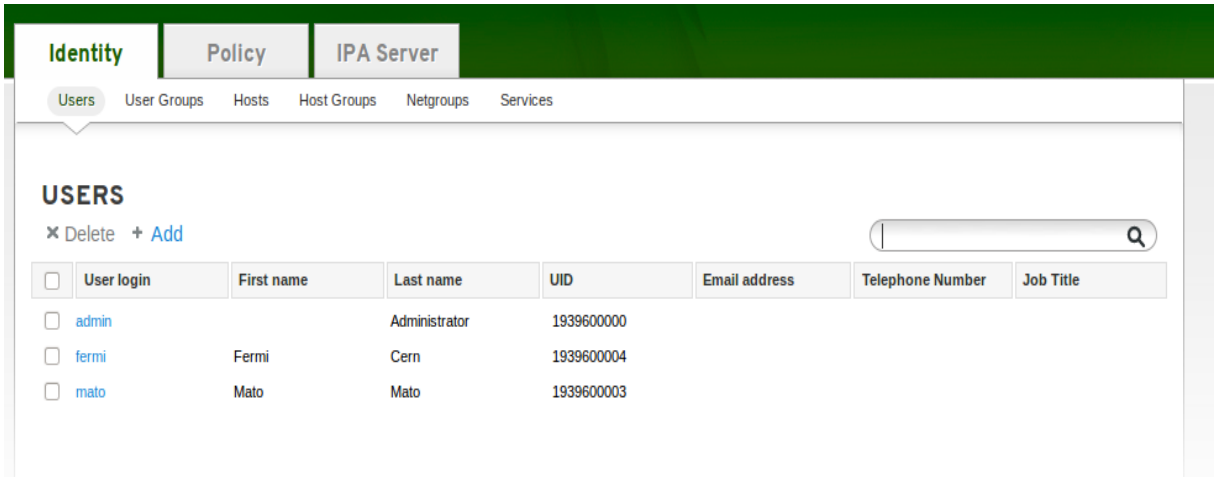
Po uspešni avtentikaciji oziroma po uspešnem vpisu v domeno FreeIPA smo morali pridobiti še tako imenovano vstopnico za določeno storitev (angl. *service ticket*). V našem primeru smo morali pridobiti vstopnico (angl. *ticket*) za storitev HTTP, ki se nahaja v domeni FreeIPA, saj smo želeli dostopati do spletnega uporabniškega vmesnika FreeIPA . Z uporabo spletnega brskalnika Firefox in s spletnim naslovom https://radius.example.com/ipa/ui smo dostopali do spletnega uporabniškega vmesnika FreeIPA ter obenem pridobili vstopnico za storitev HTTP na podlagi že pridobljene začetne vstopnice. Naš odjemalec je pridobil vstopnico za storitev HTTP ob priložitvi že pridobljene začetne vstopnice, katere veljavnost je preveril Kerberos center za distribucijo ključev znotraj domene FreeIPA, ter našemu odjemalcu izdal vstopnico za storitev HTTP. Spodnja slika 45 prikazuje pridobitev obeh vstopnic na našem odjemalcu. Izpis smo izvedli z uporabo ukaza `klist`.

```
matej@mato-laptop:~$ klist
Credentials cache: FILE:/tmp/krb5cc_1000
        Principal: admin@EXAMPLE.COM

  Issued                Expires               Principal
Aug 23 11:19:44 2012  Aug 24 11:19:44 2012  krbtgt/EXAMPLE.COM@EXAMPLE.COM
Aug 23 11:52:01 2012  Aug 24 11:19:44 2012  HTTP/radius.example.com@EXAMPLE.COM
matej@mato-laptop:~$
```

Slika 45: Izpis obeh pridobljenih vstopnic.

Nato smo z uporabo spletnega uporabniškega vmesnika FreeIPA dodali dva nova uporabnika v domeni FreeIPA, ki smo ju potrebovali za testne namene. Dodana uporabnika prikazuje spodnja slika 46.



Slika 46: Spletni uporabniški vmesnik FreeIPA.

Kot je razvidno z zgornje slike 46, smo dodali uporabnika `fermi` in `mato`, medtem ko je uporabnik `admin` že bil kreiran, in sicer, pri konfiguraciji strežnika FreeIPA. Uporabnika `admin` smo uporabili za dostop do spletnega uporabniškega vmesnika FreeIPA, saj ima ta administratorske pravice (tako smo lahko dodali dva nova uporabnika). Nato smo lahko pričeli s testiranjem vpisa v naš virtualni strežnik, in sicer z uporabo obeh dodanih uporabnikov. To smo prikazali v naslednjem podpoglavju.

Poleg avtentikacije s pomočjo protokola KERBEROS bi lahko za dostop do spletnega uporabniškega vmesnika FreeIPA uporabili avtentikacijo s pomočjo gesla (angl. *password-base authentication*). Uporaba avtentikacije s pomočjo gesla je uporabna v primeru, ko avtentikacija ne uspe s pomočjo protokola KERBEROS ali v primeru, ko sistem ni del domene FreeIPA [21]. V našem primeru se naš odjemalec ni nahajal v domeni FreeIPA, vendar smo vseeno omogočili avtentikacijo s pomočjo protokola KERBEROS.

4.5.3 Odjemalec FreeIPA

Preden smo lahko pričeli s testiranjem prijave v naš virtualni strežnik, smo morali nanj namestiti še odjemalca FreeIPA ter ga ustrezno skonfigurairati. Tako smo namestili paket `ipa-client`, ki predstavlja odjemalca FreeIPA, ki smo ga skonfigurirali s pomočjo že pripravljene skripte `ipa-client-install`. Po namestitvi odjemalca FreeIPA smo pognali skripto `ipa-client-install`, ki je med konfiguracijo zahtevala določene podatke. Podatke, ki smo jih posredovali skripti med konfiguracijo, so naslednji:

- ime domene (angl. *domain name*) strežnika FreeIPA, ki je v našem primeru `example.com`. To ime smo definiriali pri konfiguraciji strežnika FreeIPA,
- polno ime strežnika FreeIPA (angl. *fully qualified domain name*), katerega smo prav tako definirali pri konfiguraciji strežnika FreeIPA ter je v našem primeru `radius.example.com`,
- uporabnika, s katerim smo dostopali do področja Kerberos (angl. *Kerberos realm*) ter se vanj pridružili; tu smo z uporabo `admin` in njegova gesla omogočili odjemalcu FreeIPA pridružitev v domeno FreeIPA.

Po izvršitvi skripte `ipa-client-install` se je naš virtualni strežnik pridružil domeni FreeIPA kot njen odjemalec. Nato smo lahko opravili testiranje prijave vanj z uporabnikoma `fermi` in `mato`, katera smo dodali s pomočjo spletnega uporabniškega vmesnika FreeIPA (angl. *FreeIPA WEB user interface*) na strani našega odjemalca (prijavili smo se lahko le v sistem, ki predstavlja odjemalca FreeIPA in je ustrezno skonfiguriran za povezavo do strežnika FreeIPA). Testiranje prijave smo izvedli oddaljeno od našega odjemalca, pri čemer smo uporabili protokol SSH. Kot parameter ukaza `ssh` smo podali uporabniško ime `fermi` oziroma `mato`, naslov IP našega virtualnega strežnika in številko vrat (angl. *port*). Izvedbo oddaljene prijave z uporabnikom `fermi` prikazuje spodnja slika 47.

```
matej@mato-laptop:~$ ssh fermi@88.200.24.242 -p 2222
fermi@88.200.24.242's password:
Last login: Thu Aug 23 16:12:18 2012 from lk.46.182.230.131.dc.cable.static.lj-kabel.net
Could not chdir to home directory /home/fermi: No such file or directory
-bash-4.1$
```

Slika 47: Oddaljena prijava v naš virtualni strežnik.

Po izvedbi ukaza smo seveda morali podati geslo za uporabnika `fermi` ter se nato uspešno prijavili v naš virtualni strežnik, kot je razvidno z zgornje slike 47. Tu ni bilo treba predhodno lokalno kreirati uporabnika `fermi` na našem virtualnem strežniku, kar je prednost produkta FreeIPA pred produktom FreeRADIUS in OpenLDAP. V primeru produkta FreeRADIUS in pri produktu OpenLDAP smo potrebovali predhodno lokalno kreiranega uporabnika za uspešen vpis v sistem.

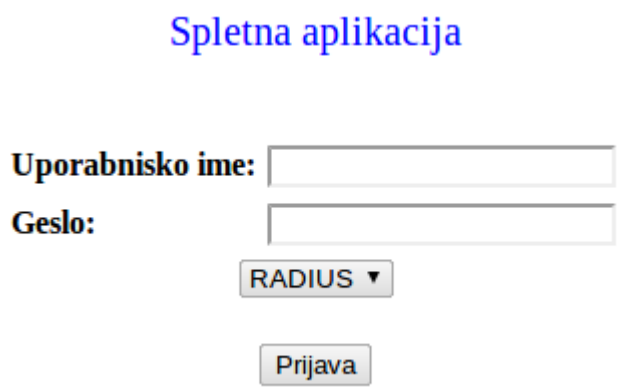
4.6 Spletna aplikacija s primerom RADIUS in LDAP avtentikacije

Z namenom prikaza uporabe RADIUS in LDAP avtentikacije smo razvili spletno aplikacijo. Tako smo prikazali, da lahko za dostop do spletnih aplikacij uporabimo prav tako RADIUS

ali LDAP avtentikacijo, ne le za prijavo v sistem. Tako smo z uporabo razvite spletne aplikacije testirali FreeRADIUS avtentikacijo s pomočjo podatkovne MySQL in strežnika OpenLDAP ter prav tako brez uporabe zunanjega vira. Poleg tega smo testirali tudi LDAP avtentikacijo, kjer smo neposredno dostopali do strežnika OpenLDAP brez uporabe strežnika FreeRADIUS.

Za spletni strežnik smo uporabili naš virtualni strežnik ter nanj namestili paket `httpd`, ki predstavlja strežnik HTTP. Poleg tega smo namestili še paket `php`, `php-pecl-radius` in paket `php-ldap`. Paket `php` nam je omogočil uporabo skriptnega jezika PHP, medtem ko smo paketa `php-pecl-radius` in `php-ldap` potrebovali za uporabo določenih ukazov, ki smo jih potrebovali za razvitje spletne aplikacije. Tako smo za razvoj spletne aplikacije uporabili skriptni jezik PHP.

Strežnik HTTP, ki smo ga namestili na naš virtualni strežnik, servira datoteke znotraj imenika `/var/www/html/`, ki so dostopne s pomočjo protokola HTTP. Tako smo vanj dodali datoteko `login.html`, v kateri smo z uporabo označevalnega jezika HTML kreirali prijavno okno ter datoteko `usmeri.php`, v kateri smo uporabili skriptni jezik PHP za dostop do strežnika FreeRADIUS oziroma OpenLDAP. Na ta način smo lahko iz našega odjemalca z uporabo spletnega brskalnika dostopali do datoteke `login.html`, ki je prikazala prijavno okno, kot prikazuje spodnja slika 48.



Slika 48: Uporabniški vmesnik spletne aplikacije.

V prijavno okno smo podali uporabniško ime in geslo določenega uporabnika ter ob kliku na gumb `Prijava`, sprožili izvedbo datoteke `usmeri.php`. Ta je nato, z uporabo skriptnega jezika PHP, dostopala do strežnika FreeRADIUS oziroma OpenLDAP ter posredovala uporabniško ime in geslo. Strežnik FreeRADIUS oziroma OpenLDAP je preveril pravilnost uporabniškega imena in gesla ter vrnil odgovor skripti PHP. Pridobljeni odgovor smo seveda

izpisali ter tako navedli uspešnost avtentikacije. V nadaljevanju smo prikazali datoteko `login.html` in `usmeri.php` ter komentirali njuno vsebino. Spodnja slika 49 prikazuje vsebino datoteke `login.html`.

```
<html>
<head>
<title>Testna Stran</title>
<style type="text/css">
</style>
</head>
<body>
<div align="center">
<p style="color:blue;font-size:20px;">Spletna aplikacija</p><br>
<form method="POST" action="usmeri.php">
<table>
<tr><td><strong>Uporabnisko ime:</strong></td>
<td><input type="text" name="upoime"></td></tr>
<tr><td><strong>Geslo:</strong></td>
<td><input type="password" name="geslo"></td></tr>
</table>
<select name="izberi">
  <option value="radius">RADIUS</option>
  <option value="ldap">LDAP</option>
</select><br><br>
<input type="submit" value="Prijava">
</form>
</div>
</body>
</html>
```

Slika 49: Datoteka `login.html`.

Kot je razvidno z zgornje slike 49, smo poleg polja za uporabniško ime in polja za geslo dodali tudi izvlečno listo (angl. *drop-down list*) z uporabo značke `select`. Z uporabo izvlečne liste smo namreč omogočili izbiro avtentikacije, ki je v našem primeru RADIUS ali LDAP. Poleg tega smo dodali gumb, imenovan `Prijava`, s katerim smo omogočili izvedbo datoteke `usmeri.php`. Vse to smo dodali znotraj obrazca (angl. *form*) ter določili metodo `POST`, ki označuje pošiljanje vnesenih podatkov na strežnik, v našem primeru na FreeRADIUS oziroma OpenLDAP. Prav tako smo kot vrednost atributa `action` znotraj obrazca dodali ime datoteke, ki se izvrši ob kliku na gumb `Prijava`. To je seveda datoteka `usmeri.php`, ki jo prikazuje spodnja slika 50 in je ključnega pomena.

```php
<?php

if ( empty($_POST['upoime']) || empty($_POST['geslo']) ){
        $host=$_SERVER['HTTP_HOST'];
        header("Location: http://$host/login.html");
}

$user=$_POST['upoime'];
$pass=$_POST['geslo'];
$meni=$_POST['izberi'];

if($meni=='radius') {
        $radh = radius_auth_open();

        if (!radius_add_server($radh, 'localhost', 1812, 'skrivnost', 3, 3)) {
                echo "Napaka pri dodajanju strežnika!";
                exit;
        }

        if (!radius_create_request($radh, RADIUS_ACCESS_REQUEST)) {
                echo "Napaka pri kreiranju zahteve!";
                exit;
        }

        radius_put_attr($radh,RADIUS_USER_NAME,$user);
        radius_put_attr($radh,RADIUS_USER_PASSWORD,$pass);

        switch (radius_send_request($radh)) {
        case RADIUS_ACCESS_ACCEPT:
          echo "Avtentikacija uspela!";
          break;
        case RADIUS_ACCESS_REJECT:
          echo "Avtentikacija ni uspela!";
          break;
        case RADIUS_ACCESS_CHALLENGE:
          echo "Zahtevan izziv";
          break;
        default:
          echo "RADIUS se ne odziva!";
        }
}
```

Slika 50: Prvi del datoteke `usmeri.php`.

Ob kliku na gumb prijava smo najprej preverili, ali je uporabnik podal uporabniško ime in geslo. Kadar ta ni podal uporabniškega imena ali gesla, smo uporabnika preusmerili na začetek (`login.html`). Tako je uporabnik ponovno pridobil prijavno okno ter moral vnesti manjkajoče podatke z namenom pridobitve odgovora o uspešni avtentikaciji. To smo omogočili s prvim stavkom IF v datoteki `usmeri.php`, kot je razvidno z zgornje slike 50. Nato smo z uporabo funkcije `$_POST` pridobili uporabniško ime in geslo, ki ga je vnesel uporabnik ter prav tako pridobili izbrano avtentikacijo uporabnika (RADIUS ali LDAP).

V nadaljevanju smo v okviru stavka IF zajeli RADIUS avtentikacijo. Kadar je uporabnik izbral avtentikacijo RADIUS s pomočjo izvlečne liste, se je izvršil ta stavek IF. Znotraj stavka IF smo najprej z uporabo funkcije `radius_auth_open()` kreirali ročico (angl. *handle*) za

proces avtentikacije. To ročico smo uporabili v vseh naslednjih funkcijah. Z uporabo naslednje funkcije `radius_add_server` smo dodali strežnik FreeRADIUS oziroma definirali dostopne podatke (naslov IP strežnika FreeRADIUS, številka vrat (angl. *port*), skupna skrivnost (angl. *shared secret*), itd.). Nato smo z uporabo funkcije `radius_create_request` kreirali zahtevo `Access-Request` ter ji dodali vrednosti prilastka `RADIUS_USER_NAME` in `RADIUS_USER_PASSWORD`. Tako smo zahtevi `Access-Request` dodali pridobljeno uporabniško ime in geslo, in sicer z uporabo naslednje funkcije `radius_put_attr`. Zahtevo `AccessRequest` smo nato poslali z uporabo funkcije `radius_send_request`, katero smo podali kot parameter stavku SWITCH. Ta stavek nam je omogočil, da smo lahko ob določenem odgovoru strežnika ustrezno reagirali, kar smo opravili z ustreznim izpisom. Tako smo za vsak pridobljen odgovor (`RADIUS_ACCESS_ACCEPT`, `RADIUS_ACCESS_REJECT` ali `RADIUS_ACCESS_CHALLENGE`) izpisali ustrezen niz ter tako seznanili uporabnika o uspešnosti avtentikacije. Prav tako smo v primeru neodzivnosti strežnika FreeRADIUS, izpisali ustrezen niz.

Kadar je uporabnik izbral avtentikacijo LDAP namesto RADIUS s pomočjo izvlečne liste, se je izvršil stavek ELSE, ki zajema LDAP avtentikacijo. Vsebino stavka ELSE prikazuje spodnja slika 51.

```
} else {
        $hash=sha1($pass);

        $host="localhost";
        $port=389;

        $ldapconn = ldap_connect($host, $port);

        if ($ldapconn) {

                $ldapbind = ldap_bind($ldapconn);

                if (!$ldapbind) {
                        echo "Neuspešna anonimna prijava";
                }
        } else {
                echo "LDAP se ne odziva!";
        }

        $dn = "dc=my-domain, dc=com";
        $filter="(&(uid=$user)(userPassword=$hash))";
        $just = array("uid");

        $sr=ldap_search($ldapconn, $dn, $filter, $just);

        $info = ldap_get_entries($ldapconn, $sr);

        if ($info["count"]==1){
                echo "Avtentikacija uspela!";
        }
        else{
                echo "Avtentikacija ni uspela!";
        }
}
?>
```

Slika 51: Drugi del datoteke `usmeri.php`.

Na začetku stavka ELSE, kot je razvidno z zgornje slike 51, smo vzpostavili povezavo s strežnikom OpenLDAP z uporabo funkcije `ldap_connect`, ki smo ji podali naslov IP strežnika OpenLDAP in številko vrat. Naslov IP strežnika OpenLDAP je seveda v našem primeru `localhost`, saj se strežnik HTTP nahaja na istem strežniku kot OpenLDAP. Nato smo se z uporabo funkcije `ldap_bind` anonimno prijavili (angl. *bind*) v strežnik OpenLDAP ter ob morebitni neuspešni prijavi izpisali ustrezno sporočilo. Prav tako smo v primeru neuspešne vzpostavljene povezave s strežnikom OpenLDAP izpisali ustrezno sporočilo. Po uspešni anonimni prijavi smo nad strežnikom OpenLDAP izvedli funkcijo `ldap_search`, s katero smo pridobili informacijo o vsebovanosti predmeta z uporabniškim imenom in geslom v podatkovni bazi Berkeley (angl. *Berkeley database*), ki smo ju pridobili v prvem delu datoteke `usmeri.php` (z uporabo funkcije `$_POST`). Funkciji `ldap_search` smo kot parameter podali razločevalno ime (angl. *distinguished name*) podatkovne baze Berkeley, filter, s katerim smo podali iskano uporabniško ime in geslo, ter

tabelo zahtevanih atributov v iskanih predmetih. V našem primeru smo podali le atribut `uid`, saj smo iskali le predmete s tem atributom. Rezultat funkcije `ldap_search` smo podali kot argument naslednji funkciji `ldap_get_entires`, katera je nato vrnila iskane predmete v tabeli. Nato smo v zadnjem stavku IF preverili, ali pridobljena tabela vsebuje predmet ter izpisali ustrezen niz. Kadar se v tabeli nahaja predmet, smo dodali izpis o uspešni avtentikaciji, medtem ko smo v nasprotnem primeru dodali izpis o neuspešni avtentikaciji.

Kot je razvidno z zgornje slike 51, smo na začetku stavka ELSE uporabili funkcijo `sha1` ter z njo šifrirali pridobljeno uporabniško geslo, katerega smo nato uporabili kot filter pri argumentu funkcije `ldap_search`. Pri LDAP avtentikaciji smo morali šifrirati uporabniško geslo za primerjavo s tistim v podatkovni bazi Berkeley, saj v nasprotnem primeru nismo pridobili odgovora, ki označuje uspešno avtentikacijo. Prav tako smo pri testiranju LDAP avtentikacije s pomočjo spletne aplikacije dodali novega uporabnika (`metod`) v podatkovno bazo Berkeley, s katerim smo opravili testiranje LDAP avtentikacije. Pri tem uporabniku (`metod`) smo njegovo geslo šifrirali z uporabo funkcije `sha1` ter tako omogočili ujemanje s pridobljenim uporabniškim geslom znotraj stavka ELSE, saj smo tega prav tako šifrirali z uporabo funkcije `sha1` (uporabnik `metod` je bil dodan z enakimi atributi kot uporabnik z zgornje slike 31, le da smo njegovo geslo šifrirali z uporabo zgoščevalne funkcije SHA-1 namesto s SSHA). Tako je avtentikacija v tem primeru uspela in pridobili smo izpis o uspešni avtentikaciji.

5 Sklepne ugotovitve

V okviru celotnega diplomskega dela smo pridobili veliko novega oziroma razširjenega znanja. V prvem delu, kjer smo prikazali pregled področja, smo se spoznali s podrobnostmi pri treh avtentikacijskih protokolih (RADIUS, LDAP in KERBEROS). Prav tako smo pridobili več znanja na področju računalniške varnosti, saj smo poleg avtentikacijskih protokolov predstavili sodobne kriptografske metode za zagotavljanje zaupnosti in številne zgoščevalne funkcije, ki nam omogočajo preverjanje integritete posameznega sporočila. Poleg tega smo se podrobneje seznanili z varno infrastrukturo PKI in s protokolom SSL/TLS, katerega uporaba je danes zelo razširjena. V nadaljevanju diplomskega smo se spoznali s konkretnimi produkti, ki predstavljajo avtentikacijske strežnike, ter nato ob primerih izvedli primerjavo avtentikacije z izbranimi produkti, kar nam je omogočilo natančno seznanitev s temi produkti (konfiguracija strežnikov, testiranje avtentikacije, slabe in dobre lastnosti izbranih avtentikacijskih strežnikov itd.).

Primerjava avtentikacije s pomočjo izbranih produktov je prikazala različne možnosti avtentikacije uporabnikov, da bi se prijavili v sistem (v primeru RADIUS in LDAP avtentikacije smo prikazali tudi možnost prijave v spletno aplikacijo). Poleg tega so iz te primerjave razvidne tudi prednosti oziroma slabosti pri določeni avtentikaciji. V primeru RADIUS avtentikacije Linux uporabnikov smo naleteli na veliko pomanjkljivost modula `Pam_Radius`, saj ta omogoča le pošiljanje PAP sporočil ter posledično ne omogoča varne komunikacije med odjemalcem in strežnikom (prenos gesla je v čistopisu (angl. *cleartext*)). Na drugi strani, v primeru LDAP avtentikacije Linux uporabnikov, modul `Pam_Ldap` omogoča varno komunikacijo s strežnikom LDAP, saj smo uporabili protokol TLS (na ta način se je geslo do strežnika LDAP preneslo šifrirano). Tako ima modul `Pam_Ldap` v tem primeru veliko prednost pred modulom `Pam_Radius`, saj je šifrirana komunikacija med odjemalcem in strežnikom ključnega pomena. V obeh primerih avtentikacije (RADIUS in LDAP avtentikacija Linux uporabnikov) pa smo naleteli na manjšo pomanjkljivost, saj smo morali predhodno lokalno kreirati uporabnika, da smo se lahko z njim prijavili v sistem. To manjšo pomanjkljivost odpravlja produkt FreeIPA s pomočjo katerega smo zajeli KERBEROS avtentikacijo, saj tu nismo potrebovali predhodno lokalno kreirati uporabnika. Tako produkt FreeIPA predstavlja določeno prednost pred produktoma FreeRADIUS in OpenLDAP, saj poleg tega vključuje tudi številne storitve znotraj domene FreeIPA, ki jih lahko uporabljajo člani domene FreeIPA. Prav tako produkt omogoča varen proces avtentikacije, saj v ta namen uporablja protokol KERBEROS, ki temelji na vstopnicah. Namreč tu se geslo ne pošlje po medmrežju, kar zagotavlja še večjo varnost.

Avtentikacija s pomočjo protokola RADIUS, katero smo zajeli v okviru produkta FreeRADIUS, je prisotna v brezžičnem omrežju EDUROAM, saj tu nastopi potreba po protokolu tipa AAA. Pogosto nastopi RADIUS avtentikacija s pomočjo zunanjega vira, kot je to prav tako v primeru brezžičnega omrežja EDUROAM. Kot zunanji vir tu nastopa strežnik LDAP, kjer ima strežnik RADIUS shranjena gesla uporabnikov in dostopa do strežnika LDAP, v okviru procesa avtentikacije. Na drugi strani avtentikacija s pomočjo protokola KERBEROS, katero smo zajeli v okviru produkta FreeIPA, nastopa v operacijskih sistemih Windows, kar nam pove, da je protokol KERBEROS zelo priljubljen.

Sama primerjava avtentikacij s pomočjo izbranih produktov nam je omogočila lažjo predstavo o varnosti ter o lažji izbiri avtentikacijskega strežnika. S procesom avtentikacije se zelo pogosto srečamo na področju računalništva, saj moramo na številnih mestih izkazati svojo identiteto ter si pri tem ne želimo, da bi se nekdo izdajal za nas ob morebitnem prestrezanem geslu. Da se izognemo tej nevšečnosti, je ključnega pomena izbran varen avtentikacijski protokol, s katerim lahko na varen način opravimo proces avtentikacije, ki ga zahteva sistem oziroma določena spletna aplikacija.

Pri RADIUS avtentikaciji Linux uporabnikov bi lahko omogočili varno povezavo med našim odjemalcem in virtualnim strežnikom, saj je modul `Pam_Radius` ne omogoča. V tem primeru bi lahko uporabili virtualno zasebno omrežje (angl. *virtual private network*) med našim virtualnim strežnikom in odjemalcem, vendar ta rešitev ni bila v sklopu zahtev, saj se je iskala rešitev, ki bi na enostavnejši način pripomogla k varni komunikaciji med odjemalcem in virtualnim strežnikom (npr. uporaba avtentikacijskega protokola EAP namesto PAP), vendar je sam modul `Pam_Radius` ne omogoča, kot smo ugotovili.

Literatura

[1] B. Aboba, A. Palekar, *IEEE 802.1X and RADIUS Security*, http://www.slideworld.org/ViewSlides.aspx/39645, nov. 2011.

[2] M. Butcher, *Mastering OpenLDAP Configuring, Securing, and integrating Directory Services*, Birmingham: Packt Publishing, avg. 2007, pogl. 1, 2, 3, 6.

[3] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, *RFC 3588: Diameter base protocol*, http://www.ietf.org/rfc/rfc3588.txt, sept. 2003.

[4] G. Carter, *LDAP System Administration*, Sebastopol: O' Reilly Media, mar. 2003, pogl. 1, 2, 3.

[5] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, *RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, http://www.ietf.org/rfc/rfc5280.txt, maj 2008.

[6] T. Dierks, E. Rescorla, *RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2*, http://tools.ietf.org/html/rfc5246, avg. 2008.

[7] A. Freier, P. Karlton, P. Kocher, *RFC 6101: The Secure Socket Layer (SSL) Protocol Version 3.0*, http://tools.ietf.org/html/rfc6101, avg. 2011.

[8] J. Garman, *Kerberos: The Definitive Guide*, Sebastopol: O'Reilly & Associates, avg. 2003, pogl. 1, 2, 3, 6.

[9] J. Hassell, *RADIUS*, Sebastopol: O'Reilly Media, okt. 2002, pogl. 1, 2, 3, 4, 5.

[10] A. Melnikov, K. Zeilenga, *RFC 4422: Simple Authentication and Security Layer (SASL)*, http://www.ietf.org/rfc/rfc4422.txt, jun. 2006.

[11] C. Neuman, T. Yu, S. Hartman, K. Raeburn, *RFC 4120: The Kerberos Network Authentication Service (V5)*, http://www.ietf.org/rfc/rfc4120.txt, jul. 2005.

[12] C. Rigney, W. Willats, P. Calhoun, *RFC 2869: RADIUS Extensions*, http://freeradius.org/rfc/rfc2869.html#Message-Authenticator, jun. 2000.

[13] C. Rigney, S. Willens, A. Rubens, W. Simpson, *RFC 2865: Remote Authentication Dial In User Service (RADIUS)*, http://www.ietf.org/rfc/rfc2865.txt, jun. 2000.

[14] S. Turner, L. Chen, *RFC 6151: Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms*, http://tools.ietf.org/html/rfc6151, mar. 2011.

[15] D. van der Walt, *FreeRADIUS Beginner's Guide*, Birmingham: Packt Publishing, sept. 2011, pogl. 1, 4, 6, 8, 10, 12.

[16] S. Winter, M. McCauley, S. Venaas, K. Wierenga, *RFC 6614: Transport Layer Security (TLS) Encryption for RADIUS*, http://tools.ietf.org/html/rfc6614, maj 2012.

[17] (2012) ArticSoft, ″An introduction to PKI (Public Key Infrastructure)″. Dostopno na: http://www.articsoft.com/public_key_infrastructure.htm.

[18] (2006) Cisco, ″How Does RADIUS Work?″. Dostopno na: http://www.cisco.com/en/US/tech/tk59/technologies_tech_note09186a00800945cc.shtml.
[19] (2011) Difference Between, ″Diferrence Between LDAP and AD″. Dostopno na: http://www.differencebetween.com/difference-between-ldap-and-vs-ad/.
[20] (2009) Difference Between, ″Difference Betwen LDAP and Active Directory″. Dostopno na: http://www.differencebetween.net/technology/difference-between-ldap-and-acitve-directory/.
[21] (2012) Fedora Documentation, ″FreeIPA: Identity/Policy Management″. Dostopno na: https://docs.fedoraproject.org/en-US/Fedora/17/html/FreeIPA_Guide/index.html.
[22] (2012) FreeRADIUS The world's most popular RADIUS server, ″The FreeRADIUS Project″. Dostopno na: http://freeradius.org/.
[23] (2010) Gemmi Media Productions, ″What is the difference between LDAP and Active Directory?″. Dostopno na: http://www.geminimediaproductions.com/activedir_ldap.htm.
[24] (2006) IBM developerWorks, ″Introduction to Diameter″. Dostopno na: http://www.ibm.com/developerworks/library/wi-diameter/.
[25] (2008) Kioskea, ″LDAP protocol″. Dostopno na: http://en.kioskea.net/contents/internet/ldap.php3.
[26] (2011) Learn Telecom, ″Diameter Protocol – an Overview″. Dostopno na: http://learntelecom.com/telephony/lte/diameter-protocol-an-overview.
[27] (2012) Microsoft Developer Network, ″So What Is Active Directory?″. Dostopno na: http://msdn.microsoft.com/en-us/library/windows/desktop/aa746492(v=vs.85).aspx.
[28] (2002) Microsoft TechNet, ″RADIUS Protocol Security and Best Practices″. Dostopno na: http://technet.microsoft.com/en-us/library/bb742489.aspx.
[29] (2009) Oracle Blogs, ″Introduction to Diameter Protocol″. Dostopno na: https://blogs.oracle.com/naman/entry/introduction_to_diameter_protocol.
[30] (2012) openSUSE 12.2 Security Guide, ″Authentication with PAM″. Dostopno na: http://doc.opensuse.org/documentation/html/openSUSE/opensuse-security/cha.pam.html.
[31] (2012) Penn Computing, ″Kerberos Tickets and How They Work″. Dostopno na: http://www.upenn.edu/computing/pennkey/use/tktmgr.html.
[32] (2006) Red Hat Enterprise Linux Deployment Guide, ″Pluggable Authentication Modules (PAM)″. Dostopno na: http://www.centos.org/docs/5/html/Deployment_Guide-en-US/ch-pam.html.
[33] (2007) SearchNetworking, ″PPP (Point-to-Point Protocol)″. Dostopno na: http://searchnetworking.techtarget.com/definition/PPP.
[34] (2006) SearchSecurity, ″PKI (public key infrastructure)″. Dostopno na: http://searchsecurity.techtarget.com/definition/PKI.
[35] (2006) SearchSecurity, ″encryption″. Dostopno na: http://searchsecurity.techtarget.com/definition/encryption.

[36] (2008) SearchSecurity, ″asymmetric cryptography (public-key cryptography)″. Dostopno na: http://searchsecurity.techtarget.com/definition/asymmetric-cryptography.
[37] (2009) Stack Overflow, ″Active Directory vs OpenLDAP″. Dostopno na: http://stackoverflow.com/questions/997424/active-directory-vs-openldap.
[38] (2012) Wikipedia, ″Caesar cipher″. Dostopno na: http://en.wikipedia.org/wiki/Caesar_cipher.
[39] (2012) Wikipedia, ″Viegenere cipher″. Dostopno na: http://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher.
[40] (2012) Wikipedia, ″Symmetric-key algorithm″. Dostopno na: http://en.wikipedia.org/wiki/Symmetric-key_algorithm.
[41] (2012) Wikipedia, ″Data Encryption Standard″. Dostopno na: http://en.wikipedia.org/wiki/Data_Encryption_Standard.
[42] (2012) Wikipedia, ″Advanced Encryption Standard″. Dostopno na: http://en.wikipedia.org/wiki/Advanced_Encryption_Standard.
[43] (2012) Wikipedia, ″OpenLDAP″. Dostopno na: http://en.wikipedia.org/wiki/OpenLDAP.
[44] (2012) Wikipedia, ″Transport Layer Security″. Dostopno na: http://en.wikipedia.org/wiki/Secure_Sockets_Layer.
[45] (2012) Wikipedia, ″Public-key cryptography″. Dostopno na: http://en.wikipedia.org/wiki/Public-key_cryptography.
[46] (2012) Wikipedia, ″Cryptographic hash function″. Dostopno na: http://en.wikipedia.org/wiki/Cryptographic_hash_function.
[47] (2012) Wikipedia, ″Message authentication code″. Dostopno na: http://en.wikipedia.org/wiki/Message_authentication_code.
[48] (2012) Wikipedia, ″Hash-based message authentication code″. Dostopno na: http://en.wikipedia.org/wiki/Hash-based_message_authentication_code.
[49] (2012) Wikipedia, ″MD5″. Dostopno na: http://en.wikipedia.org/wiki/MD5.
[50] (2012) Wikipedia, ″SHA-1″. Dostopno na: http://en.wikipedia.org/wiki/SHA-1.
[51] (2012) Wikipedia, ″SHA-2″. Dostopno na: http://en.wikipedia.org/wiki/SHA-2.
[52] (2012) Wikipedia, ″RADIUS″. Dostopno na: http://en.wikipedia.org/wiki/RADIUS.
[53] (2012) Wikipedia, ″Point-to-point protocol″. Dostopno na: http://en.wikipedia.org/wiki/Point-to-point_protocol
[54] (2012) Wikipedia, ″Public-key infrastructure″. Dostopno na: http://en.wikipedia.org/wiki/Public-key_infrastructure
[55] (2005) Windows Server, ″MS-CHAP version 2″. Dostopno na: http://technet.microsoft.com/en-us/library/cc739678(v=ws.10).
[56] (2005) Windows Server, ″Shared secrets″. Dostopno na: http://technet.microsoft.com/en-us/library/cc740124(v=ws.10).aspx.

[57] (2012) ZYTRAX, ″Chapter 3. LDAP Schemas, ObjectClasses and Attributes″. Dostopno na: http://www.zytrax.com/books/ldap/ch3/.