Tampereen teknillinen yliopisto
Tietoliikennetekniikan laitos

83962, Tietoliikennetekniikan projektityö
kevät 2005

Tut1x-projektin laajennus

Loppuraportti 20.5.2005

Projektin valvoja ja ohjaaja:
Jukka Koskinen

Projektiryhmä:
Jouni Honkala, jouni.honkala@tut.fi
Antti Laiti, antti.laiti@tut.fi
Petri Niemi, petri.niemi@tut.fi

Sisällysluettelo

Sanasto

802.1x	RFC 3580, IEEE:n standardi joka määrittelee autentikointitavan portteihin pohjautuvaan pääsynvalvontaan. Toimii OSI-mallin toisella kerroksella eli linkkitasolla ja mahdollistaa valvonnan IEEE 802 -verkoissa kuten Ethernet (802.3) ja WLAN (802.11.) Pääsynvalvonta voidaan WLAN-tekniikkaa käytettäessä toteuttaa käyttäjänimi-salasana -pareilla tai digitaalisella varmenteella. [1]
doc++	Dokumentointijärjestelmä C-, C++-, IDL- ja Java-ohjelmointikielille. Muodostaa koodin kommenteista Tex- ja HTML-dokumentit. [3]
draft	Työryhmän tekemä tekninen raportti, joka kertoo meneillään olevasta työstä. Toimii dokumentin esiasteena.
GTC	Generic Token Card; Käyttäjäkohtaisia merkkikortteja käyttävä autentikointimenetelmä. Perustu varmenne/merkkikorttipohjaiseen varmentamiseen (RFC 3748.)[6]
IEEE	The Institute of Electrical and Electronics Engineers, Inc. Voittoa tavoittelematon yhdistys, jolla on yli 330 000 jäsentä 150 maassa. Yhdistys tekee teknisiä julkaisuja, järjestää konferensseja ja pyrkii luomaan standardeja.
IETF	Internet Engineering Task Force; Työryhmiin jakautunut, avoin ja kansainvälinen yhteisö johon kuuluu mm. verkkosuunnittelijoita sekä operaattoreita ja tutkijoita, jotka ovat kiinnostuneita Internetin kehityksestä.
OTP	One-time-password; Kertakäyttösalasanoihin perustuva autentikointimenetelmä (RFC 2289.)[5]
PEAP	Protected EAP; Suojattu autentikointimenetelmä langattomaan verkkoon. Ei vaadi varmenteita. Tekijöinä Microsoft, Cisco ja RSA Security. IETF:n laajennus RFC 2284:ään (EAP.)[4]
RADIUS	Remote Authentication Dial-In User Service; Standardi, jonka avulla voidaan lähettää tietoja keskitetylle tietokantapalvelimelle (RADIUS-palvelin), jossa tiedot varmennetaan (RFC2866.)[7]
tut1x	Tampereen teknillisen yliopiston avoimen lähdekoodin projekti, jonka tarkoituksena on tehdä helppokäyttöinen ja helposti konfiguroitava 802.1x-asiakasohjelma Linux ympäristöön. [2]

1 Johdanto

Nykyään yhä useampi tietoliikenneratkaisu on langaton. Langattomat verkot tuovat mukanaan liikkuvuuden ja joustavuuden lisäksi muutamia ongelmia. Kupariverkkoon ei pääse kuka tahansa kiinni, sillä se vaatii fyysistä yhteyttä ja verkon pääsynvalvontana voi toimia esimerkiksi rakennuksen kulunvalvonta. Langaton verkko sen sijaan kuuluu helposti rakennuksen ulkopuolellekin jolloin siihen saattaa päästä kiinni ei-toivottujakin laitteita.

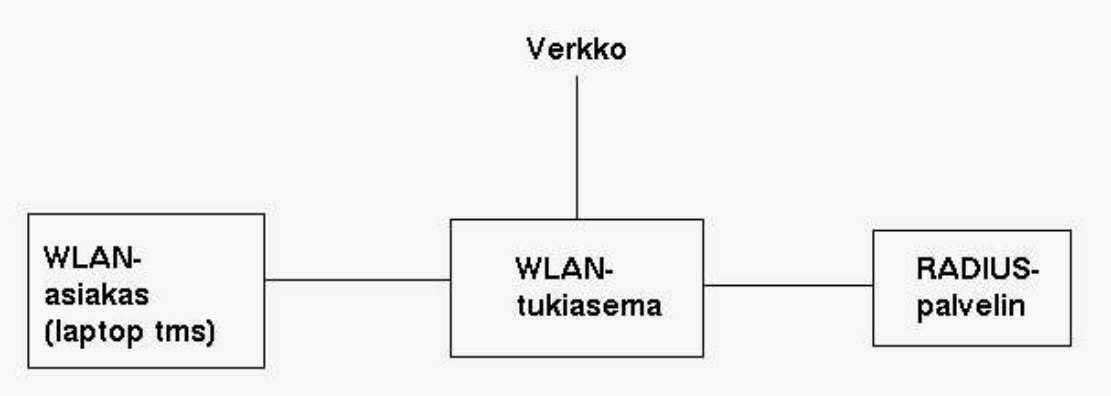
Tämän vuoksi langattomissa verkoissa on hyvä olla pääsynvalvonta. Pääsynvalvonta voi perustua esimerkiksi verkkolaitteiden MAC-osoitteisiin, mutta sellaisen toteuttaminen on hankalaa isoissa verkoissa. Lisäksi MAC-osoite voidaan helposti väärentää. Usein parempi tapa pääsynvalvontaan onkin jokin salasanaan tai muuhun varmenteeseen perustuva autentikointimenetelmä. Yksi käytetyimmistä menetelmistä on 802.1x, joka on monipuolinen ja toimii monella eri alustalla. Sen ongelmana on kuitenkin se, että joillakin alustoilla sille on olemassa vain yksinkertaisia ja moneen käyttötapaukseen riittämättömiä toteutuksia. Joillakin alustoilla sille ei ole olemassa minkäänlaista toteutusta.

Linux-käyttöjärjestelmälle on olemassa xsupplicant-niminen 802.1x-asiakasohjelma. Se on toteutukseltaan hyvin yksinkertainen, eikä tarjoa paljoa vaihtoehtoja autentikointitapoja ajatellen. Lisäksi sen käyttöliittymä on usealle käyttäjälle liian vaikea. Tämän vuoksi Tampereen teknillisen yliopiston tietoliikennetekniikan laitos aloitti projektin nimeltä tut1x, jonka tarkoituksena on saada aikaiseksi asiakasohjelma, jossa nämä epäkohdat on korjattu.

2 Taustaa

Tut1x on Tampereen teknillisen yliopiston tietoliikennetekniikan laitoksen avoimen lähdekoodin projekti. Sen tarkoituksena on tarjota helposti konfiguroitava ja helppokäyttöinen 802.1x-asiakas Linux-ympäristöön. 802.1x on IEEE:n standardi, joka määrittelee portteihin pohjautuvan pääsynvalvontatavan langattomissa verkoissa. Tut1x-projektin laajennusprojektin tarkoituksena on edistää tut1x-projektin etenemistä tuottamalla sen autentikointiosuuteen lisää koodia hyvän dokumentaation kera.

2.1 802.1x:n toiminta



Kuva 1. esimerkki 802.1x:n toimintaympäristöstä.

Kuvassa 1 on 802.1x:n toimintaympäristö. WLAN-verkkoon haluava käyttäjä käynnistää 802.1x-asiakasohjelmansa. Se katsoo asetustiedoistaan, mitä

autentikointitapaa sen halutaan käyttävän ja lähettää WLAN-tukiasemalle pyynnön kyseisen autentikointitavan sääntöjä noudattaen. WLAN-tukiasema puolestaan välittää pyynnön eteenpäin RADIUS-palvelimelle, joka keskustelee WLAN-asiakkaan kanssa pakettivälitteisesti WLAN-tukiaseman kautta halutun autentikointitavan mukaisesti. Jos autentikointi onnistuu, RADIUS-palvelin ilmoittaa onnistumisesta WLAN-tukiasemalle joka päästää WLAN-asiakkaan verkkoon. [1]

2.2 Autentikointitavat

Asiakkaan ja palvelimen välillä käytettäviä autentikointimenetelmiä on useita. Tut1x:n tarkoituksena on tarjota toiminnallisuus mahdollisimman montaa eri menetelmää varten. Laajennusprojekti tulee juuri tässä kohtaa kuvaan; sen tarkoitus on tuottaa kolme uutta autentikointitapaa tut1x:lle c-kielellä. Nämä autentikointitavat ovat OTP, GTC ja PEAP ja ne toimivat kukin omalla tavallaan. Tarkoitus on kaikilla kuitenkin sama: asiakkaan autentikointi. 802.1x-standardin mukaisissa järjestelmissä voidaan käyttää monia eri autentikointitapoja; osa niistä on jo toteutettu tut1x:ään, osa on vielä toteuttamatta. Nämä kolme menetelmää valittiin toteutettaviksi siksi, että ne ovat yleisessä käytössä. Lisäksi OTP ja GTC ovat standardisoituja autentikointitapoja; niille löytyy oma RFC-dokumentti IETF:n arkistosta. Seuraavassa on selitetty tarkemmin autentikointitapojen toimintaa.

2.2.1 Extensible Authentication Protocol (EAP)

EAP tarjoaa verkkoon kirjautumista varten useita eri autentikointitapoja tukevan protokollakerroksen. Se sijoittuu perinteisessä kerrosmallissa linkkikerroksen, kuten Ethernetin IEEE 802.11:n tai PPP:n päälle. EAP toteuttaa itse mm. viestin perillemenon varmistamisen (vastaanottokuittaus, uudelleenlähetys, duplikaattien tunnistus), joten protokolla voi toimia tarvitsematta erillisen verkkokerroksen, esim. IP:n palveluita.

EAP:n suurin hyöty on joustavuus: protokollan avulla autentikoinnin osapuolet kykenevät mm. neuvottelemaan molemmille soveltuvasta autentikointitavasta. Itse autentikoinnin suorittavat ns. sisemmät protokollat (engl. inner protocols), joiden välinen liikenne ohjataan EAP-tason kautta. Tämän lisäksi käyttäjän autentikointipyyntö voidaan ohjata verkkoon pääsyä valvovasta verkkoelementistä ulkopuoliselle autentikointipalvelimelle. Näin ollen mahdollistuu myös ns. verkkovierailu usean organisaation hallinnoimien pääsyrajoitettujen lähiverkkojen välillä.

EAP on standardoitu viimeksi RFC3748:ssa. Itse määrittelyyn sisältyy muutamia yksinkertaisia autentikointitapoja, kuten MD5-Challenge sekä OTP ja GTC, joista lisää jäljempänä tässä raportissa. Myös lukuisia muita EAP-laajennuksia on olemassa ja uusia voidaan periaatteessa kehittää lisää tarpeen mukaan.[6]

2.2.2 Protected EAP

EAP-protokolla kehitettiin alunperin fyysistä kaapelointia käyttäviin verkkoihin. Tavallisin käyttökohde oli autentikointi puhelinverkon yli tapahtuvien PPP-yhteyksien alustuksessa. Tällöin yhteyden ns. fyysinen turvallisuus oletettiin riittäväksi – mahdollisen hyökkääjän olisi pitänyt tunkeutua puhelinverkkoon havaitakseen keskustelun sisällön. Lähiverkkostandardi IEEE 802.1x määrittelee 802-standardiperheen lähiverkoissa toimivan EAP-variantin olettaen, että käytössä on yhteisen siirtomedian sijasta ns. tähtimallin verkko, jossa koneet liittyvät verkkoon

omilla kaapeleillaan. Tällöin puolestaan mahdollisen hyökkääjän olisi saatava vapaa pääsy verkkolaitteisiin keskustelun tallentamiseksi. Koska verkkojen fyysinen turvallisuus oletettiin riittäväksi, EAP-standardi ei sisällä keinoja EAP-keskustelun suojaamiseksi kolmansilta osapuolilta. Myöhemmin, eritoten kun otetaan käyttöön avoimeen radiotiehen perustuvia lähiverkkoja, todettiin EAP-protokollan tarjoama tietoturvan taso liian heikoksi. Kolmansilla osapuolilla olisi varsin vähäisin järjestelyin mahdollisuus seurata autentikointia sekä puuttua siihen lähettämällä verkkoon väärennettyjä paketteja. Yhteyden fyysisen turvallisuuden oletus ei pitäisi enää paikkaansa. Käyttäjän identiteetti tai salainen tunniste ei enää olisi turvassa. Lisäksi havaittiin tarve kaksisuuntaiseen autentikointiin: sekä käyttäjän että verkon tulisi voida varmistua toistensa ilmaisemien identiteettien aitoudesta, jotta vältyttäisiin kirjautumasta salakuuntelua tms. varten pystytettyyn valeverkkoon.

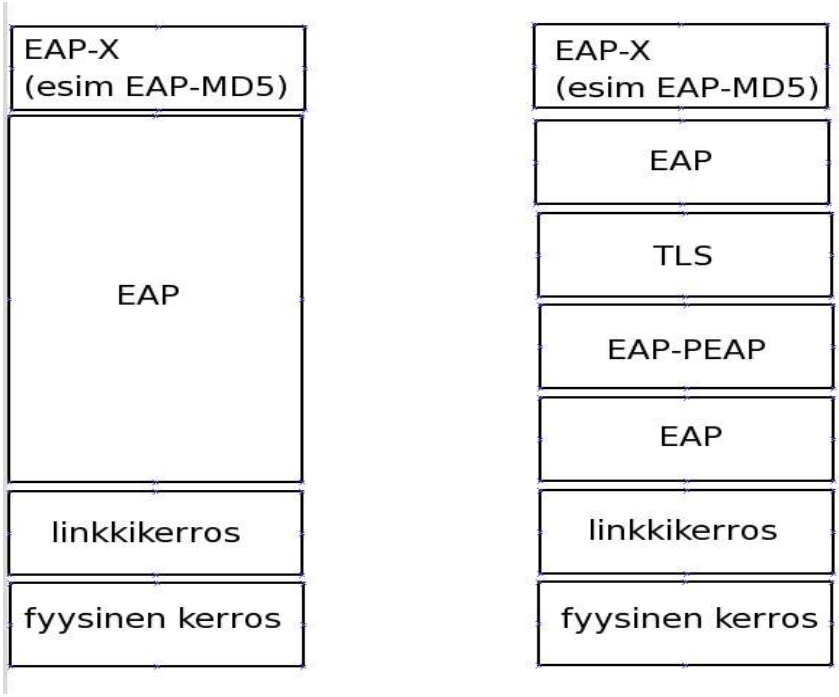
EAP:n tietoturvaongelmat ratkaistiin kehittämällä uusi autentikointimenetelmä nimeltä PEAP, joka tarjoaa palvelun autentikointiviestien salaukseen. Apuna käytetään tunnettua sekä Internet-sovelluksissa varsin usein käytettävää TLS-protokollaa. PEAP-keskustelun aluksi alustetaan salauksen toteuttava TLS-protokollataso. Varsinainen identiteettien vaihto sekä verkkoon autentikointi tapahtuu tämän jälkeen aloitettavalla ns. sisemmällä EAP-keskustelulla, jonka sanomat tunneloidaan verkon yli salattuina PEAP-sanomiin pakattuina. Sisemmän EAP-keskustelun päätteeksi tieto onnistumisesta/epäonnistumisesta välitetään sekä salattuna TLS-tunnelin sisällä että avoimena ulommaisella EAP-protokollatasolla. Tämä ns. suojattu päättäminen suojaa verkkoa sellaisilta palvelunestohyökkäyksiltä, jotka perustuvat väärennettyjen epäonnistumisviestien levittämiseen.

PEAP:n suurimpia hyötyjä ovat identiteettien suojaus kolmansilta osapuolilta sekä parempi suoja autentikointia häiritseviä hyökkääjiä vastaan. Lisäksi PEAP määrittelee katkenneen verkkoyhteyden nopeutetun uudelleenmuodostamisen, mikä on varsin toivottu uusi toiminnallisuus etenkin ajoittain toiminnaltaan epävarmoissa langattomissa lähiverkoissa.

PEAP-keskustelu jakautuu aina kahteen vaiheeseen. Aluksi osapuolet neuvottelevat keskenään TLS-tunnelin alustamiseksi. Tässä vaiheessa PEAP-sanomat sisältävät TLS-standardin määrittelemiä kättelysanomia. Kättelyvaiheen päätyttyä onnistuneesti aletaan kuljettaa sisemmän EAP-autentikointikeskustelun sanomia TLS-tunnelin lävitse salattuina verkon yli. Onnistuneen yhden tai useamman peräkkäisen EAP-keskustelun jälkeen suoritetaan ns. suojattu päättäminen, jossa autentikoinnin onnistuminen kuitataan sekä salattuna TLS-tunnelin sisällä että avoimena kaikkein uloimmalla EAP-kerroksella onnistumisen aitouden varmistamiseksi.

Tunnelin sisällä kuljetettava data sisältää kokonaisia EAP-protokollan paketteja. Mikäli nämä paketit ovat EAP-alityyppiä EAP-TLV, ne sisältävät PEAP-protokollan kontrolliviestejä. Näitä ovat mm. Crypto-Binding TLV, jolla osapuolet varmistuvat PEAP:n sekä toisaalta sisempien EAP-protokollien kryptografisen avaintenvaihdon onnistumisesta. Muut EAP-alityypit indikoivat jonkin sisemmän EAP-keskustelun käynnissä oloa.

Virhetilanteista ilmoittaminen sekä niistä toipuminen jätetään PEAP-protokollassa TLS:n tai sisemmän EAP:n tehtäväksi. Tämä osaltaan edistää verkon turvallisuutta, sillä esim. virhetilanteiden väärentämisen vaikeuttaminen parantaa turvaa palvelunestohyökkäyksien varalta. Virhesanomien välitys tapahtuu joko TLS-protokollan sisäisillä Alert-viesteillä tai EAP:n Notification-sanomilla. Jonkin virheen keskeyttäessä autentikoinnin suoritetaan edelleen suojattu päättäminen, jossa osapuolet vaihtavat keskenään EAP Failure -viestejä sekä salattuna että salaamattomana.[4]



Kuva 2. EAP:n ja PEAP:n protokollapinot.

Kuvassa 2 on EAP:n ja PEAP:n protokollapinot. PEAP-pinossa on TLS- ja EAP-PEAP-kerrokset liikenteen salaamiseksi sekä toinen EAP-kerros yhteensopivuuden takaamiseksi. Näitä kerroksia ei alkuperäisessä EAP:ssa ole.

2.2.3 Generic Token Card

EAP-GTC (Generic Token Card) –protokolla soveltuu käytettäväksi kun tarvitaan käyttäjän vahvaa tunnistusta erillisen tunnistekortin (mm. SecurId) avulla. Protokollan toiminta perustuu yksinkertaiseen haaste-vaste –menetelmään. Autentikointipalvelin lähettää verkkoon pyrkivälle käyttäjälle selkokielisen kehotetekstin, jolla pyydetään käyttäjää vastaamaan tunnistelaitteelta saamansa kertakäyttöisen tunnisteluvun tai -sanan. Yleensä autentikointiin tunnistekortilla liittyy jokin jaettu salaisuus, joka on sovittu käyttäjän ja autentikointipalvelimen välillä. Tätä salaisuutta käytetään muodostettaessa palvelimelle lähetettävää tunnistelukua. Käyttäjän identiteetti (käyttäjätunnus yms) saadaan tiedoksi EAP-protokollan kättelyvaiheesta, joten sitä ei enää kysytä uudelleen GTC-protokollassa.

Esimerkiksi SecurID:tä käytettäessä käyttäjä on ensimmäisellä käyttökerralla ilmoittanut valitsemansa PIN-koodin autentikoivalle palvelimelle. Myöhemmillä kerroilla verkkoon kirjauduttaessa käyttäjä näppäilee PIN-koodinsa SecurID-korttiin ja saa vastineeksi tunnisteluvun, jonka avulla verkkoon pääsy onnistuu. Tunnisteluku vanhenee noin minuutin kuluessa, joten seuraavallakin kirjautumiskerralla koodi on syötettävä uudelleen voimassaolevan tunnisteluvun laskemiseksi. Tämä tekee SecurID-kortin käytännössä hyödyttömäksi kolmannen osapuolen käytössä ellei PIN-koodi ole tiedossa (olettaen, että tunnisteen muodostavissa algoritmeissa ei ole tiedossa olevia haavoittuvuuksia).[6]

2.2.4 One-time password

OTP-autentikointimenetelmä (RFC 2289) kehitettiin Bellcore-nimisen yhtiön toteuttaman S/KEY One-Time Password –menetelmän (RFC 1760) pohjalta. Tässä

autentikointimenetelmässä perusajatuksena on estää niin sanotut ”Replay Attack” -hyökkäykset, missä kolmas osapuoli saa salakuuntelemalla selville käyttäjän tunnuksen ja salasanan ja voi yrittää käyttää näitä tietoja hyväkseen. OTP:ssä käyttäjäkohtaista pysyvää salasanaa ei missään vaiheessa lähetetä verkon yli, joten tämä menetelmä ei ole haavoittuvainen edellisen tyyppisiä hyökkäyksiä vastaan. Tätä pysyvää salasanaa käytetään kertakäyttösalasanan luomiseen myöhemmin kuvattavalla tavalla. Lisäturvaa tuo myös se ominaisuus, että mitään käyttäjäkohtaista salaista tietoa, kuten edellä mainittua pysyvää salasanaa, ei tarvitse tallentaa selväkielisenä missään autentikoinnin vaiheessa esimerkiksi autentikointipalvelimelle.

Kun autentikointipalvelimelle (esimerkiksi järjestelmänvalvojan toimesta) luodaan uusi käyttäjä OTP:tä varten, syötetään järjestelmään kyseisen käyttäjän pysyvä salasana. Salasanaa ei kuitenkaan tallenneta sellaisenaan, vaan se salataan käyttämällä sopivaa kryptografista tiivistefunktiota. OTP:ssä määritellyt tiivistefunktiot ovat MD4, MD5 jaSHA1, joista ainoastaan MD5 on pakollinen. Yleisesti ottaen tiivistefunktio takaa, että samasta syötteestä eri kerroilla laskettu arvo tuottaa aina saman tuloksen, ja että tästä tuloksesta on lähes mahdotonta päätellä alkuperäistä syötettä.

Järjestelmään syötettyyn salasanaan yhdistetään jokin palvelimen valitsema siemenmerkkijono (seed) ja tämä tulos ajetaan valitun tiivistefunktion läpi n kertaa, missä n on vapaasti valittavissa oleva kokonaisluku (tyypillisesti noin sata.) Tiivistefunktion syötteeksi kierroksella k annetaan siis aina kierroksen k-1 tulos. Palvelimen päässä talletetaan käyttäjätunnus, luku n, siemenmerkkijono ja edellä laskettu tiivistearvo.

Kun käyttäjä nyt ottaa yhteyttä autentikointipalvelimeen käyttäen OTP-menetelmää, lähettää palvelin käyttäjälle haasteen. Haaste sisältää käytetyn tiivistefunktion tunnisteen, luvun n-1 ja käytetyn siemenmerkkijonon. Haaste voi olla esimerkiksi merkkijono ”otp-md5 99 someseed”, jos palvelimen päässä käytetty tiivistefunktio oli MD5, funktio ajettiin uutta käyttäjää luotaessa palvelimella sata kertaa ja siemenmerkkijono oli ”someseed”.

Käyttäjä laskee palvelimelle lähetettävän kertakäyttösalasanan (one-time password) yhdistämällä oman, pysyvän salasanansa haasteessa annettuun siemenmerkkijonoon, ja ajamalla näin syntyvän merkkijonon haasteessa mainitun tiivistefunktion läpi annetun määrän kertoja. Tässä esimerkissä siis merkkijono ajettaisiin MD5 funktion läpi 99 kertaa. Palvelin osaa päätellä käyttäjän autentikoinnin onnistumisen tai epäonnistumisen, kun se kertakäyttösalasanan saadessaan ajaa sen yhden kerran saman tiivistefunktion läpi ja vertaa tulosta aiemmin talletettuun arvoon. Jos tulokset ovat samat, autentikointi onnistui. Tämän jälkeen palvelin tallentaa käyttäjältä saamansa uuden kertakäyttösalasanan aiemmin talletetun arvon tilalle ja pienentää lukua n yhdellä. OTP-spesifikaatio ei ota kantaa siihen, miten järjestelmään syötetään uudet alkuarvot laskurin n mennessä nollaan.[5]

3 Projektinhallinta

3.1 Organisaatio

Projektiryhmään kuuluu kolme tekniikan ylioppilasta; Antti Laiti toimi projektipäällikkönä ja hänen vastuullaan oli kehitysympäristön pystytys ja ylläpito sekä projektiin liittyvien raporttien sekä esitysmateriaalin laadinta. Lisäksi hän osallistui moduulien testaamiseen. Petri Niemi ja Jouni Honkala toimivat projektin ohjelmoijina. He hoitivat ohjelmointityön sekä dokumentoinnin. Lisäksi he testasivat ohjelmistoja ja osallistuivat projektin raporttien laadintaan sekä esitysten pitämiseen.

Projektin asiakkaana on Tietoliikennetekniikan laitoksen työntekijä sekä Tut1x-projektin vetäjä Ilkka Karvinen. Projektin valvojana on Tietoliikennetekniikan laitoksen lehtori Jukka Koskinen.

3.2 Yhteydenpito

Projektiryhmän jäsenet pitivät toisiinsa yhteyttä sähköpostin, pikaviestinohjelmien sekä puheluiden avulla. Lisäksi sovittiin pidettäväksi viikoittainen palaveri keskiviikkoisin kello 16:00. Palavereita jouduttiin perumaan usein, sillä ryhmän jäsenillä oli aika usein muuta menoa tuohon aikaan. Kommunikointi sujui kuitenkin ongelmitta muiden medioiden kautta, mikäli palaveri jouduttiin perumaan jostain syystä.

Asiakkaaseen oltiin yhteydessä koko projektin ajan sähköpostin avulla. Lisäksi projektiryhmän jäsenet kävivät usein hänen työhuoneessaan keskustelemassa projektiin liittyvistä asioista ja ongelmista. Kysyttävää ja neuvoteltavaa oli suhteellisen paljon, joten yhteydenpito oli tiivistä.

3.3 Riskit

Projektin alussa arvioitiin muutamia siihen liittyviä riskejä. Näistä oikeastaan kaikki toteutuivat jossain määrin ja niiden lisäksi ilmaantui muitakin projektin etenemistä uhkaavia tekijöitä. Näihin kuitenkin vastattiin, ja projekti saatiin vietyä loppuun, vaikkakin myöhässä aikataulustaan.

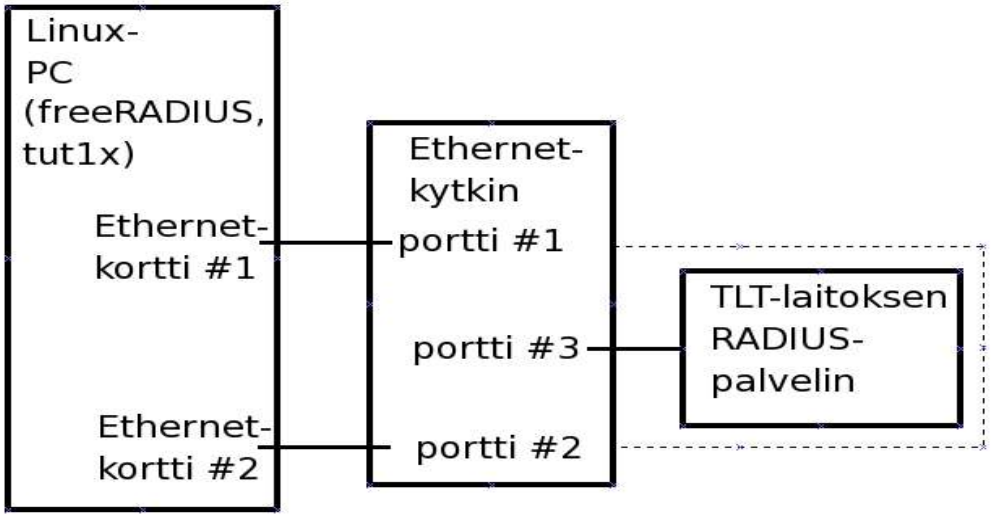
Alussa arvioitu henkilöstöriski toteutui, sillä projektiryhmän jäsenillä ei ollut työkiireidensä vuoksi aina tarpeeksi aikaa projektille. Laitteistoriski toteutui myös, sillä testausvaiheessa tapahtui se mitä laitteiston puolesta eniten pelättiin; laitteisto meni jumiin, eikä siihen saatu enää yhteyttä verkon yli millään keinolla. Maanantai-aamuna laitteisto saatettiin jälleen toimintaan asiakkaan avustuksella. Arvokasta testausaikaa meni kuitenkin jonkin verran hukkaan. Suurin toteutunut riski oli kuitenkin sellainen, jota ei aluksi huomioitu ollenkaan. Testausympäristössä käytetyt RADIUS-palvelinohjelmistot eivät suostuneet aluksi toimimaan kunnolla ja sen korjaamiseen meni huomattavan paljon aikaa. Tämä kuuluu silti oikeastaan osittain projektin alussa esiteltyyn arviointivirheet-kategoriaan, sillä tällaiseen ajankuluun ei osattu varautua. Osittain se kuuluu myös henkilöstöriskit-kategoriaan, sillä osasyyllinen palvelinohjelmiston toimimattomuuteen oli projektiryhmän jäsenten osaamisen puute.

Projekti viivästyi aikataulustaan noin kuukauden. Pääasiallisena viivästyksen syynä oli testausympäristöjen toimimattomuus, sekä muutama vaikeasti huomattavissa ollut ohjelmointivirhe ryhmän tekemässä koodissa. Lisäksi osasyyllisenä viivästykseen olivat projektiryhmän jäsenten sekä asiakkaan työkiireet ja niistä johtuva ajanpuute Projektin ajankäyttö pysyi kuitenkin hyvin suunnitelman mukaisena, eikä ylimääräisiä työtunteja tarvinnut tehdä juuri ollenkaan. Projektin alussa epäiltiin, että itse ohjelmointityö tulisi olemaan suunniteltua hitaampaa, mutta se sujui hyvin ja sekä suurimmaksi osaksi aikataulujen mukaisesti. Tästä on kiittäminen Petrin ja Jounin ohjelmointitaitoja ja osaamista.

3.4 Resurssit

Projektissa käytetty testausympäristö poikkesi jonkin verran luvussa 2.1 esitetystä kuvasta. Asiakas- ja palvelinkoneena toimi tietoliikennetekniikan laitoksen tarjoama Linux-PC, jossa asiakasohjelmana oli luonnollisesti tut1x ja palvelinohjelmistona

freeRADIUS. PC:ssä oli kaksi verkkokorttia: toinen tut1x-asiakkaan liikennettä varten ja toinen RADIUS-palvelinohjelmistoa varten. WLAN-tukiaseman virkaa toimitti 802.1x:ää tukeva ethernet-kytkin ja ulkoisena verkkona oli Internet.



Kuva 3. Projektissa käytetty laitteisto.

Kuva 3 esittää projektissa käytettyä laitteistoa. Katkoviivalla neliöity RADIUS-palvelin otettiin käyttöön projektin testausvaiheessa freeRADIUS-ohjelman ollessa projektin tarkoituksiin riittämätön. Linux-PC toimi tällöin vain 802.1x-asiakkaana.

Projektissa käytettiin seuraavia ohjelmistoja:

- radius-palvelinohjelmisto: freeRADIUS
- c-kääntäjä: GNU project C and C++-compiler
- kääntäjän apuväline: GNU flex
- verkko-analysaattoriohjelmat: Ethereal ja Tethereal
- dokumentoinnin teko koodin pohjalta: Doc++
- raporttien laadinta: Openoffice.org
- laitteiston etäkäyttö: openSSH, PuTTY, Secure Shell Client, MiniCom

Kaikki ohjelmistot ovat vapaita, kuka tahansa voi ladata ne Internetistä ja ottaa omaan käyttöönsä. Niiden käytöstä ei siis aiheutunut minkäänlaisia kustannuksia. Lisäksi projektissa käytettiin projektiryhmän omia tietokoneita ja ohjelmistoja Linux-PC:n etäkäyttöä, koodin tuottamista sekä projektin esityksien pitämistä varten.

4 Projektin vaiheet

4.1 Alku

Projektin alkuajankohdaksi voidaan katsoa 4.2., jolloin pidettiin aloituspalaveri asiakkaan, Ilkka Karvisen, sekä toisen Tietoliikennetekniikan laitoksen edustajan, professori Jarmo Harjun kanssa. Palaverissa luvattiin projektille resursseiksi tietoliikennetekniikan laitoksen puolesta Linux-PC, hallittava kytkin sekä niiden käyttöön tarvittava varustus. Ohjelmointityö tultaisiin suorittamaan lähinnä projektiryhmän jäsenten omilla tietokoneilla tai Tampereen teknillisen yliopiston mikroluokkien koneilla, jotka ovat vapaasti opiskelijoiden käytössä.

Toisessa palaverissa, joka pidettiin projektiryhmän jäsenten kesken, suoritettiin tehtävien jako ja päätettiin aikatauluista. Palaverissa päätettiin lisäksi projektin päivämääristä ja luotiin pohja projektin alustavalle aikataululle. Aikataulun laadinnassa tehtiin muutama virhe. Loppuraportille ei varattu aikaa ollenkaan, vaikka se on suuri osa projektia. Lisäksi väliraportin palautusajankohta oli liian varhainen ja aikataulu ylipäänsä oli liian tiukka, ongelmatilanteiden ratkaisuun oli varattu liian vähän aikaa.

4.2 Alkuesitys ja alkuraportti

Projektin alkuraportin ensimmäinen versio saatiin valmiiksi 10.2. Tähän tarvittiin vielä jonkun verran korjauksia, joita tehtiin ajan mittaan. Lopullinen versio oli valmis samana päivänä kuin projektin alkuesitys pidettiin, eli 25.2. Tähän mennessä kehitysympäristö oli saatu laitteiston puolesta kuntoon ja ohjelmointikin oli jo hyvällä mallilla: OTP- ja GTC-moduulit tut1x:lle olivat melkein valmiit jo. GTC-moduulin kohdalla oli kuitenkin ongelma: tut1x ei tarjonnut mahdollisuutta GTC-autentikoinnille, sillä kyseinen autentikointitapa vaatii käyttäjän syötettä eikä tut1x tarjoa mitään mahdollisuutta sen toteuttamiselle, sillä se siirtää itsensä tausta-ajoon välittömästi käynnistymisensä jälkeen. Ryhmä otti Asiakkaaseen yhteyttä ja kertoi ongelmasta. Ongelma päätettiin ratkaista luomalla koodiin itseensä kokeilua varten syöte, joka voitaisiin tut1x-projektin myöhemmässä vaiheessa poistaa samalla lisäten tekstinsyöttömahdollisuus ohjelman käyttäjälle. Ongelma olisi selvinnyt aiemmin, mikäli GTC-autentikoinnin suorittava moduuli olisi suunniteltu huolella etukäteen tut1x-ohjelman rajoitukset huomioiden.

Perjantaina 11.3. pidettyyn palaveriin mennessä huomattiin, että projektille varatut resurssit eivät riitä ohjelmiston puolesta; freeRADIUS-palvelinohjelmisto ei tue OTP-autentikointitapaa ja sen PEAP-toteutuskin noudattaa liian vanhaa standardia. Projektiryhmä otti PEAP- ja OTP-ongelmien johdosta yhteyttä projektin asiakkaaseen. Asiakkaan kanssa tultiin siihen tulokseen, että projektin testausvaiheissa voidaan käyttää Tietoliikennetekniikan laitoksen omistamaa ja ylläpitämää RADIUS-palvelinta freeRADIUS-palvelinohjelmiston sijaan. Asiakkaan kanssa sovittiin tällöin myös PEAP-moduulin kanssa käytettävästä draft-versiosta. Versioksi valittiin numero kuusi, sillä sille löytyi suoraan tuki RADIUS-palvelimesta. Kyseiset ongelmat olisi vältetty tutustumalla ajoissa freeRADIUS-ohjelmiston ominaisuuksiin ja rajoituksiin.

Lisäksi ko. ohjelmistossa oli useamman päivän ajan pieni konfigurointi-ongelma, joka kuitenkin ratkesi seuraavana päivänä ja GTC-moduulia päästiin testaamaan. Tämäkin ongelma olisi vältytty lukemalla tutustumalla freeRADIUS-ohjelmaan paremmin hyvissä ajoin, sillä kyseessä oli ainoastaan ohjelman käyttäjän virhe, ei virhe itse ohjelmassa. Koska näin ei tehty, jouduttiin vasta suhteellisen myöhäisessä vaiheessa toteamaan ko. ohjelmisto riittämättömäksi projektin tarpeisiin.

4.3 Projekti etenee

Tiistaina 15.3 projektiryhmä toimitti väliraportin projektin valvojalle. Alunperin väliraportin toimittaminen oli suunniteltu päivämäärälle 7.3, mutta alkuesityksen yhteydessä sitä päätettiin siirtää viikolla. Syynä tähän oli projektin senhetkinen tila. Väliraportti sisälsi informaatiota projektin senhetkisestä tilasta; GTC ja OTP olivat valmiina testausta varten, dokumentointi oli aloitettu ja asiakkaaseen oli pidetty yhteyttä. Lisäksi raportissa oli senhetkiset työtunnit listattuna kultakin projektiryhmän jäseneltä.

OTP-moduulin testaus viivästyi entisestään, sillä Tietoliikennetekniikan

laitoksen RADIUS-palvelinkaan ei toiminut täysin odotusten mukaisesti; se antoi failure-viestin, mikäli asiakasohjelma ehdotti OTP:tä autentikointitavaksi. Testausta ei siis voitu suorittaa ollenkaan. Myös PEAP-moduulin toteuttamisen yhteydessä löydettiin ongelma: tut1x:n koodissa määritelty puskurin maksimikoko on 16 kilotavua, mutta PEAP tarvitsee toimiakseen vähintään 64 kilotavun puskurin. Ongelmasta keskusteltiin asiakkaan kanssa ja tultiin siihen tulokseen, että projektiryhmä tekee tarvittavan muutoksen tut1x-koodissa esiintyvän muuttujan TUT1X_MAX_BUFSIZ arvoon. Tämän muuttujan arvoa siis yksinkertaisesti muutettiin 16:sta 64:ään.

4.4 Projektin loppuvaiheet

Keskiviikkona 13.4 pidetyssä palaverissa todettiin projektiryhmän jäsenten kesken että projekti tulee hyvin todennäköisesti viivästymään aikataulustaan. Loppuesitys päätettiin siirtää alunperin suunnitellusta päivämäärästä 22.4. aina toukokuun puolelle saakka, päivämäärälle 13.5.

Tämän jälkeen PEAP-moduulia testattaessa ilmeni sama ongelma kuin aiemmin OTP:n kanssa: palvelin palautti failure-paketin, mikäli asiakasohjelma ehdotti PEAP:ia autentikointitavaksi. Lisäksi PEAP-moduulin kanssa oli ilmennyt toinenkin ongelma: tut1x:n tarjoama TLS-funktio ei toiminut kaikkien PEAP:ssa käytettävien pakettien kanssa. Näitä ongelmia yritettiin ratkoa projektin asiakkaan kanssa pidetyssä palaverissa tuloksetta. Asiakas kertoi kuitenkin parin päivän kuluttua (28.4) ratkaisseensa muut ongelmat, mutta TLS-ongelmaan hän ei ollut keksinyt ratkaisua. OTP-moduulia päästiin siis testaamaan ja testaus saatiin loppuun 1.5, johon mennessä OTP:tä oli testattu kaikilla eri tapauksilla, mitä RFC 2289 toi esille. OTP:n toiminta oli moitteetonta kaikissa näissä tapauksissa, joten tulokseen voitiin olla sen osalta tyytyväisiä.

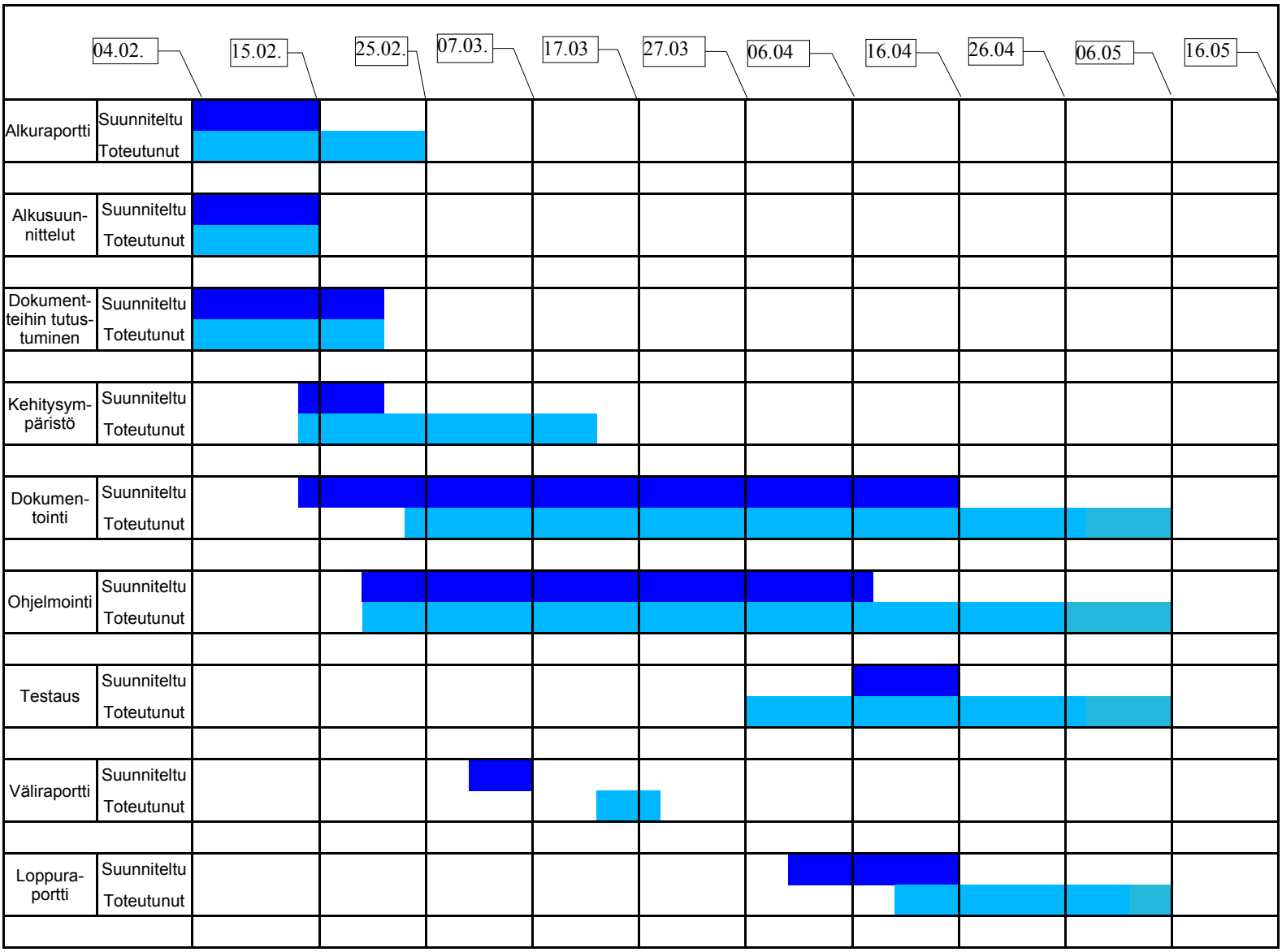
Projektin aikarajan lähestyessä löydettiin vihdoin ratkaisu PEAP:in kanssa esiintyneeseen TLS-ongelmaan: projektiryhmän koodissa oli vaikeasti havaittavissa oleva virhe, joka ratkesi vasta viikkoja kestäneen testauksen jälkeen. Ajanpuutteen vuoksi päätettiin PEAP-moduulin kanssa tyytyä siihen, mitä siihen asti oltiin saatu aikaiseksi. Moduuli saatiin testattua muutaman kerran ja se toimi joka testauskerralla hyvin, mutta sopivaa testausympäristöä pitempiin ja mutkikkaampiin testeihin ei ollut.

5 Ajankäyttö

	Antti	A Suunniteltu	Jouni	J Suunniteltu	Petri	P Suunniteltu
Alkuraportti ja alkuesitys	10	10	4	1	2	1
Alustava suunnittelu	4	8	12	6	6	6
Palaverit	7	0	7	0	7	0
Dokumentteihin tutustuminen	3	8	12	6	8	6
Kehitysympäristön rakentaminen	15	8	2	2	2	2
Väliraportti	3	6	0	1	0	1
Ohjelmointi	0	0	44	30	39	30
Dokumentointi	4	20	5	30	5	30
Testaus	12	20	20	20	19	20
Loppuraportti	35	12	3	2	2	2
Loppuesitys	10	8	1	2	1	2
yhteensä	103	100	110	100	91	100

Kaavio 1: Projektin suunniteltu ja toteutunut ajankäyttö.

Projektin ajankäyttö meni melko hyvin suunnitelman mukaisesti. Dokumentointiin meni huomattavasti vähemmän aikaa kuin suunniteltiin, sillä se toteutettiin doc++ -kommentoinnilla ohjelmoinnin ohessa. Oikeastaan siis osa ohjelmointiajasta on dokumentointiin käytettyä aikaa. Loppuraportin tekoon varattiin liian vähän aikaa. Syy tähän on todennäköisesti se, ettei sen laajuutta ymmärretty projektin alkaessa.



Kaavio 2: Projektin aikajana: suunniteltu ja toteutunut.

Kaaviosta numero 2 näkyy hyvin, kuinka projekti pysyi ensin hyvin aikataulussaan. Aikataulu alkoi kuitenkin pettää siinä vaiheessa, kun testausympäristön kanssa alkoi ilmetä ongelmia.

6 Tulos

Projektin tuloksena saatiin kolme toimivaa c-kielellä toteutettua, doc++-standardin mukaan kommentoitua moduulia tut1x-ohjelmaa varten, joista kukin tarjoaa yhden autentikointitavan. Moduulit on testattu toimiviksi, eikä niissä pitäisi olla mitään suurempia vikoja tai tietoturvaongelmia. Doc++-standardia noudattamalla saatiin myös koodin ulkoinen dokumentaatio aikaiseksi, josta selviää hyvin mitä koodi tekee ja milloin.

6.1 Ohjelmisto

Projektin päätarkoitus oli tuottaa muutama laajennus tut1x-projektin tuottamaan lähdekoodiin. Seuraavassa on esitelty kyseisten laajennusten toteutus.

6.1.1 PEAP

Tut1x:n PEAP-moduuli suunniteltiin ja toteutettiin lopulta testiympäristön valinnan sanelemana IETF:n PPPEXT Working Groupin Internet-Draftin draft-josefsson-pppext-eap-tls-eap-06.txt mukaisesti. Moduuli ottaa vastaan tut1x:n alemmilta kerroksilta palvelimelta saapuvia PEAP Request -paketteja. Protokollan luonteen mukaisesti käytännössä kaikissa keskustelun vaiheissa pakettien käsittely tapahtuu yhteistoiminnassa tut1x:ssä olemassa olevan TLS-moduulin kanssa. Tämä edelleen käyttää OpenSSL-kirjaston palveluita salaustunnelin luomiseen, hyödyntämiseen sekä salauksiin liittyvien virheiden hallintaan.

PEAP-moduuli toteuttaa paketeissa saapuvan TLS-kerroksen datan fragmentoinnin hallinnan. Saapuvaa pakettivirtaa tarkkaillaan, ja kun havaitaan kokonainen PEAP-paketti vastaanotetuksi, sen hyötykuorma annetaan TLS-kerrokselle käsiteltäväksi. PEAP-keskustelun kättelyvaiheessa, jolloin osapuolet neuvottelevat TLS-tunnelin muodostamisesta, saadaan vastineeksi TLS-kerroksen signalointidataa, joka välitetään tarvittaessa useampaan PEAP-pakettiin pilkottuna palvelimelle. Kun tunneli on jo saatu muodostettua, TLS-kerros purkaa protokollan hyötykuormana kuljetetun salatun datan PEAP-moduulissa tapahtuvaa jatkokäsittelyä varten.

EAP-TLV -alityypin hyötykuorma koostuu kokonaisesta EAP-paketista otsikoineen. Hyötykuormana kuljetetaan PEAP-protokollan kontrolliviestejä koodattuna ns. Type-Length-Value (TLV) -muotoon. TLV-viestin ensimmäiset kaksi oktettia osoittavat viestin tyypin sekä hyötydataosuuden pituuden. Kolmas ja sitä seuraavat oktetit ovat TLV-viestin sisältöä. Tällaisia TLV-tyyppejä lähdemateriaali määrittelee neljä: Result TLV (osoittaa PEAP-keskustelun onnistumisen/epäonnistumisen), NAK TLV (vastine TLV-viestiin, jota ei ymmärretty), Method-Identity TLV sekä Crypto-Binding TLV (liittyvät kryptografisten avainten vaihtoon sekä osapuolten identiteettien varmistamiseen). Saapuvat TLV-viestit puretaan ja ne käsitellään omissa funktioissaan. Paluudata lähetetään TLV-viestejä sisältävän EAP-TLV -tyypin paketin muodossa.

Havaittaessa muun EAP-alityypin paketti annetaan se ko. protokollan toteuttavalle moduulille tut1x:n handlers.c-moduulissa toteutetun tut1x_handlers_call ()-funktion kautta. Ko. funktio valitsee tarvittavan moduulin ja luovuttaa paketin sen käsiteltäväksi. Paluuarvona saadaan ko. alityypin vastauspaketti, joka välitetään PEAP-tason kautta RADIUS-palvelimelle.

PEAP lähettää vastauspakettinsa saapuvan datan tavoin tarvittaessa jaettuna useaan EAP-tason pakettiin. Lähdemateriaalina toimiva draft-standardi suosittaa yksittäisen fragmentin kooksi korkeintaan 16 kilotavua, mutta käytännön testeissä havaittiin testauksessa käytetyn RADIUS-toteutuksen lähettävän korkeintaan 1000 tavua PEAP-tason dataa yhtenä fragmenttina. EAP on rakenteeltaan Request-Response -protokolla. Muun muassa tästä syystä kunkin PEAP-fragmentin lähettämisen jälkeen jäädään odottamaan vastapuolen kuittausta, jollaisena toimii tyhjä PEAP-tason paketti. Paketteihin liittyy EAP-tasolla juokseva numerointi, joka eliminoi mahdolliset duplikaatit. Uudelleenlähetys tapahtuu joko kuittauksen jäädessä saapumatta tietyn ajan sisällä tai vastaanotettaessa kuittaus edellisen paketin saapumisesta.

Ennen verkkoon lähettämistä, PEAP:n vastauspaketti salataan TLS:llä, se pilkotaan osiin ja kukin fragmentti lähetetään EAP-pakettien sisällä sitä mukaa kun vastaanottokuittauksia saapuu. Protokollan ns. suojattu päättäminen tapahtuu molempiin suuntiin vaihdettavilla Result TLV-viesteillä, joiden Status-kenttä on arvossa 1. Tämän jälkeen TLS-tunneli voidaan purkaa. EAP-tasolla palvelin lähettää vielä salaamattoman Success -viestin onnistuneen verkkoon kirjautumisen indikoimiseksi.

PEAP-protokolla ei itsessään toteuta virheenhallintaa, vaan virheet havaitaan joko TLS:n tai EAP-tason toimesta. Toteutuksessa mahdollinen virhetilanne havaitaan palvelimen lähettämällä EAP Notification -viestillä, josta kirjoitetaan tieto tut1x:n virhelokiin. Autentikointiyritys päättyy palvelimen lähettämään EAP Failure -viestiin. Tällöin tut1x suorittaa PEAP-moduulin tietosisällön hallitun siivoamisen kutsumalla erityistä siivousfunktiota.

6.1.2 GTC

Generic Token Card -autentikointi toteutettiin RFC 3748:n GTC:tä käsittelevän luvun mukaisesti. Toteutuksessa palvelimelta vastaanotettavaan kehotteeseen vastataan EAP-viestillä, joka sisältää EAP-otsikoiden lisäksi salasanan/kertakäyttöisen tunnisteen selkokielisenä. Toteutus sallii myös useampiin peräkkäisiin palvelimelta saapuviin kehotteisiin vastaamisen.

GTC:n toiminnallisuus perustuu käyttäjän tunnistamiseen erillistä tunnistekorttia hyväksi käyttäen. EAP-GTC -protokollan tehtävänä on välittää tämä haaste-vaste -keskustelu verkon yli RADIUS-palvelimella sijaitsevan pääsynvalvontaprosessin sekä verkkoon pyrkivän käyttäjän välillä. Autentikointiin liittyy GTC-tunnisteen lisäksi aiemmin EAP-keskustelussa välitetty käyttäjätunnus. Keskustelun aikana EAP-GTC -tasolla vaihdetaan vähimmillään vain kaksi viestiä. Tämän vuoksi myös tut1x:n GTC-moduulin toteutus osoittautui projektin aikana tuotetuista moduuleista kaikkein yksinkertaisimmaksi.

Toteutuksen yhteydessä ongelmaksi muodostui tut1x-ohjelmistoalustan puutteellisuus. Ohjelmistoon ei ollut toteutettu interaktiota käyttäjän ja ohjelmiston välillä, lukuun ottamatta kiinteitä asetustiedostoja, jotka luetaan ainoastaan ohjelmaa käynnistettäessä. Kuitenkin on mahdollista, että EAP-GTC -autentikointitapaa käytettäessä käyttäjä tarvitsisi RADIUS-palvelimelta jonkin haasteluvun muodostaakseen autentikointiin vaadittavan tunnisteen. Asiakkaan kanssa sovittiin, että toteutukseksi riittää jonkin koko ohjelman suorituksen ajan kiinteänä säilyvän tunnisteen käyttö, koska interaktion toteutus ylittäisi huomattavasti projektin suunnitellun laajuuden.

EAP-GTC:n interaktiiviseen käyttöön on kuitenkin varauduttu moduulin suunnittelussa. Tällöin olisi määrä välittää palvelimen kehote mm. asynkronisena sanomana käyttöliittymäprosessille, jonka vastaavalla tavalla tarjoama vaste välitettäisiin RADIUS-palvelimelle EAP-GTC-vastauksena. Asiakkaan edustajan kertoman mukaan käyttöliittymä olisi ensi vaiheessa tarkoitus toteuttaa tut1x:n yhteyteen toteutetun konfiguraationhallintasovelluksen välillä. Tätä toiminnallisuutta ei kuitenkaan vielä ole olemassa.

6.1.3 OTP

One-Time Password-menetelmä toteutettiin pohjautuen IETF:n dokumenttiin RFC 2289. RFC:ssä mainituista tiivistefunktioista toteutus tukee funktioita MD5 ja SHA1, vaikka vain MD5 olisi ollut pakollinen. Tuki myös optionaaliselle SHA1-funktiolle oli helppo toteuttaa, sillä sen toteutus oli valmiina saatavilla avoimena lähdekoodina

OpenSSL-kirjastossa.

OTP-moduulin toteutus on melko yksinkertainen. Tut1x välittää palvelimelta saamansa OTP-haasteen sisältävän viestin OTP-moduulille. Moduuli selvittää haasteesta laskuissa käytettävän tiivistefunktion, iteraatioiden lukumäärän, sekä palvelimen lähettämän siemenmerkkijonon. Käyttäjän salasana saadaan luettua tut1x:n konfiguraatiotiedostosta. Näistä tiedoista lasketaan kertakäyttösalasana, joka lähetetään paluuviestinä palvelimelle.

OTP-menettelyn tuloksena palvelimelle lähetettävä kertakäyttösalasana voidaan RFC:n mukaan ennen lähetystä koodata käyttäen niin sanottua ”six-word”-muotoa. Kyseisessä tapauksessa tiivistefunktion tuloksena saatu 64 bitin (8 tavun) kertakäyttösalana esitetään kuutena sanana, joiden kunkin pituus on kolme tai neljä merkkiä. Käytetyt sanat voidaan ottaa RFC:ssä määritellystä taulukosta, tai OTP-järjestelmän toteuttaja voi määritellä vastaavan taulukon itse. Menetelmän tarkoituksena on helpottaa järjestelmän käyttöä toteutuksissa, joissa salasanaa ei lähetetä suoraan palvelimelle, vaan käyttäjän on itse syötettävä erillisessä moduulissa muodostettu salasana järjestelmään. Koska tut1x-toteutuksessa salasana voidaan lähettää suoraan generoinnin jälkeen palvelimelle ilman käyttäjän apua, ei toteutus muunna salasanaa ”six-word”-muotoon.

Projektin asiakkaan kanssa sovittiin, että tässä projektissa toteutettu OTP-moduuli ei ota kantaa siihen, kuinka käyttäjän salasana alun perin tallennetaan autentikointipalvelimelle. Moduuli ei myöskään hoida salasanan uudelleenasettamista palvelimen ylläpitämän laskurin mennessä nollaan. Olemassa oleva tut1x-ympäristö ei tarjoa järkevää tapaa näiden toiminnallisuuksien toteuttamiseksi. Tämä ei ole itse OTP-toteutuksen kannalta suuri puute, sillä RFC ei ota kantaa siihen, kuinka nämä asiat tulisi hoitaa.

6.2 Tulos projektiryhmän jäsenten kannalta

Projektin jäsenet tulevat saamaan opintoviikot projektikurssista kunhan projekti on viety loppuun asti ja projekti on katsottu päättyneeksi myös projektin valvojan taholta. Tämän lisäksi he kuitenkin saivat runsaasti kokemusta ja oppeja projektityön teosta; tietoa ja kokemuksia siitä, mitä kaikkea projektityöskentelyssä tulee tehdä ja ottaa huomioon ja millaisia vaikeuksia saatetaan kohdata projektin edetessä.

Projektiryhmän ohjelmoivat jäsenet, Petri ja Jouni, saivat lisää ohjelmointikokemusta ja heidän taitonsa c-kielen osalta karttuivat. Lisäksi he oppivat tuottamaan doc++:n mukaista kommentointia ja havaitsivat mitä etua siitä on. Projektin laitteiston ylläpitäjä, Antti, sai lisää kokemusta Linux-ympäristön hallinnasta sekä ohjelmien asentamisesta että konfiguroimisesta ko. ympäristöön. Raporttien ja esitysten pääasiallinen laatija, Antti, kartutti taitojaan myös toimisto-ohjelmien osalta.

6.3 Tuloksen tarkastelua

Projekti on tut1x-projektin laajennus, joten on helppo nähdä miten projektia voitaisiin viedä eteenpäin. Tut1x tarvitsee lisää autentikointitapoja, jotta se voi tarjota tarpeeksi vaihtoehtoja autentikointiin. Eri verkoissa käytetään erilaisia autentikointitapoja, joten valikoimaa on oltava runsaasti. Lisäksi projektissa jo toteutetut autentikointimoduulit tarvitsevat mahdollisia päivityksiä. Esimerkiksi PEAP-autentikointimoduuli toteutettiin draftin numero kuusi mukaisesti vaikka siitä on uudempikin versio olemassa. Tämä tehtiin siksi, ettei sopivaa testausalustaa draftin numero seitsemän mukaisesti toteutettuun koodiin ollut saatavilla.

Projektissa luotujen moduulien testaus jäi liian suppeaksi testausympäristön

ongelmien ja ajanpuutteen vuoksi. Tämän vuoksi kaikki moduuleissa olleet ohjelmointivirheet eivät välttämättä tulleet esille. Projektia voitaisiin siis viedä eteenpäin myös testauksen osalta. Moduuleita ei testattu kuin yhdessä testiympäristössä, joten on mahdollista että ne eivät toimi odotetusti jossain toisessa ympäristössä, esimerkiksi käytettäessä eri c-kääntäjää tai eri RADIUS-palvelinta.

Lähteet

[1] IETF: 802.1X Remote Authentication Dial In User Service (RADIUS)
http://www.ietf.org/rfc/rfc3580.txt
[2] Ilkka Karvinen: tut1x-projektin kotisivu
http://atm.tut.fi/tut1x/
[3] Dracos Acostachioaie: doc++'s Home Page
http://docpp.sourceforge.net/
[4] Symbol Technologies: Protected Extensible Authentication Protocol (PEAP)
http://www.symbol.com/products/wireless/peap.html
[5] IETF: The One-Time-Password SASL Mechanism
http://www.ietf.org/rfc/rfc2289.txt
[6] IETF: Extensible Authentication Protocol (EAP)
http://www.ietf.org/rfc/rfc3748.txt
[7] IETF: RADIUS Accounting
http://www.ietf.org/rfc/rfc2866.txt
[8] IETF: Port Based Network Access Control
http://www.ieee802.org/1/pages/802.1x.html