VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

AUTENTIZACE A AUTORIZACE UŽIVATELE V POČÍTAČOVÝCH SÍTÍCH NOVÉ GENERACE

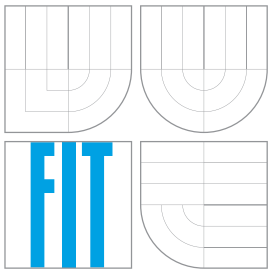
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADEK PŘIBYL

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

AUTENTIZACE A AUTORIZACE UŽIVATELE V POČÍTAČOVÝCH SÍTÍCH NOVÉ GENERACE

USER AUTHENTICATION AND AUTHORISATION FOR NEW GENERATION NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADEK PŘIBYL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR MATOUŠEK, Ph.D.

BRNO 2007

Zadání diplomové práce/5445/2006/xpriby10

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Přibyl Radek, Bc.**
Obor: Informační systémy
Téma: **Autentizace a autorizace uživatele v počítačových sítích nové generace**
Kategorie: Počítačové sítě

Pokyny:

1. Prostudujte způsoby autentizace a autorizace uživatele pro běžné služby (elektronická pošta, informační systém, přístup k síťovým zdrojům apod.).
2. Seznamte se s autentizací Kerberos a možnostmi autorizace uživatele vůči službám.
3. Navrhněte systém autentizace a autorizace pomocí důvěryhodného serveru. Zvolte vhodný způsob autentizace (veřejný klíč, certifikát apod.).
4. Implementujte modelový příklad této autentizace. Využijte globální síť PlanetLab pro implementaci.
5. Zhodnoťte bezpečnostní rizika navrženého modelu, jeho výhody i nevýhody oproti ostatním bezpečnostním modelům.

Literatura:

- Fundamentals of Network Security, Companion Guide, Cisco Press, 2004.
- G. De Laet, G. Schauwers: Network Security Fundamentals, Cisco Press, 2005.
- www.planet-lab.org
- J.Kohl, C.Neuman: The Kerberos Network Authentication Service (V5), IETF RFC 1510, 1993.

Při obhajobě semestrální části diplomového projektu je požadováno:

- body 1-3

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Matoušek Petr, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 28. února 2006
Datum odevzdání: 22. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Bc. Radek Přibyl**
Id studenta: 49439
Bytem: Holubice 52, 683 51 Holubice
Narozen: 15. 04. 1983, Brno
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Autentizace a autorizace uživatele v počítačových sítích nové generace
Vedoucí/školitel VŠKP: Matoušek Petr, Ing., Ph.D.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

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

Článek 2
Udělení licenčního oprávnění

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

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

Článek 3
Závěrečná ustanovení

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

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Tato práce popisuje způsoby autentizace a autorizace uživatele pomocí důvěryhodného serveru. Je zde rozebrán systém Kerberos, který slouží jako inspirace při návrhu vlastního autentizačního schéma. Na konkrétním příkladu aplikací jsou analyzovány programové vrstvy a jejich rozhraní zajišťující autentizaci a autorizaci uživatele. V dokumentu je navržen a popsán model autentizačního mechanismu. Tento mechanismus je implementován do komunikace mezi emailovým klientem a imap serverem.

Klíčová slova

kryptografie, autentizace, autorizace, AAA schéma, systém Kerberos, GSSAPI, SAUP protokol, Radius

Abstract

This document describes methods of user authentication and authorisation via a trusted server. There is analysis of the system Kerberos, which is used as an inspiration for desing of a new authentication scheme. There are analysed programming layers and interfaces for specific applications ensuring user authentication and authorisation. The document contains a design and detailed description of a new authentication scheme. This scheme is implemented into the communication between email client and imap server.

Keywords

cryptography, authentication, authorisation, AAA scheme, system Kerberos, GSSAPI, SAUP protocol, Radius

Citace

Radek Přibyl: Autentizace a autorizace uživatele v počítačových sítích nové generace, diplomová práce, Brno, FIT VUT v Brně, 2007

Autentizace a autorizace uživatele v počítačových sítích nové generace

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Petra Matouška

.....
Radek Přibyl
22. května 2007

Poděkování

Tímto bych chtěl poděkovat Ing. Petru Matouškovi za pomoc a rady, které mi poskytl při řešení dilpomové práce.

Obsah

Kapitola 1

Úvod

V současné době je kladen stále větší důraz na zabezpečení systémů a komunikace před útočníky. V minulosti byla proto do systémů zavedena autentizace a autorizace uživatelů. Ve většině případů se ale uživatel přihlašuje k systému přes počítačovou síť. Proto musí být tato komunikace mezi uživatelem a serverem, ke kterému uživatel přistupuje, zabezpečená. Úroveň zabezpečení může být na různých vrstvách komunikačních protokolů. Pokud ovšem nedůvěřujeme způsobu zabezpečení protokolů nižších vrstev, musíme implementovat zabezpečení komunikace na aplikační úrovni. V praxi je většinou v aplikacích implementováno více druhů mechanismů pro způsob autentizace a zabezpečení přenosu dat.

Cílem diplomové práce je návrh nového mechanismu autentizace a autorizace uživatele využívající důvěryhodný server pro ověření totožnosti uživatele. Dalším úkolem je tento mechanismus implementovat a ukázat jeho funkčnost.

Diplomová práce je rozdělena do dvou velkých celků. V prvním jsou zahrnuty kapitoly, které slouží jako teoretický základ k pochopení problematiky autentizace a autorizace uživatele. V tomto celku jsou popsány také protokoly a systémy, které realizují autentizaci uživatele pomocí důvěryhodného serveru. V těchto protokolech jsem hledal inspiraci pro druhou část diplomové práce. Druhá část obsahuje návrh nového schématu pro autentizaci uživatele a jeho implementaci. Dále je zde také na konkrétním příkladu popsán způsob, jakým řeší implementaci autentizačních mechanismů stávající aplikace.

Tato práce navazuje na Semestrální projekt, v rámci kterého jsem studoval způsoby autentizace a autorizace uživatele. Do diplomové práce jsem převzal části Semestrálního projektu, které popisují teoretický základ pro danou problematiku a kterými jsem se inspiroval při návrhu vlastního autentizačního schématu. Jsou to kapitoly zabývající se kryptografií (kapitola 2), autentizací a autorizací uživatele (kapitola 3), AAA schématem (kapitola 4) a protokolem Kerberos (kapitola 5).

Kapitola 2

Kryptografie

Kryptografie neboli šifrování je nauka o metodách utajování obsahu zpráv převodem do podoby, která je čitelná jen se speciální znalostí. Slovo kryptografie pochází z řečtiny - *kryptós* skrytý a *gráphein* znamená psát. Někdy je pojem obecněji používán pro vědu o čemkoli spojeném se šiframi jako alternativa k pojmu kryptologie. Kryptologie zahrnuje kryptografii a kryptoanalýzu, neboli luštění zašifrovaných zpráv.

Šifrování je důležitým prvkem zabezpečujícím data. Je použito všude tam, kde je požadováno nevyzrazení tajné informace. Používá se proto pro zabezpečení přenosu dat nebo archivaci dat.

Cílem této diplomové práce je návrh systému pro autentizaci a autorizaci uživatelů. Uživatel se bude autentizovat a autorizovat vůči vzdálenému serveru. A proto bude potřeba přenášet důvěrná data přes počítačovou síť. Kryptografické metody budou použity právě k účelu utajení přenášených důvěrných dat.

2.1 Symetrická šifra

Symetrické šifrování je založeno na principu jednoho klíče, kterým lze zprávu jak zašifrovat, tak i dešifrovat.

Podstatnou výhodou symetrických šifer je jejich nízká výpočetní náročnost. Algoritmy pro šifrování s veřejným klíčem (asymetrické algoritmy) mohou být i stotisíckrát pomalejší. Na druhou stranu velkou nevýhodou je nutnost sdílení tajného klíče, proto se odesilatel a příjemce tajné zprávy musí předem domluvit na tajném klíči.

Obrázek 2.1: symetrický algoritmus

2.2 Asymetrická šifra

Asymetrická kryptografie (kryptografie s veřejným klíčem) je skupina kryptografických metod, ve kterých se pro šifrování a dešifrování používají odlišné klíče.

Šifrovací klíč pro asymetrickou kryptografii sestává z dvou částí: jedna část se používá pro šifrování zpráv (a příjemce zprávy ani tuto část nemusí znát), druhá pro dešifrování (a odesilatel šifrovaných zpráv ji zpravidla nezná). Je vidět, že ten, kdo šifruje, nemusí s dešifrujícím příjemcem zprávy sdílet žádné tajemství, čímž eliminují potřebu výměny klíčů; tato vlastnost je základní výhodou asymetrické kryptografie.

Nejběžnější verzí asymetrické kryptografie je využívání tzv. veřejného a soukromého klíče. Šifrovací klíč je veřejný, majitel klíče ho volně uveřejní, a kdokoli jím může šifrovat jemu určené zprávy; dešifrovací klíč je soukromý, majitel jej drží v tajnosti a pomocí něj může tyto zprávy dešifrovat.

Je zřejmé, že šifrovací klíč `ecryptionKey` a dešifrovací klíč `decryptionKey` spolu musí být matematicky svázány, avšak nezbytnou podmínkou pro užitečnost šifry je praktická nemožnost ze znalosti šifrovacího klíče spočítat dešifrovací.

Matematicky tedy asymetrická kryptografie postupuje následujícím způsobem:

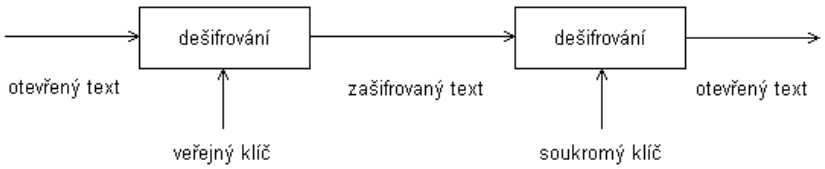
Šifrování

```
crypted = f(message, ecryptionKey)
```

Dešifrování

```
message = g(crypted, decryptionKey)
```

V principu se mohou šifrovací a dešifrovací funkce lišit, zpravidla jsou však matematicky přinejmenším velmi podobné.



Obrázek 2.2: asymetrický algoritmus

Kapitola 3

Autentizace a autorizace

Autentizace a autorizace se používá pro zajištění přístupu k systému nebo službě jen těm uživatelům, kteří jsou k tomu oprávněni. Tyto pojmy úzce souvisí s tématem diplomové práce, neboť jedním z cílů práce je návrh autentizačního a autorizačního mechanismu.

3.1 Autentizace

Autentizace znamená potvrzení, že uživatel požadující služby je platným uživatelem poskytovaných síťových služeb. Autentizace je dosažena pomocí představení identity a jistého pověření nebo tajemství.

Autentizace může být jednosměrná nebo obousměrná. Jednosměrná označuje autentizaci uživatele vůči serveru. Zde může dojít k podvržení identity serveru, proto je vhodné využít autentizaci obousměrnou. Ta umožňuje ověření identity jak uživatele tak i serveru poskytujícího službu. V ideálním případě je tedy zamezeno podvržení identity od obou koncových členů.

3.2 Způsoby autentizace

Uživatelské jméno a heslo

Znalost uživatelského jména a hesla je nejznámějším a nejběžnějším způsobem autentizace uživatele. K potvrzení identity stačí znát uživatelské jméno a heslo. Výhodou je snadná implementace tohoto autentizačního procesu a jeho nenáročnost na samotného uživatele.

Autentizační údaje se mění jen zřídka a proto je možno je získat velmi jednoduše. Ve chvíli, kdy se nám podaří tyto údaje získat, ať už odposlechem nebo jiným způsobem, získáme plný přístup k uživatelovu účtu. Spolehlivost autentizace je navíc velmi závislá na zvoleném hesle.

Jednorázové heslo

Jednou z nejjistějších obran proti odhalení hesla nebo jeho odchycení při komunikaci je používání jednorázových hesel. Většina implementací poskytujících autentizaci jednorázovým heslem vyžaduje kromě zvláštní konfigurace počítačového systému i speciální zařízení v podobě kalkulátoru velikosti běžné kreditní karty, kterým si uživatel pokaždé vygeneruje

nové heslo. Toto heslo je platné pouze po omezený časový úsek (maximálně několik minut) a celý přístroj je chráněn proti zneužití cizí osobou dalším heslem nebo číselným kódem. Jednorázové heslo se většinou generuje v závislosti na aktuálním čase a/nebo sériovém čísle zařízení. Posloupnost generovaných hesel nelze odhadnout, dvojí použití stejného hesla není možné.

Certifikát

Další možností je použití certifikátu, který byl vystaven na straně serveru. Certifikát pak používá služba pro ověření totožnosti jeho držitele. Po technické stránce se jedná o textový dokument, který může být uložen na libovolném médiu. Velmi často se můžeme setkat s certifikáty uloženými na pevném disku, odkud je lze zkopírovat. Jakmile dojde k odcizení certifikátu a příslušných privátních klíčů, je možno opět získat plný přístup k účtu původního majitele. Používá se proto umístění certifikátu na čipovou kartu, jejíž čtečku pak musí mít uživatel připojenu ke svému počítači, nebo uložení certifikátu na serveru certifikační autority (CA).

Autentizační kalkulátor

Autentizační kalkulátor je elektronické zařízení, které dokáže generovat jednorázová hesla pro přístup k účtu uživatele. Tato zařízení bývají synchronizována se systémy na straně serveru tak, aby obě strany generovaly stejné klíče. Ty pak uživatel opisuje do aplikace a ověřuje tak svou totožnost jednoduše tím, že dokáže, že je majitelem příslušného kalkulátoru. Jedná se o jednu z nejbezpečnějších metod autorizace uživatele.

3.3 Autorizace

Autorizace znamená udělení specifického typu služby (včetně odmítnutí služby) uživateli na základě jeho autentizace, služeb, které požaduje a aktuálního stavu systému. Autorizace může být založena na omezeních, například omezení na určité hodiny v rámci dne, nebo omezení na fyzickou polohu, nebo omezení vícenásobného přihlášení jednoho uživatele. Autorizace určuje povahu služby, která je poskytnuta uživateli. Typy služeb jsou například: filtrování IP adres, přidělení adresy, přidělení cesty, QoS, řízení šířky pásma/řízení toku, tunelování do konkrétního koncového bodu.

Ukázka autorizačního záznamu:

```
iptables -A INPUT -s 192.168.0.2 -p tcp --dport 443 -j REJECT
```

Popis: všechny dotazy na TCP portu 443 přicházející z adresy 192.168.0.2 odmítni

Kapitola 4

AAA schéma

V oblasti počítačové bezpečnosti AAA znamená *authentication, authorisation and accounting protocol*, tj. česky autentizační, autorizační a účtovací protokol. Autentizace a autorizace byla vysvětlena v úvodu. Účtování znamená sledování využívání síťových služeb uživateli. Tyto informace mohou být použity pro správu, plánování, účtování, nebo další účely. Účtování v reálném čase je doručeno současně s využíváním zdrojů. Běžně se sbírají informace o identitě uživatele, povaze dodaných služeb a časy počátků a konců dodaných služeb.

Pokud bychom chtěli kontrolovat autentizaci, autorizaci a účtování pro celou lokální síť, bylo by potřeba tyto tři služby umístit do jednoho uzlu, který bude dané žádané údaje ověřovat a zaznamenávat. K tomuto účelu se používají ověřovací servery, zapojené do sítě a ukryté někde hlouběji (viz obrázek 4.1).

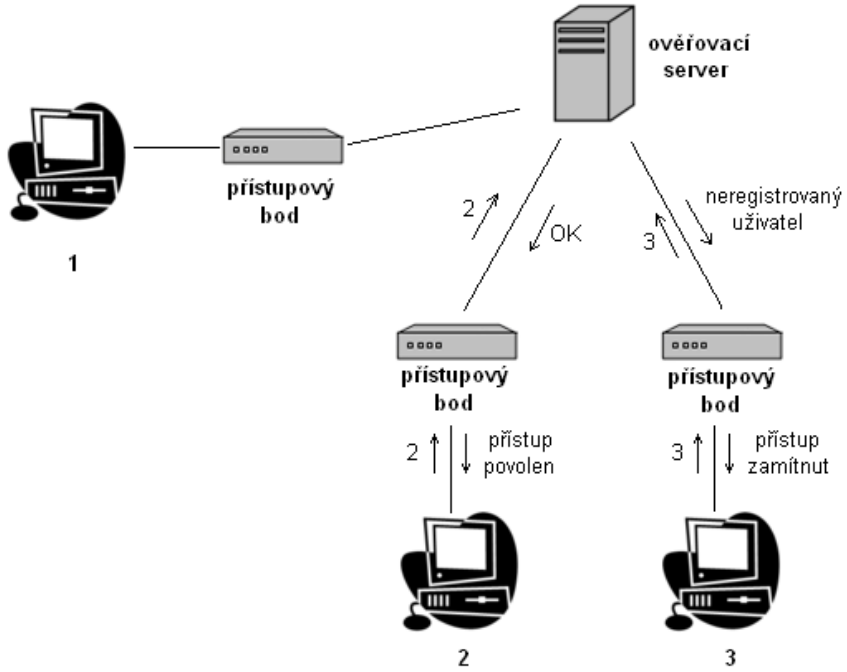
V diplomové práci jsem navrhoval autentizační a autorizační mechanismus, který vychází z AAA schématu. Informace uvedené v této kapitole sloužily jako inspirace při návrhu tohoto mechanismu.

Postup ověřování (obrázek 4.1):

- Klient č. 2 je oprávněný uživatel. Žádá přístupový bod (AP) o přístup k síti. AP se zeptá ověřovacího serveru, zda má klient č. 2 povolen přístup k síti. Ověřovací server odpovídá, že uživatel č. 2 je oprávněným uživatelem. Přistupový bod povolí uživateli č. 2 vstup do sítě.

- Klient č. 3 je neoprávněný uživatel. Žádá přístupový bod (AP) o přístup k síti. AP se zeptá ověřovacího serveru, zda má klient č. 3 povolen přístup k síti. Ověřovací server odpovídá, že uživatel č. 3 není oprávněným uživatelem. Přistupový bod zakáže uživateli č. 3 vstup do sítě.

Principu protokolu AAA využívají mnohé další protokoly. Jsou použity například v lokáních ethernetových sítích, mobilních sítích a bezdrátových sítích.



Obrázek 4.1: autentizace pomocí ověřovacího serveru

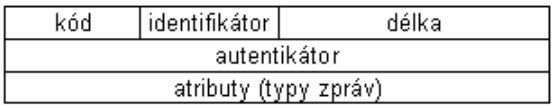
4.1 Radius

RADIUS (Remote Authentication Dial In User Service, česky Uživatelská vytáčená služba pro vzdálenou autentizaci) je AAA protokol používaný pro přístup k síti nebo pro IP mobilitu.

Při připojení k poskytovateli Internetu pomocí vytáčeného připojení, DSL, nebo Wi-Fi je u některých poskytovatelů vyžadováno přihlašovací uživatelské jméno a heslo. Tato informace je poslána do takzvaného Network Access Server (NAS) zařízení přes Point-to-Point Protocol (PPP [15]). Poté je předána RADIUS serveru přes RADIUS protokol. RADIUS server ověří pravost informace. Pokud je uživatelské jméno a heslo přijato, server autorizuje přístup k poskytovateli internetu a vybere IP adresu a další parametry spojení.

RADIUS server bude také upozorněn na spuštění nebo ukončení sezení, takže uživatel může platit přesně podle těchto RADIUS informací nebo mohou být tyto použity pro statistické účely.

RADIUS byl původně vyvinut společností Livingston Enterprises pro jejich PortMaster série Network Access Servers a později (1997) zveřejněny jako RFC 2058 [13] a RFC 2059 [11] (současné verze jsou RFC 2865 [14] a RFC 2866 [12]). V současné době existuje několik komerčních a open-source RADIUS serverů. Vlastnosti se liší, ale většina umožňuje dohledávat uživatele v textových souborech, LDAP serverech, různých databázích a podobně. RADIUS proxy servery jsou používány pro centrální správu a mohou přepisovat RADIUS pakety za běhu.



Obrázek 4.2: Radius paket

- Kód – definuje typ RADIUS paketu Nejčastější typy zpráv v protokolu RADIUS:
  - **Access-Request** - klient protokolu RADIUS (přístupový server) odesílá požadavek na ověření uživatele na RADIUS serveru.
  - **Access-Accepted** - RADIUS server sděluje, že uživatel může dostat povolení do síťě.
  - **Access-Reject** - RADIUS server zamítá přístup do síťě.
  - **Access-Challenge** - RADIUS server odesílá požadavek přístupovému serveru, aby klient zadal jednorázové heslo.
- Identifikátor – číslo zprávy
- Délka – délka paketu
- Autentikátor – slouží pro autentizaci (je zde uložen MD5 hash uživatelského hesla)
- Atributy – zde se zasílají různé informace jako uživatelské jméno, heslo...

Autentizace obsahuje dva dialogy:

1. mezi uživatelem a přístupovým bodem a cílem je sestavit zprávu Access-Request
2. dialog protokolu RADIUS obsahující např. zprávy Access-Request a Access-Accepted mezi pří-stupovým bodem a RADIUS serverem

Přístupový server si také musí být jistý, že jedná s korektním RADIUS serverem. Může nastat případ, že podvržený RADIUS server by povolil přístup do sítě neoprávněnému uživateli. Proto RADIUS server do své odpovědi vkládá řetězec, kterým svou odpověď autorizuje. Autorizace probíhá na základě sdíleného tajemství, které zná pouze RADIUS server a přístupový server.

RADIUS je jako autentizační protokol běžně používán v IEEE 802.1x bezpečnostním standardu [6] (často používán v bezdrátových sítích). I když nebyl RADIUS původně vytvořen pro autentizační metody v bezdrátových sítích, vylepšuje zabezpečení ve spojení s ostatními bezpečnostními metodami jako EAP-PEAP.

RADIUS je rozšiřitelný a většina výrobců zařízení a software používají vlastní RADIUS dialekty.

V roce 2003 byl vytvořen návrh protokolu DIAMETER, který je plánován jako náhrada RADIUS protokolu.

4.2 Diameter

Hlavní koncept tvoří základní protokol (RFC 3588 [5]), který může být rozšířen pro poskytování AAA služeb novým přístupovým technologiím.

Předchůdcem protokolu DIAMETER je protokol RADIUS (anglické diameter je česky průměr, což je dvakrát více než poloměr, anglicky radius). Diameter není přímo zpětně kompatibilní, ale poskytuje rozšířenou cestu pro RADIUS.

Hlavní rozdíly protokolu DIAMETER oproti protokolu RADIUS jsou:

- používá spolehlivý transportní protokol (TCP nebo SCTP, nepoužívá nespolehlivý UDP)
- podporuje přenos RADIUS
- má větší adresní prostor pro dvojice hodnot atributů (anglicky Attribute Value Pairs, AVPs) a širší identifikátory (32bitové místo 8bitových)
- jde o klient-server protokol, s výjimkou podpory některých zpráv inicializovaných serverem
- lze použít stavový i bezstavový model
- podporuje dynamické objevování uzlů (používá DNS, SRV a NAPTR)
- obsahuje schopnost vyjednávání
- podporuje dohody na aplikační vrstvě, definuje metody odolávající chybám a stavové stroje (RFC 3539 [2])
- oznamuje chyby
- má lepší podporu roamingu
- je snadněji rozšiřitelný; lze definovat nové příkazy a atributy
- je zarovnán na 32bitové hranice
- má základní podporu uživatelských sezení a účtování

Základní protokol Diametru

Základní protokol Diameteru (Diameter Base Protocol) je definován v RFC 3588 [5]. Určuje minimální požadavky AAA protokolu. Aplikace Diameteru mohou rozšířit základní protokol přidáním nových příkazů nebo atributů. Aplikace zde není program, nýbrž protokol založený na Diameteru.

4.3 EAP

EAP (Extensible Authentication Protocol - RFC 2284 [3]) byl původně určen pouze pro protokol PPP (Point-to-Point Protocol). Zajišťuje pro něj rámec (transportní mechanismus) pro všechny druhy ověřovacích metod, nicméně není jeho nedílnou součástí. Definice EAP

mimo PPP, do samo-statného protokolu, umožnilo jeho použití i v jiných prostředích. Např. modifikace EAP definovaná pod specifikací IEEE 802.1x je v podstatě rozšíření EAP pro sítě typu 802.

Protokol EAP sám o sobě nezajistí bezpečný ověřovací mechanismus - ten se vyjedná až protokolem EAP. Protokol EAP umožňuje použít libovolný ověřovací mechanizmus, který stačí implementovat na obou stranách spojení. Toto umožňuje poměrně snadno vytvářet nové modifikace protokolu. Protokol se v principu nemění, jen ověřovací mechanismus musí být známý na obou stranách spojení.

Pokud se použije protokol EAP, pak se fáze autentizace skládá:

- dohoda na ověřovacím mechanizmu
- vlastní autentizace

Navazování spojení

1. Přístupový bod, který ověřuje totožnost klienta, zašle zprávu EAP-Request. Tím žádá klienta o prokázání totožnosti.
2. Pokud klient souhlasí - odpovídá zprávou EAP-Response.
3. Přístupový bod zašle zprávu EAP-Request výzvu (Challenge)
4. Klient spojí sdílené tajemství (heslo) s výzvou a minimálně aplikuje jednocestnou funkci MD5 a výsledek vloží do odpovědi EAP-Response.
5. Přístupový bod ověří totožnost klienta a odpoví EAP-Succes/EAP-Failure (potvrdí/zamítne).

Nejčastější typy ověřovacích mechanismů:

- EAP-MD5 (RFC 1994, RFC 2284) Pro ověřování je používáno uživatelské jméno a heslo. Tyto údaje jsou pro zajištění pravosti hashovány pomocí MD5. Jedná se o původní specifikaci s jednocestným ověřováním. Tento způsob ověřování není vhodný u bezdrátových sítí - hrozí riziko odposlouchání.
- EAP-TLS (RFC 2716) Transport Level Security Pro ověřování použito PKI (Public Key Infrastructure) a SSL (Secure Sockets Layer), mechanismus ověřování je založen na použití ověření serveru na základě certifikátu serveru a ověření uživatele na základě certifikátu uživa-tele. Klíče jsou generovány dynamicky. Zprávy jsou chráněné před naslouchánim TLS (Transport Layer Security) tunelem. Nevýhoda - udržování PKI certifikátů pro klienta a server. Vhodné pro WLAN - TSL tunel chrání přenos před odposlechem.
- EAP-TTLS (Tunneled TLS); rozšíření modifikace EAP-TLS, ale bez použití certifikátů. Klient se prokazuje jménem a heslem. Vhodné tam, kde jsou požadavky na bezpečnost bez použití společných certifikátů.
- EAP-PEAP (Protected EAP); podobný TTLS, zajišťuje TLS k vytvoření tunelu pro druhý ověřovací algoritmus, ten je typu EAP. EAP-PEAP dovoluje dalším EAP ověřovacím protokolům používat zabezpečující přenos TLS tunelem.

- EAP-LEAP (Lightweight Extensible Authentication Protocol); Navrženo firmou Cisco. Podporuje vzájemné ověření klienta a sítě. Podobné jako EAP-TLS, ale místo certifikátů používá jméno a heslo.
- EAP-OTP One-Time Password Ověřování je zajišťováno pomocí systémů typu RSA SecureID, CRYPTOCard Token,...
- EAP-GTC (Generic Token Card) Obdoba EAP-OTP.
- EAP-AKA - Authentication and Key Agreement Použití v UMTS sítích.
- EAP-SKE - Shared Key Exchange
- EAP-GPRS - modifikace pro GPRS

4.4 802.1x

Rámec 802.1x nepatří přímo do rodiny protokolů AAA, ale lze na něm demonstrovat principy AAA protokolu.

IEEE 802.1x je obecný bezpečnostní rámec pro všechny typy LAN, zahrnující autentizaci uživatelů, integritu zpráv (šifrováním) a distribuci klíčů. Protokol 802.1x má za cíl blokovat přístup k segmentu lokální sítě pro neoprávněné uživatele.

Je založený na protokolu Extensible Authentication Protocol (EAP, RFC 2284 [3]). Jedná se o mechanizmus přenosu EAP paketů prostřednictvím spojové vrstvy LAN (typu 802) - zprávy EAP se zapouzdřují do rámců 802.1x.

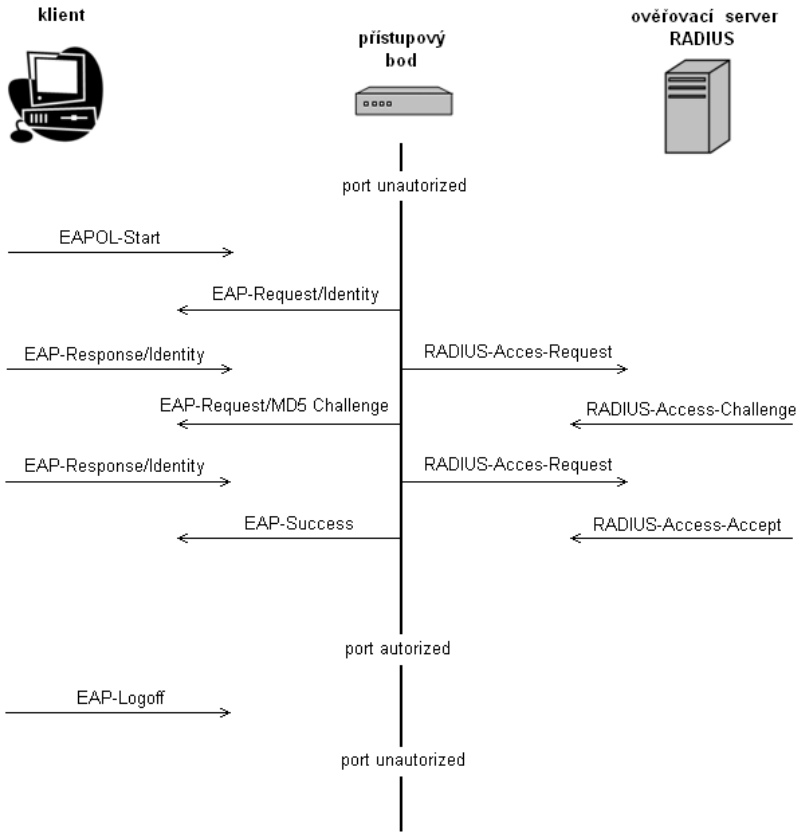
Jednou z klíčových vlastností 802.1x je to, že přístupový bod – NAS (network access server) může být v podstatě jednoduchý. Veškerá inteligence je v klientovi a ověřovacím serveru. To je pro přístupové body ideální, protože jsou typicky poměrně jednoduché s malou pamětí a výkonem procesoru.

Ověřování provádí přístupový bod pro klienty na základě jejich výzvy pomocí seznamu nebo externího autentizačního systému (serveru Kerberos nebo RADIUS). Pouze ověřený uživatel má možnost přístupu do sítě.

Postup autentizace podle 802.1x - obrázek 4.3 (je uvažován jen jednoduchý ověřovací algoritmus jednocestnou funkcí MD5)

Postup ověřování:

1. 1. Klient posílá požadavek přístupovému serveru (tj. switch nebo bezdrátový přístupový bod), že se chce přihlásit do sítě (EAPOL-Start).
2. Přístupový server posílá klientovi požadavek na ověření totožnosti (EAP-Request/ Identity). Pokud přístupový server (Network Acces Server - NAS) detekuje přítomnost klienta, tak zašle požadavek na ověření bez čekání na kontakt od klienta.
3. Klient posílá přístupovému serveru informaci o své totožnosti (EAP-Response/Identity). Přístupový server ji přebalí z EAP do RADIUS protokolu a posune ji na RADIUS server.



Obrázek 4.3: 802.1x autentizace

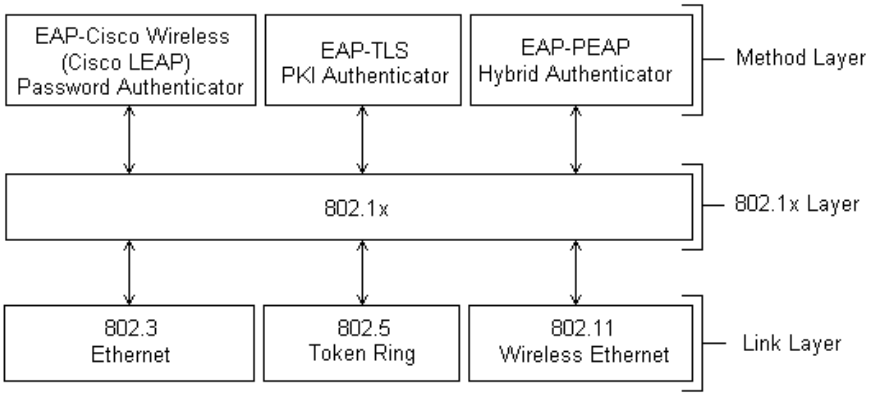
4. RADIUS server posílá zpět na přístupový server výzvu pro ověření (challenge). Přístupový server přebalí zprávu z formátu RADIUS do EAPOL a pošle ji klientovi . Rozdílné ověřovací metody mají různé množství paketů v této části procesu . EAP podporuje jak jednoduché ověření klienta tak i silné vzájemné ověřování.

5. Klient na výzvu odpovídá např. tak, že k výzvě přidá své heslo a tento řetězec zašifruje hash funkcí. Zašifrovanou zprávu posílá jako odpověď na přístupový server. Ten ji přepošle dál na RADIUS server.

6. RADIUS server zkontroluje údaje a odpoví zprávou o úspěšnosti/neúspěšnosti, která je přeposlána klientovi. Přístupový server umožní (neumožní) přístup do sítě - s možností restrikcí na základě atributů od RADIUS serveru. Například přiřadí klienta do konkrétní VLAN nebo nastaví filtrovací pravidla.

Architektura 802.1x

Jak již bylo řečeno EAP protokol zajišťuje pouze transportní mechanismus pro ověřování - tím je umožněno vytvářet nové modifikace protokolu - obrázek 4.4.



Obrázek 4.4: architektura

- vrstva ověřovací metody - řeší konkrétní metodu ověření uživatele (viz. protokol EAP)
- 801.1x vrstva - zajištění pravidel pro komunikaci komponentami systému (supplicant, authenticator a authentication server)
- Linková vrstva - uzpůsobení konkrétní LAN technologii (definice specifických rámců)

4.5 Shrnutí

V této kapitole byly rozebrány principy AAA protokolu a byly zde zmíněny aplikace používající přístup AAA schématu. Cílem diplomové práce byl návrh autentizačního mechanismu pomocí důvěryhodného serveru. Tento server měl v podstatě splňovat základní požadavky AAA schématu. Proto jsem pro roli důvěryhodného serveru vybral server Radius.

Kapitola 5

Autentizace pomocí systému Kerberos

Kerberos je jak protokol tak i systém zajišťující autentizaci a autorizaci uživatelů. Je definován standardem IETF RFC 1510 [9] a tvoří základní autenizační prvek v řadě komerčních i open-source systémů. Kerberos je navržen tak, aby zajišťoval silné zabezpečení současně s jednoduchým uživatelským rozhraním. Způsob autentizace je založen na použití tzv. *lístků* vydávaných centrálním autentizačním serverem, který spravuje databázi všech uživatelů. Lístky jsou analogické např. certifikátům veřejných klíčů, ale narozdíl od nich jsou kerberovské lístky a všechny kerberovské protokoly založeny výhradně na použití symetrické kryptografie, tj. hesla sdíleného mezi uživatelem a centrálním autentizačním serverem. Kerberovské lístky mají také kratší dobu platnosti (zpravidla deset hodin) a vždy obsahují informaci, pro kterou konkrétní koncovou službu jsou určeny.

Systém Kerberos ve vztahu k diplomové práci sloužil jako inspirace. Byly na něm zkoumány autentizační postupy a principy. Při implementaci byl použit pro analýzu zdrojových kódů aplikací používajících autentizaci pomocí důvěryhodné třetí strany.

Historie

První verze byly vyvinuty v rámci projektu Athena v letech 1983-1991 v laboratořích MIT (Massachusetts Institute of Technology) a první veřejnou verzí byla verze v4. Po zhruba šestiletém vývoji a upravování na základě zkušeností uživatelů vznikla verze v5, která se v současnosti používá.

5.1 Protokol Kerberos

Hlavní částí systému Kerberos je KDC (Key Distribution Center), které je důvěryhodnou třetí stranou umožňující bezpečnou autentizaci a autorizaci. KDC bezpečně uchovává data o uživatelích a službách v síti.

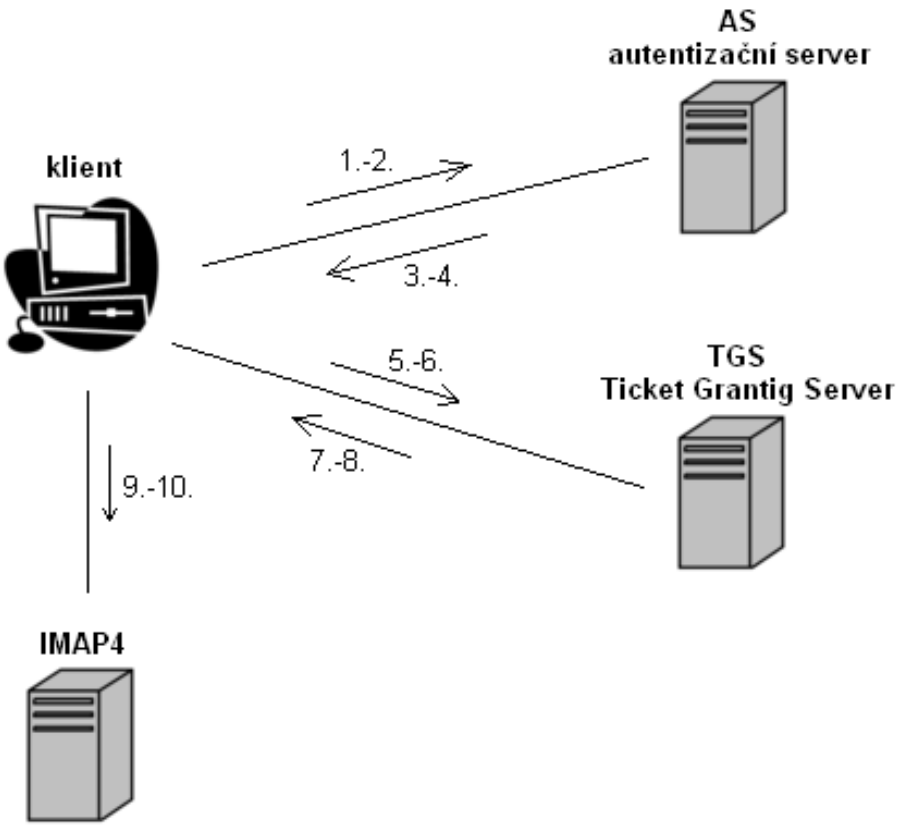
KDC se dělí na dva subsystémy:

- AS - autentizační server, zajišťuje autentizaci a autorizaci uživatelů a služeb
- TGS - Ticket Grantig Server, zajišťuje přidělování oprávnění - lístků - k použití zdroje (služby)

Dále v systému vystupují klientské aplikace a servery, na nichž běží služby.

Při přístupu ke službě je potřeba ověřit autentizační údaje klienta serverem AS, který mu po úspěšné autentizaci vydá lístek umožňující požádat TGS o udělení přístupu ke službě. Jakmile ho klient obdrží a prokáže se jím požadované službě, která ověří jeho platnost, může být služba používána po dobu platnosti tohoto lístku.

Proces autentizace a autorizace - viz. obrázek 5.1



Obrázek 5.1: autentizace a autorizace pomocí protokolu kerberos

1. Klientský proces si vyžádá uživatelské jméno, které pošle v nezašifrované podobě AS (tzn. klient tímto žádá o přidělení TGT). AS má centrální databázi uživatelů a služeb, kde se nalézá heslo uživatele, které musí být shodné s heslem známým na straně klienta.

2. AS přijme tuto žádost (většinou tyto žádosti ukládá kvůli větší bezpečnosti do cache), následně podle uživatelského jména vyhledá v centrální databázi heslo uživatele a z něj vygeneruje dočasný klíč - session-key. Vygeneruje dále tzv. TGT lístek - Ticket Granting Ticket, který obsahuje: uživatelské jméno, jméno TGS serveru (vydává oprávnění použít zdroj), síťovou adresu klienta, session-key

   Tento TGT lístek je poté zašifrován pomocí tajného klíče, který je bezpečně sdílen pouze AS a TGS.

3. AS posílá klientskému procesu tento TGT lístek a spolu s ním mu posílá také dvojici: session-key, n (nonce, náhodný identifikátor či timestamp kvůli zvýšení bezpečnosti). Tato dvojice je zašifrována klientským (uživatelským) heslem uloženým v centrální databázi AS.

4. Klientský proces pomocí klíče vygenerovaného ze svého uživatelského hesla dešifruje tuto dvojici a získá tak session-key a může také zkontrolovat platnost n. Klient tedy nyní vlastní TGT a session-key, které budou dále použity v komunikaci s TGS. To, že klient správně dešifruje a získá session-key, v podstatě znamená úspěšnou autentizaci vůči AS.

5. Klientský proces žádá nyní o použití služby tímto způsobem: vytvoří si tzv. autentifikátor, který obsahuje (username,useraddress,timestamp), timestamp udává čas vytvoření autentifikátoru a jeho platnost. Tento autentifikátor je zašifrován pomocí session-key a odeslán TGS spolu s žádostí o využití požadované služby a TGT lístkem.

6. TGS přijme tyto údaje a pomocí klíče, který bezpečně sdílí s AS dešifruje TGT lístek, z něhož získá session-key. Pomocí tohoto klíče dešifruje autentifikátor a zkontroluje jestli souhlasí informace na TGT lístku (username,useraddress) a v autentifikátoru. Dále zkontroluje jeho platnost - timestamp.

7. Po kladném ověření generuje nový lístek = pověření použít požadovanou službu běžící na serveru S. TGS generuje nový klíč TGS-session-key, který bude použit v komunikaci klienta a serveru S. TGS vytvoří lístek (username,useraddress,timestamp), který zašifruje pomocí klíče TGS-session-key a odešle jej klientskému procesu spolu s novým TGS-session-key šifrovaným pomocí session-key.

8. Klient tyto lístky dešifruje pomocí session-key a tak získá TGS-session-key. Vlastní tedy tento klíč a současně lístek opravňující použít službu serveru S.

9. Klientský proces vytváří další autentifikátor (username,useraddress,timestamp) šifrovaný pomocí TGS-session-key a posílá jej na server S, kde běží požadovaná služba.

10. Server S ověří údaje z autentifikátoru tak, že dešifruje lístek klíčem, který sdílí s TGS. Získá z něj TGS-session-key, kterým dešifruje autentifikátor a zkontroluje zda údaje souhlasí.

Shrnutí

V této kapitole byl popsán protokol Kerberos a proces autentizace pomocí systému Kerberos. V další práci byl použit jako inspirace pro návrh autentizačního mechanismu. Byl rovněž využitý pro analýzu zdrojových kódů aplikací používajících autentizaci pomocí důvěryhodné třetí strany.

Kapitola 6

Programové nástroje pro autentizaci

Každá správně navrhnutá aplikace by se měla skládat z hierarchicky strukturovaných vrstev, které by měly zajišťovat specifickou funkčnost. Tyto vrstvy mezi sebou musí komunikovat přes jasně definované rozhraní. Tento přístup má řadu výhod a mezi nejdůležitější patří:

- Odstínění implementačních detailů nižších vrstev před vrstvami vyššími – poskytuje vyšším vrstvám abstraktní pohled.
- Nižší vrstvy nejsou nijak svázány s vrstvami vyššími. O jejich existenci nevědí a nijak se na ně neodvolávají.
- Zvýšení přehlednosti celé aplikace, která je přehledně rozdělena na jednotlivé funkční celky.
- Jednoduché rozšiřování funkčnosti aplikací přidáním nové nižší vrstvy.

Ideální stav by byl, pokud by aplikace se stejným zaměřením (např. síťové aplikace s požadavkem autentizace uživatele) využívaly totožnou hierarchickou strukturu vrstev a jejich rozhraní. Lišily by se pouze ve funkčnosti jednotlivých vrstev. Rozhraní by stále zůstalo stejné. Toto by například umožnilo jednoduchou implementaci nového autentizačního mechanismu do všech existujících aplikací. Avšak dodržení této podmínky je nereálné.

Každá rodina aplikací používá svoje vlastní definované vrstvy. Proto je nutné abych si vybral pro řešení diplomové práce právě jednu takovou skupinu aplikací. Prozkoumal její vrstvy a rozhraní a modifikoval aplikace tak, aby byly schopny autentizovat uživatele nově navrhnutým mechanismem. Pro konkrétní případ studia autentizačního rozhraní aplikací jsem vybral komunikaci mezi poštovním serverem `imapd` a emailovým klientem `pine`.

6.1 Aplikace vybrané pro analýzu

University of Washington (dále UW) a její fakulta Počítačů a Komunikací se zabývá již dlouhou dobu výzkumem a vývojem síťových aplikací. Již v roce 1989 lidé z univerzity vytvořili známého emailového klienta `pine`. Dále vytvořili i emailové servery `POP2`, `POP3` a `IMAP`, které stále aktualizují a přizpůsobují požadavkům moderní doby.

Jedním z požadavků byla i podpora nových autentizačních protokolů. Proto v dnešní době všechny aplikace vytvořené washingtonskou univerzitou podporují širokou škálu těchto protokolů.

Mezi nejdůležitější autentizační mechanismy patří:

- ověření pomocí unixového hesla
- ověření pomocí PAM modulu
- ověření pomocí systému Kerberos

Z posledně zmíněného systému Kerberos jsem vycházel při analýze autentizačních mechanismů implementovaných v aplikacích University of Washington. Jednalo se konkrétně o balíček imap2006e (Příloha A), který obsahuje aplikace `imapd`, `pop2d` a `pop3d` a balíček pine4.64 (Příloha A), který obsauje emailového klienta `pine`. Tyto aplikace jsem rovněž využil pro implementaci nově navrhnutého autentizačního mechanismu a demonstroval jsem na nich jeho funkčnost.

6.2 Instalace a konfigurace systému pro analýzu

Jak již bylo zmíněno, aplikace UW z balíčku imap2006e a pine4.64 obsahují stejné programové rozhraní pro autentizační moduly. Proto nebylo nutné zkoumat všechny aplikace přítomné v tomto balíčku ale stačilo vybrat jen dvě pro komunikaci klient-server. Pro analýzu a implementaci jsem vybral jako klientskou aplikaci emailového klienta `pine` a jako poštovní server sloužil `imapd`.

Instalace

Cílem instalace bylo rozběhnutí poštovního IMAP serveru. Autentizace byla požadována přes systém Kerberos. Proto jsem k instalaci vybral následující balíčky:

- `postfix`2.3.3-2- jako SMTP server
- `imap`2006e- jehož součástí je IMAP server
- `krb5`1.6.1- systém Kerberos
- `pine` emailový klient

Aby byl postup instalace jednoduše opakovatelný, provedl jsem instalaci systému na virtuálním stroji aplikace VMware Player (Příloha A). Jako operační systém posloužila linuxová distribuce Fedora6 (Příloha A).

Seznam balíčků potřebných pro instalaci systému (detailní postup instalace je popsán v příloze A):

poštovní server postfix
postfix-2.3.3-2
pcre-6.6-1.1
maildop-2.0.3 (verze 2.0.4 způsobovala problémy)

Kerberos
gcc-4.1.1-51
gcc-c++-4.1.1-51
openssl-0.9.8e (verze 0.9.8b způsobovala problémy)
ncurses-5.6
byacc-1.9-29.2.2
flex-2.5.4a-41
libsepol-1.14
checkpolicy-1.34.1
libselinux-1.34.7
Linux-PAM-0.99.7.1
krb5-1.6.1

imap server
xinetd-2.3.14-8
imap-2006e

emailový klient pine
pine-4.64

Konfigurace

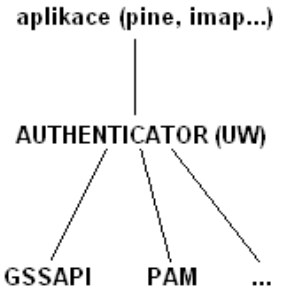
Provedl jsem jednoduchou konfiguraci poštovního serveru a systému Kerberos (popis konfigurace je v příloze B). Jednotlivé systémy se konfigurovaly buď pomocí konfiguračních souborů, nebo jako v případě IMAP serveru změnou parametrů při kompilaci aplikace.

6.3 Kerberizované aplikace

Chceme-li používat autentizaci pomocí systému Kerberos, musí být na klientském počítači nainstalován klient systému Kerberos. Tento balíček obsahuje samostatné nezávislé aplikace (`kinit`, `klist`, `kdestroy`, `kpasswd`), které udržují informace o lístcích (viz. Kerberos 5) a zprostředkovávají komunikaci mezi uživatelem a serverem Kerberos. Dále jsou v balíčku obsaženy sdílené knihovny, přes které mohou aplikace komunikovat se serverem Kerberos. Využití těchto knihoven není pro programátora aplikace povinné, ale usnadňuje mu výrazně práci. Tyto knihovny zajišťují autentizaci a autorizaci klienta vůči serveru a informace o úspěchu či neúspěchu operace předávají výše aplikaci.

6.4 Programové rozhraní aplikací

Všechny aplikace z balíčku imap2006e obsahují jednotné autentizační vrstvy, které udávají metodu autentizace a její způsob provedení. Pokud se jedná o autentizaci přes systém Kerberos, mohou se jednotlivé vrstvy seřadit hierarchicky v následujícím pořadí.



Obrázek 6.1: Hierarchie autentizačních vrstev

Jak je vidět z obrázku 6.1, výběr způsobu autentizace je ekvivalentní použitému autentikátoru (pro systém Kerberos: GSSAPI nebo PAM modul). U jiných aplikací je členění autentizačních mechanismů podobné a stojí proto za zmínku ještě nativní podpora Kerbera. Všechny důležité vrstvy (AUTHENTICATOR, GSSAPI, nativní podpora, PAM) budou vysvětleny dále v textu.

Struktura Authenticator (UW)

Toto jednotné rozhranní bylo vytvořeno UW za účelem výběru autentizačního mechanismu. Jak již bylo zmíněno, autentizační mechanismus může být ověření pomocí unixového hesla, PAM modulu nebo systému Kerberos. Pokud se podíváme na Authenticator z pohledu programátora, jedná se o strukturu v jazyce C, která obsahuje informace o autentizační metodě a ukazatele na funkce, které autentizaci na straně klienta a serveru provádějí. Abych ukázal konkrétní příklad authentikátoru, demonstroval jsem vše na příkladu autentizace pomocí GSSAPI.

Samotný Authenticator je struktura:

```
AUTHENTICATOR {
    long flags;            - příznaky autentikátoru
    char *name;            - jméno autentikátoru
    authcheck_t valid;     - adresa ověřovací funkce
    authclient_t client;   - adresa klientské funkce
    authserver_t server;   - adresa serverové funkce
    AUTHENTICATOR *next;   - ukazatel na další autentikátor
};
```

V případě autentikátoru pro metodu GSSAPI bude struktura definovaná následovně:

```
AUTHENTICATOR auth_gss = {
    AU_SECURE | AU_AUTHUSER,
    ‘‘GSSAPI’’,
    auth_gssapi_valid,
    auth_gssapi_client,
    auth_gssapi_server,
    NIL
};
```

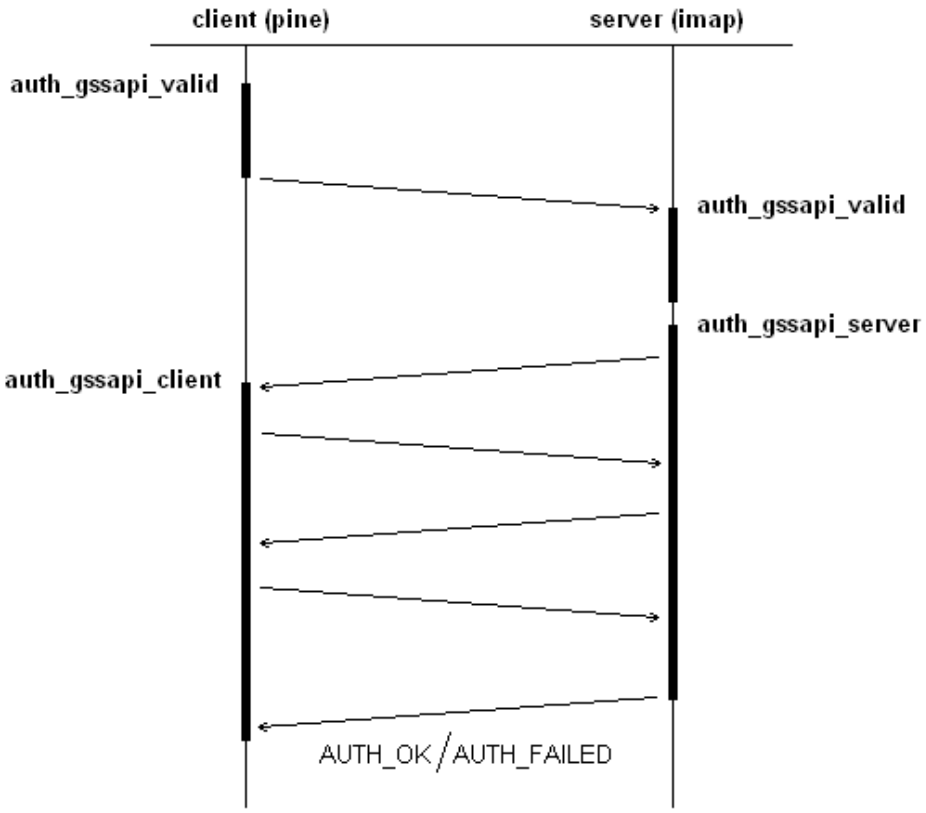
parametr	**význam**
`auth_gss`	jméno autentizační metody, která byla vybrána před samotným překladem aplikace. Jméno metody se skládá z prefixu “`auth_`” + “`gss`”. Hodnota “`gss`” odpovídá metodě autentizace vybrané v hlavním Makefile jako hodnota parametru PASSWDTYPE

`AU_SECURE`
`AU_AUTHUSER`
`AU_HIDE`
	příznaky autentikátoru. Pomocí logického součtu může být vybrána kombinace více možností

`AU_SECURE`	bezpečný autentikátor, autentikátor zajišťující bezpečnou výměnu hesel a bezpečnou autentizaci
`AU_AUTHUSER`	obousměrná autentizace
`AU_HIDE`	skrytý autentikátor, metoda autentizace nebude zobrazena uživateli ani serveru
`auth_gssapi_valid`	ověření, zda je metoda autentizace dostupná na obou stranách (klient-server)
`auth_gssapi_client`	autentizační funkce pro klienta
`auth_gssapi_server`	autentizační funkce pro server. Nejprve se volá funkce `auth_gssapi_server` na straně serveru. To způsobí zavolání funkce `auth_gssapi_client` na straně klienta. Dále běží tyto funkce souběžně a ověřují autentizační údaje. Na konci vrací údaje o úspěchu/neúspěchu autentizace. Celý proces komunikace je znázorněn na obrázku 6.2

Funkce `auth_gssapi_valid`, `auth_gssapi_client` a `auth_gssapi_server` musí mít také přesně definované parametry.

```
long auth_gssapi_valid (void);
```



Obrázek 6.2: Atentizace GSSAPI - časový diagram

```
long auth_gssapi_client (
    authchallenge_t challenger, - funkce, přijímající příchozí zprávy
    authrespond_t responder,    - funkce, odesílají zprávy
    char *service,              - název služby (IMAP, POP3...)
    NETMBX *mb,                 - emailová schránka, kterou chceme otevřít
    void *stream,               - aktuální spojení mezi klientem a serverem
    unsigned long *trial,       - počet zbývajících pokusů o přihlášení
    char *user                  - jméno uživatele
);

char *auth_gssapi_server (
    authresponse_t responder, - funkce odesílající a přijímající zprávy
    int argc,                 - počet parametrů příkazové řádky
    char *argv[]              - pole parametrů příkazové řádky
);
```

Nativní podpora

Aplikace si řídí komunikaci se serverem Kerbera sama. Všechny náležitosti jako získání a distribuce lístků je plně v moci aplikace. Toto řešení není obsaženo v aplikacích UW, ale je přítomno v balíčku, který nabízí Massachusetts Institute of Technology (MIT) (zde byl systém Kerberos vytvořen).

Rozhraní GSSAPI

GSSAPI (Generic Security Services Application Programming Interface) RFC 2744 [16] je generické aplikační rozhranní poskytující autentizaci klient-server. Jedná se pouze o mezivrstvu mezi aplikací a autentizačním protokolem. Smyslem GSSAPI je poskytnout aplikaci jednotné programové rozhraní k různým autentizačním metodám (prakticky mnohem detailnější definici rozhraní AUTHENTICATOR vytvořeného v UW). V praxi se používá právě pro autentizaci pomocí Kerbera. V důsledku toho, když se o aplikaci prohlásí, že podporuje GSSAPI, tak se myslí, že podporuje systém Kerberos.

Mezi nejdůležitější funkce GSSAPI patří:

`gss_acquire_cred` - server zasílá požadavek na získání tzv. gss-api credential. Credential ke struktura s jedinečným ID. Tato struktura obsahuje informace o autentizačních mechanismech klienta (v našem případě `pine`). Server porovná dostupné mechanismy klienta se svými a nabídne klientovi použití společně známých mechanismů (ve výsledku nabízí většinou autentizaci systémem Kerberos).

`gss_release_cred` - zrušení credentials

`gss_init_ces_context` - vytvoření bezpečného spojení mezi aplikací a vzdálenou stranou. Funkce může vrátit autentizační token, který bude přeposlán druhému účastníkovi. Ten token předá funkci `gss_accept_sec_context`

`gss_accept_sec_context` - souhlas k vytvoření bezpečného spojení

`gss_delete_sec_context` - zrušení bezpečného spojení

`gss_wrap` - připojení kontrolního součtu ke zprávě a volitelně její zašifrování

`gss_unwrap` - dešifruje, pokud byla zpráva zašifrovaná a zkontroluje kontrolní součet, zda nebyla zpráva pozměněna

Demonstrujeme-li autentizaci pomocí GSSAPI (Kerberos) mezi aplikacemi `pine` a `imap`, budou volání důležitých funkcí řazena hierarchicky v následujícím sledu (pořadí všech volaných funkcí je v příloze C):

```
pine: auth_gssapi_valid

imap: auth_gssapi_valid
      auth_gssapi_server
      gss_acquire_cred

pine: auth_gssapi_client
      gss_init_sec_context

imap: gss_accept_sec_context

pine: gss_init_sec_context

imap: gss_wrap

pine: gss_unwrap
      gss_wrap
      gss_delete_sec_context

imap: gss_unwrap
      gss_delete_sec_context
      gss_release_cred
```

Autentizační schéma PAM

PAM (Pluggable Authentication Modules) je sjednocené autentizační schéma, které je implementováno v mnoha open source verzích UNIXu. Je to sada sdílených knihoven, která umožňuje administrátorovi snadno (za běhu) přizpůsobovat autentizační služby pro různé aplikace, a ty již tedy nemusejí být při jakékoliv změně autentizačních mechanismů opětovně přepisovány a překompilovávány. Jedná se o modulární přístup k autentizaci, který vznikl jako reakce na neustále se zvyšující počet nově vznikajících autentizačních mechanismů. Kromě statických hesel umožňuje PAM používání například jednorázových hesel, čipových karet či biometrik.

Shrnutí

V této kapitole jsem popsal instalaci a konfiguraci systému pro autentizaci pomocí systému Kereberos. Prozkoumal a popsal jsem autentizační vrstvy aplikací balíčku imap2006e pro implementaci autentizace přes Kerberos.

Jedním z cílů diplomové práce je implementace nového autentizačního mechanismu. Před samotnou implementací se musím rozhodnout do jaké vrstvy aplikací tento mechanismus přidám. V zásadě jsou dvě možnosti, buď se zaměřím na vrstvu PAM a nebo přidám nový mechanismu do vrstvy AUTHENTICATOR.

Kapitola 7

Model autentizace a autorizace uživatele v globálních sítích

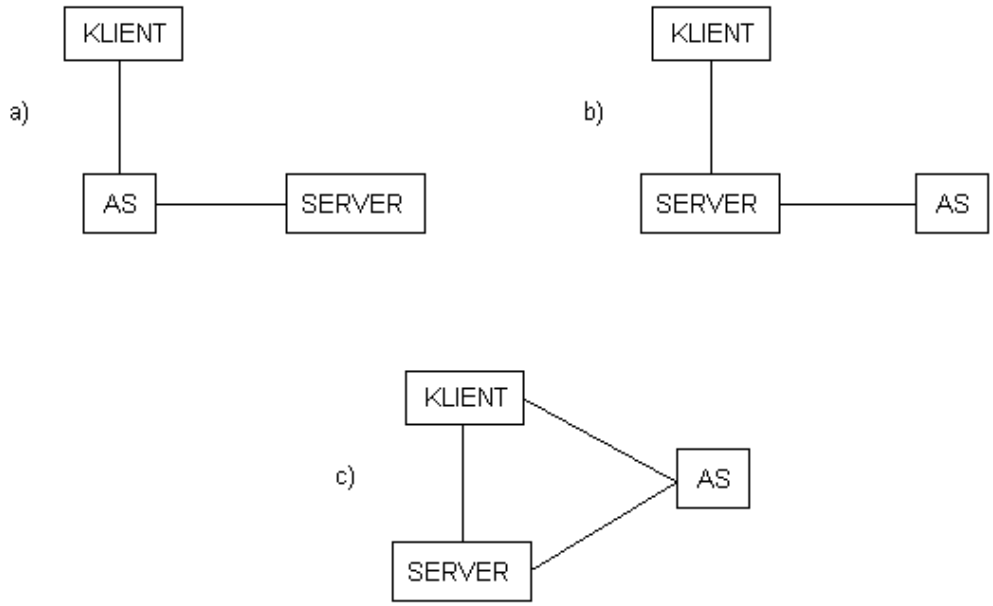
Další částí diplomové práce je návrh modelu autentizačního mechanismu. Tento model by měl splňovat určité požadavky definované v kapitole 7.1. V dalších kapitolách je popsán celý proces návrhu modelu a výsledné specifikace vytvořených nebo použitých protokolů.

7.1 Požadavky

- **Bezpečnost modelu** – Bezpečnost je jedním z nejdůležitějších požadavků. Pokud by někdo dokázal získat cizí uživatelské jméno a heslo, dostal se tak ke službě, ke kterým má napadený uživatel přístup. Z toho důvodu bude na bezpečnost kladena vysoká míra pozornosti.
- **Přístup login/heslo** – Autentizační mechanismus by měl využívat přístup klasického zadávání loginu a hesla. Bude tím uživatelské rozhraní nezměněno a uživatel se nebude muset učit nové postupy pro autentizaci.
- **Autentizace a autorizace uživatele pomocí důvěryhodné třetí strany** – Požadovaná služba nebude provádět sama autentizaci a autorizaci uživatele. Svěří tuto činnost důvěryhodné třetí straně.
- **Jedno heslo k více službám** – Model systému by měl umožňovat zjednodušení pro uživatele tím, že bude moci přistupovat k více službám přes jedno uživatelské jméno a heslo. Tím pádem bude moci uživatel zvolit složitější heslo a tím zvýší bezpečnost.
- **Administrace práv uživatele v jednom místě** – Systém by měl využívat důvěryhodnou třetí stranu pro ověřování uživatelů. Ta bude sloužit jako centrální správa uživatelských přístupových práv ke službám.
- **Přiměřené zatížení sítě** – Komunikace mezi ověřovacím serverem, serverem s požadovanou službou a klientem nesmí příliš zatížit síť. Proto bude dbáno na přiměřenou složitost protokolu.
- **Jednoduchost komunikačního protokolu** – Přílišná složitost protokolu může zabránit jeho použití, proto by měl být tak jednoduchý, jak to jen dovolí bezpečností požadavky.

7.2 Typy modelů komunikace

Autentizace uživatele vůči serveru pomocí důvěryhodné třetí strany může být z hlediska řazení jednotlivých komponent (klient, server, ověřovací server) rozdílná. Pokud uvažujeme, že autentizační mechanismus navrhujeme celý od začátku bez daných pravidel, musíme vybrat jedno z následujících tří schémat.



Obrázek 7.1: Možnosti výběru modelů

KLIENT - klientská aplikace
SERVER - server, kde běží požadovaná služba
AS - autentizační a autorizační server (důvěryhodná třetí strana)

- varianta a) - Klient se nejprve autentizuje vůči AS, který tvoří spojovací prvek mezi serverem a klientem. AS dále kontaktuje server a zprostředkovává následnou komunikaci. Zde se naskýtá další možnost, zda toto zapojení bude platit pouze pro autentizaci a autorizaci a přenos dat již bude probíhat přes přímé spojení.
- varianta b) - Klient přistupuje přímo k serveru a ten se ptá AS, zda tento klient je autentizován a oprávněn využívat službu na serveru. Toto zapojení je v praxi využíváno například pro přístup klientů k Wi-Fi síti, kde jako AS může sloužit server Radius.
- varianta c) - Klient přistupuje nejprve k AS, zde se autentizuje, autorizuje a získává identifikaci, díky které může přistupovat k serveru. Server si tuto identifikaci může ověřit u AS, zda je opravdu platná. Pokud ji AS potvrdí, server povolí klientovi přístup. Tato varianta - mírně upravená - je využita v systému Kerberos.

Vybíral jsem variantu s ohledem na možnost začlenění nového mechanismu do stávajících aplikací. Rovněž jsem bral zřetel na míru nutných úprav systémů pro jednotlivé možnosti. Po zvážení těchto okolností jsem vybral variantu b).

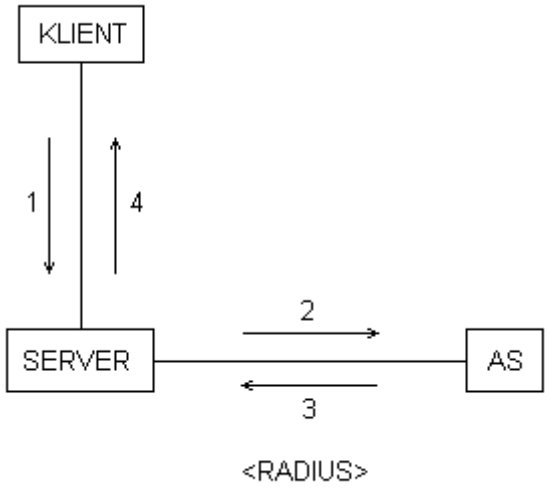
Klient bude přistupovat k požadovanému serveru stejným způsobem, jako by žádný autentizační server (AS) neexistoval. Server dále předá přihlašovací údaje AS, který klienta ověří a předá serveru informace o výsledku. Dále probíhá komunikace již jen mezi klientem a serverem.

Schéma komunikace je nyní vybráno a v další kapitole vytvořím model komunikačního protokolu.

7.3 Návrh modelu komunikace

Tím, že komunikace klient-server a server-AS bude na sobě nezávislá, budu moci použít pro komunikaci server-AS již existující protokol. Jak již bylo zmíněno, vybraná varianta se používá při ověřování uživatele pomocí serveru Radius ve Wi-Fi sítích. V mém případě se tímto inspiruji a použiji protokol Radius právě pro komunikaci server-AS. Jediné, co bude nutné definovat, budou přenášené informace na cestě server-AS(Radius) a AS(Radius)-server.

Obecně bude autentizace probíhat následovně:



Obrázek 7.2: Schéma komunikace

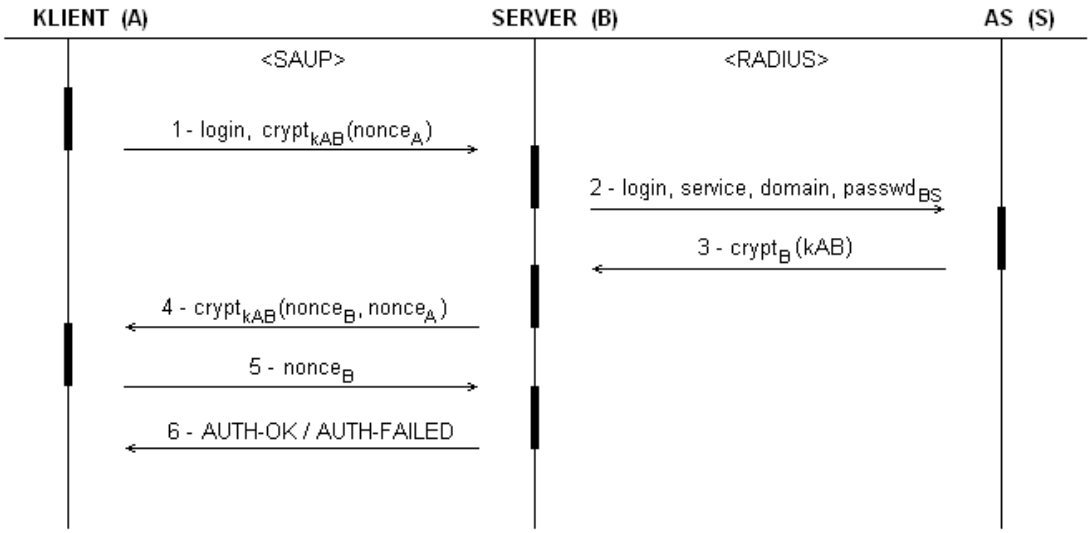
Popis schématu:

1. klient posílá autentizační údaje serveru
2. server přeposílá informace AS(Radius)
3. AS(Radius) zasílá výsledek autentizace klienta
4. server odpovídá klientovi “Autentizován”/“Neautentizován”

Zbývá ještě navrhnout komunikaci mezi klientem a serverem.

7.4 Návrh protokolu SAUP - Single Authentication Protocol

Komunikace mezi klientem a serverem bude probíhat mnou navrženým protokolem SAUP (Simple Authentication Protocol). Měla by být zabezpečená a odolná vůči útokům. Z toho důvodu použiji šifrování komunikace, bude se jednat o symetrickou šifru AES s délkou klíče 128 bitů. Výsledek návrhu je zobrazen na obrázku 7.3.



Obrázek 7.3: Časový průběh komunikace

Popis komunikace SAUP protokolu:

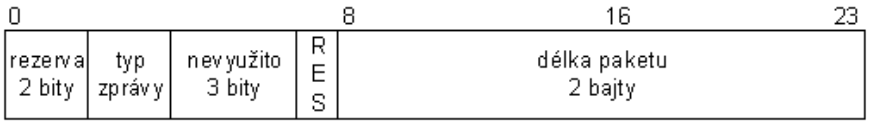
1. Klient posílá serveru *login* a zašifrované náhodné číslo $nonce_A$. Klíčem kAB je hash zadaného hesla uživatele.
2. Server potřebuje zjistit uživatelské heslo, protože je při komunikaci použito k odvození šifrovacího klíče kAB. Zasílá proto dotaz na AS(S) s informacemi:
   - login - přihlašovací jméno uživatele
   - service - jméno žádající služby (například IMAP)
   - domain - doména, ze které se server(B) dotazuje
   - passwd$_{BS}$ - heslo, které je sdíleno mezi serverem(B) a AS(S)
3. AS(S) odpovídá a posílá klíč kAB k šifrované komunikaci klient-server. Tento klíč se nezasílá v otevřené podobě, ale je zašifrován tajným klíčem kB, který zná pouze server(B). AS(S) se o šifrování klíče kAB nestará, protože ho má již zašifrovaný ve své databázi ve formě $crypt_{kB}(kAB)$.
4. Server(B) si rozšifruje klíč kAB a pomocí něj zjistí náhodné číslo $nonce_A$. Vygeneruje si své náhodné číslo $nonce_B$ a spojí s číslem $nonce_A$ v pořadí $nonce_B$, $nonce_A$. Spojené číslo $nonce_{BA}$ zašifruje a zašle klientovi.

5. Klient pomocí klíče kAB dešifruje zprávu a zjistí náhodné číslo $nonce_B$
6. Klient zašle serveru(B) v otevřené podobě náhodné číslo $nonce_B$. Pokud náhodné číslo $nonce_B$ odpovídá tomu číslu, které server(B) vygeneroval, pak proběhla autentizace úspěšně.
7. Server(B) posílá zprávu klientovi o úspěchu/neúspěchu autentizace.

Popis zprávy protokolu SAUP

Protokol SAUP dělí zprávu na dvě části: hlavičku a data.

Hlavička protokolu SAUP:



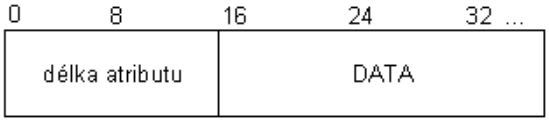
Obrázek 7.4: Hlavička protokolu SAUP

Pole příznaků:

- 2 bity - rezervováno
- 2 bity - typ zprávy
- 3 bity - nevyužito
- 1 bit - RES - výsledek autentizace `auth_ok / auth_failed`

Délka zprávy - Obsahuje délku zprávy (hlavička + data) v bajtech.
Data následují bezprostředně za hlavičkou.

Datová část SAUP:



Obrázek 7.5: Datová část protokolu SAUP

Protokol SAUP umožňuje přenášet v jedné zprávě více atributů najednou. Každý atribut je tvořen dvojicí *délka*, *data*. Maximální velikost přenášených dat je 64kB - *délka hlavičky*. Počet přenášených atributů není v principu omezen, pouze platí omezení velikosti celkových přenášených dat.

Nicméně stále platí pravidlo daného obsahu zprávy podle typu zprávy.

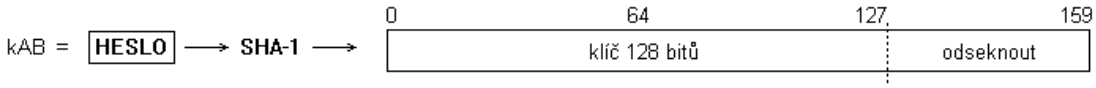
7.5 Vytvoření šifrovacího klíče k_{AB}

Protokol SAUP používá pro zabezpečení komunikace symetrickou šifru AES v režimu CBC. Všechny chráněné položky zpráv mezi klientem a serverem jsou šifrovány jedním klíčem. Šifra AES využívá klíč dlouhý 128 bitů.

Klíč komunikace protokolu SAUP se nepřenáší po síti. Na straně klienta je vygenerován ze zadaného uživatelského hesla. Na straně serveru je získán od AS(S).

Generování klíče na straně klienta:

1. Ze zadaného hesla se vypočítá SHA-1 hash. Výsledkem je 160ti bitový kód.
2. Vypočítaný kód se zkrátí useknutím pravých 32 bitů na celkovou délku 128 bitů. Výsledkem je šifrovací klíč k_{AB}.

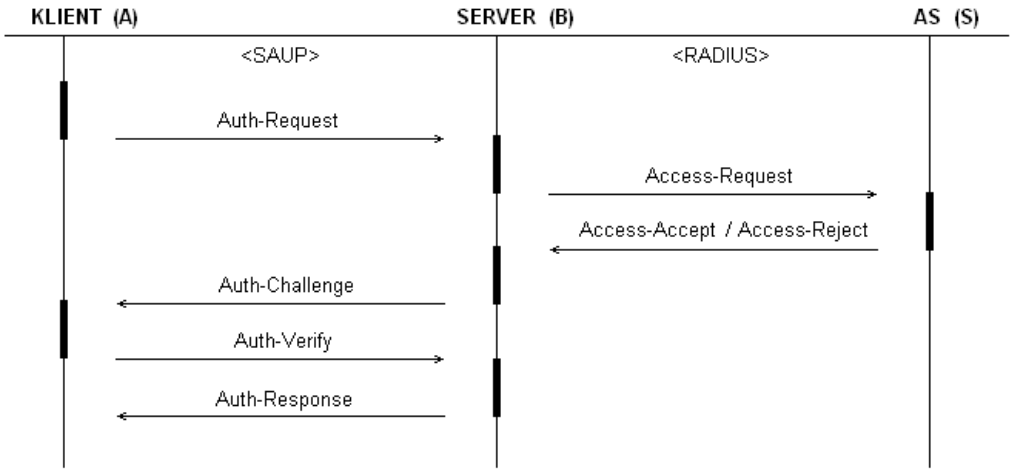


Získání klíče na straně serveru:

1. Server pošle dotaz na AS(S) s autentizačními údaji: *login*, *service*, *domain*, $passwd_{BS}$ (jednotlivé položky byly vysvětlený dříve).
2. Pokud server vyhodnotí autentizační údaje jako nesprávné, zašle zpět zprávu Access-Reject. Pokud autentizační údaje souhlasí zašle zprávu Access-Accept. Součástí této zprávy je i atribut s obsahem zašifrovaného klíče $crypt_{BS}(S_{SHA-1})$. S_{SHA-1} je klíč použitý pro šifrování komunikace klient-server. Klíč *BS* pro dešifrování zná pouze server(B). AS(S) tento klíč *BS* znát nemusí (nesmí), protože do jeho databáze byl vložen klíč již zašifrovaný přímo ve formě $crypt_{BS}(S_{SHA-1})$.

7.6 Typy zpráv protokolu SAUP

Protokol SAUP zná čtyři druhy zpráv (viz. obrázek 7.3). Typická komunikace mezi klientem a serverem je znázorněna na obrázku 7.8



Obrázek 7.6: Typy zpráv protokolu SAUP

Auth-Request

Klient posílá požadavek na autentizaci (obrázek 7.3 - zpráva č. 1). Nese informace o uživatelském jméně a zašifrované náhodné číslo.

Auth-Challenge

Server zjistí u AS šifrovací klíč pro daného uživatele. Rozšifruje přijaté náhodné číslo, přidá k němu své vygenerované náhodné číslo, zašifruje společným sdíleným klíčem a odešle zpět klientovi (obrázek 7.3 - zpráva č. 4)

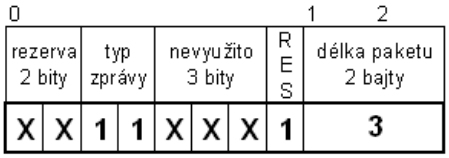
Auth-Verify

Klient rozšifruje zprávu a zašle serveru získané náhodné číslo v otevřené podobě (obrázek 7.3 - zpráva č. 5)

Auth-Response

Server zasílá klientovi výsledek autentizace (obrázek 7.3 - zpráva č. 6). Příznak `RES` je použit pro informaci o výsledku autentizace. V případě chybné autentizace je součástí Auth-Response i atribut nesoucí informaci pro uživatele o důvodu chybné autentizace.

RES - 1 ... autentizace úspěšná
RES - 0 ... autentizace neúspěšná



Obrázek 7.7: Úspěšná autentizace

Obrázek 7.8: Neúspěšná autentizace

7.7 Protokol Radius

Protokol Radius je definován v RFC 2865 [14] a je využitý pro komunikaci server-AS(S). Jediné, co je zde nutné zmínit, je definice přenášených atributů.

Jelikož jsem pro přenos některých autentizačních informací (*service*, *domain*, $crypt_{BS}(S_{SHA-1})$) nevyužil předdefinované atributy protokolu Radius, nadefinoval jsem si proto své vlastní.

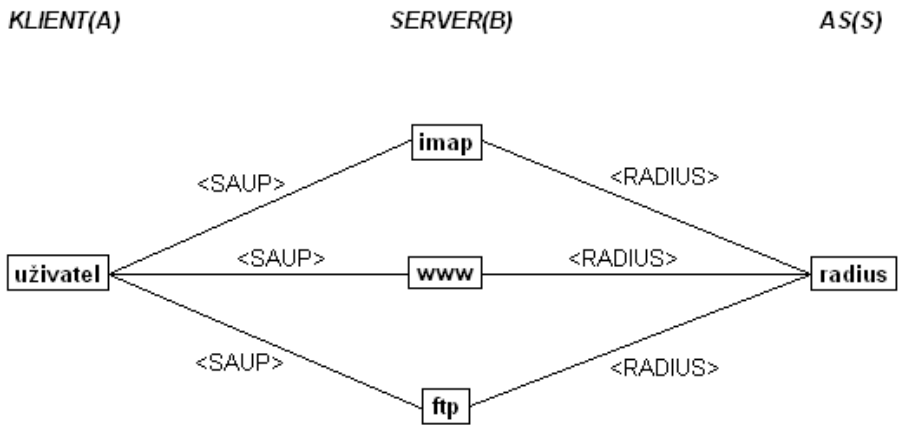
Proměnná	Atribut	Typ
service	Service	string
domain	Domain	string
$crypt_{BS}(S_{SHA-1})$	Service-User-Password	string

Předdefinované atributy *User-Name* a *User-Password* byly použity pro přenos proměnné *login* a $passwd_{BS}$.

7.8 Model systému v praxi

Jak bude vypadat model systému pro jednu službu bylo zmíněno v předchozích odstavcích. Nyní nastíním model systému se zapojením více služeb.

Uživatel potřebuje přistupovat k autorizované části webového serveru, imap serveru a ftp serveru. Zapojení systému je ukázáno na následujícím obrázku:



Obrázek 7.9: Zapojení systémů

Rozmístění důvěrných informací (klíčů a hesel):

- klient (uživatel)
  - login - uživatelské jméno
  - heslo - z hesla se vygeneruje klíč key_{AB} pro šifrování komunikace

- server (služba imap, www, ftp)
  - heslo$_{BS}$ - heslo pro přístup k Radius serveru (pro každou službu jiné). Slouží pro autentizaci služby vůči serveru Radius (standardní použití).
  - klíč$_{kBS}$ - klíč pro dešifrování uživatelského hesla $crypt_{kBS}(key_{AB})$ získaného z Radius serveru
- AS (S)
  - heslo$_{BS}$ - každé službě přísluší jedno heslo (standardně používaná hesla pro server Radius)
  - crypt$_{kB}$(key$_{AB}$) - zašifrovaný klíč $crypt_{kB}(key_{AB})$. Posílá se službě, která ho dešifruje pomocí kB a použije pro šifrování komunikace s klientem.

Způsoby šifrování komunikace:

- klient(A)-server(B) — pro šifrování je použit SHA-1 hash z uživatelského hesla
- server(B)-AS(S) — komunikace není zabezpečena. Pouze pro předání hesla serveru Radius je použit jeho MD5 součet. Přenášené heslo $crypt_{kBS}(key_{AB})$ je uložené na straně Radius serveru již zašifrované pomocí klíče kBS, tudíž je nečitelné.

Kapitola 8

Implementace

V této kapitole bude popsán způsob implementace modelu navrženého schématu autentizace. Celý proces autentizace bude implementován do aplikací `pine` a `imapd`. Tyto aplikace jsou napsány pro systém unix v programovacím jazyce C.

8.1 Použitý software

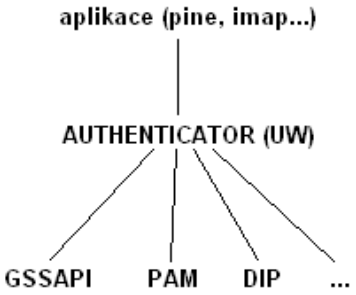
Použité nástroje pro implementaci byly dostatečně platformě přenositelné, proto jsem nevyužil globální síť PlanetLab ale implementoval jsem systém v klasické TCP/IP síti. Zdroje balíčků jsou uvedeny v sekci Příloha A.

Celý systém jsem implementoval na virtuálním stroji VMware Player s operačním systémem Fedora6. Jako emailový server byl použitý `imapd` z balíčku imap-2006e vytvořený na University of Washington. Emailový klient byl `pine` z balíčku pine4.64. Pro autentizační a autorizační server byl použit FreeRadius. Pro komunikaci se serverem Radius jsem použil a upravil knihovnu RadiusClient. Pro šifrování komunikace byla využita knihovna openssl.

8.2 Úprava software

Před samotnou implementací autentizačního mechanismu bylo nutné rozhodnout, do které vrstvy bude mechanismus přidán. Případně který stávající mechanismus (například GSS-API) bude pozměněn, pokud by přidání nové metody nebylo možné.

Po prozkoumání vrstvy AUTHENTICATOR jsem se rozhodl, že stávající mechanismy nechám beze změny a pouze přidám nový. Tím budou všechny způsoby autentizace stále dostupné. Po přidání mechanismu bude struktura autentizačních vrstev oproti nezměněné verzi (obrázek 6.1) rozšířena o autentizaci DIP (obrázek 8.1). DIP je právě typ autentizace využívající protokol SAUP.



Obrázek 8.1: Nová hierarchie autentizačních vrstev

Přidání nové autentizační metody

Následující úpravy budou probíhat ve zdrojových kódech balíčků imap-2006e a pine4.64. Aplikace `imapd` i `pine` používají pro metody autentizace společné rozhraní AUTHENTICATOR. Zdrojové kódy této vrstvy jsou společné pro obě aplikace, proto bude stačit, když postup úprav zmíním jen jednou pro balíček imap-2006e. Bude to ovšem myšleno tak, že ty samé úpravy jsou potřeba i na straně balíčku pine4.64. Přidávaná metoda má název DIP, proto toto slovo bude vkládáno všude tam, kde bude jméno metody vyžadované.

Je nutné vytvořit v adresáři imap-2006e následující soubory:
`src/c-client/auth_dip.c`
`src/osdep/unix/ckp_dip.c`

Oba soubory jsem vytvořil jako kopie těchto souborů využitých pro metodu GSS (auth_gss.c, ckp_gss.c).

Soubor **ckp_gss.c** slouží jen pro kontrolu hesla uživatele, před jeho samotným použitím při autentizaci. V mém případě žádná počáteční kontrola hesla neexistuje, proto jsem tento soubor výrazně zjednodušil.

V souboru **auth_dip.c** je uložen celý autentizační mechanismus. Tento soubor obsahuje definici jednotlivých funkcí rozhraní AUTHENTICATOR (viz. 6.4).

- `auth_dip_valid` - je prázdná - žádná kontrola před samotnou autentizací neprobíhá
- `auth_dip_client` - zajišťuje autentizaci klienta
- `auth_dip_server` - je volána na straně serveru. V této funkci je definována i komunikace mezi serverem(B) a AS(Radius). Pro komunikaci pomocí protokolu Radius jsem použil knihovnu RadiusClient, kterou jsem upravil pro potřeby nového autentizačního mechanismu.

Detailněji nebudu dále implementaci těchto dvou souborů popisovat. Přesné informace jsou k nalezení v komentářích ve zdrojových kódech.

Bylo nutné přidat nový mechanismus do kompilace aplikace - úprava souboru:
`imap-2006e/src/osdep/unix/Makefile:`

řádek 892: přidat kompilaci souboru `auth_dip.c`
`auth_ext.c auth_gss.c auth_log.c auth_md5.c auth_pla.c auth_dip.c`

řádek 989: za cíl `ckpgss` přidat cíl `ckpdip`:

```
ckpdip:
    @echo Authentication DIP
    $(LN) ckp_dip.c osdepckp.c
```

V tomto stavu jsou aplikace schopny autentizace pomocí mechanismu DIP. Zbývá pouze tento mechanismus nastavit jako požadovaný. Autentizační mechanismus si definuje server `imapd`, klientská aplikace pine vybere mechanismus podle požadavků serveru. U serveru `imapd` se volba autentizačního mechanismu neprovádí v žádném konfiguračním souboru nýbrž je dána parametry definovanými v souboru Makefile. Server imapd tudíž nepodporuje změnu mechanismu bez nutnosti kompilace.

Nastavení autentizačního mechanismu:
`imap-2006e/Makefile:`

řádek 165: nastavit způsob autentizace
`PASSWDTYPE=dip`

Nyní je autentizační metoda přidána a je nastaveno její použití.

8.3 Knihovna libdip_service.so

Autentizační mechanismus je uložen v souboru `auth_dip.c`. Bylo by ale krajně nevhodné do tohoto souboru psát definice všech podpůrných funkcí. Pokud ještě vezmeme z úvahu to, že součástí mechanismu je komunikace protokolem Radius, bylo by množství programového kódu neúnosné. Proto jsem vytvořil knihovnu libdip_service.so, která veškeré podpůrné operace zajišťuje.

Její součástí je upravená implementace RadiusClient (Příloha A) pro komunikaci protokolem Radius. Obsahuje také další funkce potřebné pro autentizaci jako šifrování a dešifrování zprávy nebo ověření uživatelského jména a hesla.

Rozhraní knihovny je navrženo tak, že pokud nebude udělána významná změna do schématu komunikace protokolem SAUP, nebude nutné znovu kompilovat celou aplikaci, ale bude stačit aktualizace sdílené knihovny libdip_service.so.

Jednou z takových úprav může být například změna šifrovacího algoritmu nebo změna délky šifrovacího klíče.

Rozhraní knihovny libdip_service.so:
Knihovna libdip_service.so poskytuje funkce přímo ekvivalentní zprávám SAUP protokolu.

`ATTRIBUTE * dip_auth_request( char * login, char * passwd )`
`login` - uživatelské jméno
`passwd` - heslo
funkce vrací přímo datagram protokolu SAUP, který se posílá po síti

`ATTRIBUTE * dip_auth_challenge( ATTRIBUTE * PDU )`
`PDU` - datagram, který byl vytvořen funkcí `dip_auth_request`
funkce vrací přímo datagram protokolu SAUP, který se posílá po síti

`ATTRIBUTE * dip_auth_verify( ATTRIBUTE * PDU )`
`PDU` - datagram, který byl vytvořen funkcí `dip_auth_challenge`
funkce vrací přímo datagram protokolu SAUP, který se posílá po síti

`ATTRIBUTE * dip_auth_response( ATTRIBUTE * PDU )`
`PDU` - datagram, který byl vytvořen funkcí `dip_auth_verify`
funkce vrací přímo datagram protokolu SAUP, který se posílá po síti

`int get_response ( ATTRIBUTE * PDU )`
`PDU` - datagram, který byl vytvořen funkcí `dip_auth_response`
funkce vrací hodnotu:
0 - pokud uživatel nebyl autentizován
1 - pokud uživatel byl úspěšně autentizován

8.4 Instalace

Chceme-li používat autentizaci pomocí metody DIP pro emailový server `imapd` a klient `pine`, je nutné nainstalovat následující balíčky (přesný popis instalace je popsán v příloze A).

poštovní server postfix
postfix-2.3.3-2
pcre-6.6-1.1
maildop-2.0.3

imap server
xinetd-2.3.14-8
imap-2006e-dip – upravený balíček (součástí přiloženého CD-R, instalace podle přílohy)

emailový klient pine
pine-4.64-dip – upravený balíček (součástí přiloženého CD-R, instalace podle přílohy)

knihovna libdip_service.so
dip-1.0

Postup instalace knihovny libdip_service.so:
`$ cd dip`
`$ make`
`$ make install`

Radius server
freeradius-1.1.4

Postup instalace:
`$ yum install freeradius`
spuštění skriptu pro upravení nastavení serveru Radius. Přizpůsobení pro autentizaci pomocí autentizační metody DIP
`$ ./update_radius.sh` - skript je součástí přiloženého CD-R

8.5 Konfigurace

Konfigurace systému se dá rozdělit na tři části. Konfigurace klienta, služby a Radius serveru.

Klient

Na straně klienta není potřeba nic nastavovat. Klient jen vyplní v přihlašovacím formuláři programu `pine` uživatelské jméno a heslo.

Služba - server(B)

Na serveru, kde běží služba (např. imap), musí být nakonfigurovaný radius-klient.

Nejdříve je nutné nastavit přístupová práva k adresáři `/usr/local/etc/raddb` a jeho souborům. Přístup musí být udělen pouze uživateli, pod kterým běží služba (např. imap), protože v souborech jsou uložena hesla pro přístup k Radius serveru (tento způsob uchování tajné informace je u serveru Radius standardní).

```
$ chown -R imap /usr/local/etc/raddb    # případná změna vlastníka souboru
$ chmod -R 600  /usr/local/etc/raddb    # změna přístupových práv
```

```
$ vim /usr/local/etc/raddb/server
    # sdílené heslo mezi radius-klientem(imap) a radius-serverem(AS).
    localhost testing123
```

```
$ vim /usr/local/etc/raddb/radiusclient.conf
    #  adresa Radius serveru
    authserver   localhost

    #  klíčBS pro dešifrování uživatelského klíčeAB
    #  (který se používá ke komunikaci klient-server)
    service_radius_passwd   klicBS
```

Radius - AS(S)

Nastavení Radius serveru. Nejprve je potřeba nastavit adresu z které se služba (imap) bude přihlašovat. Toto se nastavuje v souboru `/etc/raddb/clients.conf`. Editace uživatelů se musí provádět ručně v souboru `/etc/raddb/users`.

```
$ vim /etc/raddb/clients.conf
    client 127.0.0.1 {            # adresa serveru, kde běží služba
          secret    = testing123  # sdílené heslo mezi radius-klientem a
                                  # radius-serverem
          shortname = localhost   # interní pojmenování adresy
    }
```

Nastavení uživatele *username*: Nastavení se skládá ze dvou částí. První je autentizační část (první dva řádky). Je to podmínka, kterou musí klient splnit (zadat shodné údaje). Po posled-ním argumentu bez čárky (v tomto případě je poslední argument `Domain == "localhost"`) následuje odpověď klientovi. Zde je nastavené zašifrované heslo pro komunikaci klient(pine)-server(imap). Toto heslo se získá pomocí programu `genDipKey`, který je součástí instalace knihovny libdip_service.so. Heslo se vygeneruje na straně serveru(imap) a uloží se do tohoto souboru. Heslo je ve formě hexa znaků.

```
username Auth-Type := Local, User-Password == ''klicBS'',
          Service == ''IMAP'', Domain == ''localhost''
              Reply-Message = ''HELLO %u'',
              Service-User-Password = ''00112233445566778899aabbccddeeff''
```

Kapitola 9

Bezpečnost systému

Míra bezpečnosti systému je dána jeho nejslabší částí. Proto je nutné brát v úvahu jak komunikační protokol, tak i jeho implementaci. Neboť chyba v implementaci zabezpečeného protokolu může útočníkovi umožnit přístup k systému, nebo krádež důvěrných informací, ať už uživatelských dat nebo přístupových hesel.

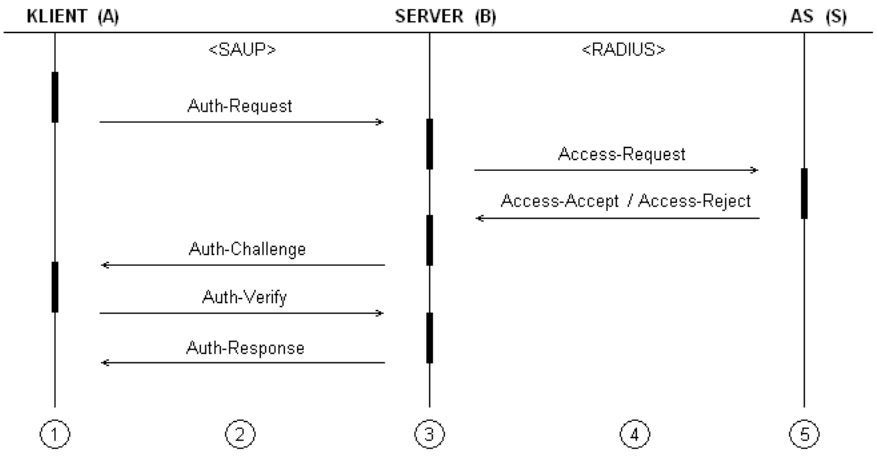
9.1 Bezpečnost protokolu SAUP

Zkoumal jsem základní možné útoky, které mohou být na protokol provedeny a nenalezl jsem jedinou možnost, kdy by útočník mohl z komunikace odposlechnout heslo nebo mu bylo umožněno neoprávněné autentizace.

Neuvažoval jsem situace, kdy útočník pozmění kód klientské aplikace, který mu umožní získat uživatelské jméno a heslo. Tento útok by byl totiž možný na jakýkoliv systém. Dále jsem také počítal s tím, že použité šifrovací algoritmy jsou bezpečné.

Při analýze bezpečnosti šlo pouze o bezpečnost navrhnutého komunikačního protokolu.

Uvažoval jsem následující možnosti:



Obrázek 9.1: Místa možného útoku

1. Snaha o prolomení na straně klienta. Na straně klienta nejsou uložena žádná důvěrná data. Pokud se útočník bude vydávat za uživatele, pokusí se přihlásit jeho uživatelským jménem, nepodaří se mu zjistit heslo, protože server(B) rozšifruje náhodné číslo nonceA jako jiné náhodné číslo nonceA' a to vrátí zpět zašifrované. Proto je zde možný pouze útok silou na šifrovací algoritmus.

2. Útok "Man In The Middle". Tento útok je nemožný bez znalosti šifrovacího klíče. Útočník má téměř ty samé možnosti jako v prvním případě.

3. Snaha o prolomení na straně serveru(B). Útočník se vydává za falešný server(B). Při komunikaci s klientem má útočník prakticky stejné šance jako v situaci, když se vydával za klienta.

   Při komunikaci s AS(S) musí zadat heslo, které zná "pravý" server(B). I když by se mu to povedlo, tak získá zašifrovaný klíč $crypt_{BS}(S_{SHA-1})$ pro komunikaci s klientem. Dešifrování tohoto klíče komunikace bez znalosti klíče BS vede zase jen k útoku silou.

   Nebezpečí by bylo, pokud by útočník získal neoprávněný přístup k serveru(B) skrze uživatele, pod kterým napadená služba běží. Mohl by pak získat šifrovací klíč ke komunikaci jak ke klientovi tak i s AS(Radius). Tato možnost ale není otázkou bezpečnosti protokolu, ale spíše bezpečnosti systému, na kterém služba běží.

4. Útok "Man In The Middle". Útočník může odposlechnout a pozměnit zašifrovanou verzi klíče $crypt_{BS}(S_{SHA-1})$. Toto povede k chybné autentizaci uživatele, ale neumožní útočníkovi získat důvěrná data.

5. Snaha o prolomení na straně AS(S). Útočník má téměř stejné šance jako v předchozím případě

9.2 Kvalita implementace

Při implementaci autentizačního mechanismu a knihovny libdip_service.so jsem se snažil vyvarovat chyb vedoucích k prolomení přístupu k systému.

Nejnebezpečnější místo komunikace celého systému se tak stává server(B). Na serveru(B) je uložené přístupové heslo k AS v konfiguračním souboru, který je přístupný pouze uživateli, pod kterým služba běží. V tom samém souboru je také uložen klíč $_{BS}$, který se používá pro dešifrování klíče komunikace klient-server. Pokud by došlo k chybné konfiguraci práv k tomuto souboru, mohl by útočník, který by měl přístup na server(B), získat klíče komunikace a důvěrná data uživatelů.

Kapitola 10

Závěr

Předmětem diplomové práce byl návrh modelu autentizace a autorizace uživatele pomocí důvěryhodného serveru.

Nejprve jsem zde uvedl teoretický základ vysvětlující danou problematiku. Analyzoval jsem systém autentizace Kerberos a ze získaných informací jsem vycházel při návrhu autentizačního mechanismu. Výsledkem návrhu bylo schéma komunikace klient - server(služba) - autentizační server. Pro komunikaci mezi klientem a serverem(službou) jsem vytvořil nový protokol SAUP (Single Authentication Protocol), pro předávání dat mezi serverem(službou) a autentizačním serverem byl použitý protokol Radius. Navržený model autentizace jsem úspěšně implementoval do aplikací emailového klienta `pine` a poštovního serveru `imap`.

Při návrhu a implementaci modelu systému byl kladen velký důraz na zabezpečení přenosu autentizačních informací. V poslední kapitole jsem rozebíral různé druhy útoků na protokol SAUP a neshledal jsem žádné kritické místo, kde by útočník mohl odposlechnout a rozluštit přenášené informace.

Navržený model slouží jako základ autentizačního mechanismu. Existuje mnoho možností, které by mohly rozšířit jeho funkčnost. Mohlo by se jednat a širší podporu šifrovacích algoritmů, schopnost jednat s více druhy autentizačních serverů (např. ldap), vyhledání sekundárního autentizačního serveru při nedostupnosti primárního. Asi největší inovací by byl přechod na dynamickou změnu hesel použitých pro šifrování komunikace. Tato varianta by ovšem znamenala zásah i na straně klienta. Musela by se přidat nová vrstva mezi uživatele a klientskou aplikaci, která by zajišťovala konverzi uživatelského hesla na dynamické heslo pro šifrování komunikace.

Literatura

[1] Libor Dostálek a kolektiv. *Velký průvodce protokoly TCP/IP: Bezpečnost*. Computer Press, 2003.

[2] B. Aboba and J. Wood. RFC 3539: Authentication, Authorization and Accounting (AAA) Transport Profile, 2003.

[3] L. Blunk and J. Vollbrecht. RFC 2284: PPP Extensible Authentication Protocol (EAP), 1998.

[4] Radim Bártů. Semestrální práce z předmětu KIV/Přenos dat. `http://www.kiv.zcu.cz/~simekm/vyuka/pd/zapocty-2004/802_1x-bartu/index.php`, 2005.

[5] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. RFC 3588: Diameter Base Protocol, 2003.

[6] P. Congdon, B. Aboba, A. Smith, G. Zorn, and J. Roese. RFC 3580: IEEE 802.1X Remote Authentication Dial In User Service (RADIUS), 2003.

[7] G. De Laet and G. Schauwers. *Network Security Fundamentals*. Cisco Press, 2005.

[8] Companion Guide. *Fundamentals of Network Security*. Cisco Press, 2004.

[9] J. Kohl and C. Neuman. RFC 1510: The Kerberos Network Authentication Service (V5). `ftp://ftp.rfc-editor.org/in-notes/rfc1510.txt`, 1993.

[10] Jakub Mahdal. Kerberos, PAM. `http://www.fi.muni.cz/~kas/p090/referaty/2005-podzim/st/kerberos_pam.html`, 2005.

[11] C. Rigney. Rfc 2059: Radius accounting, 1997.

[12] C. Rigney. RFC 2866: RADIUS Accounting, 2000.

[13] C. Rigney, A. Rubens, W. Simpson, and S. Willens. Rfc 2058: Remote authentication dial in user service (radius), 1997.

[14] C. Rigney, S. Willens, A. Rubens, and W. Simpson. RFC 2865: Remote Authentication Dial In User Service (RADIUS), 2000.

[15] W. Simpson. RFC 1661: Point-to-Point Protocol, 1994.

[16] J. Wray. RFC 2744: Generic Security Service API Version 2: C-bindings, 2003.

Dodatek A

Instalace

Zdroje

V diplomové práci byly uvedeny odkazy na následující zdroje:

imap2006e
ftp://ftp.cac.washington.edu/imap/old/imap-2006e.tar.Z

pine4.64
ftp://ftp.cac.washington.edu/pine/pine.tar.gz

VMware Player
http://www.vmware.com/download/player/

Fedora6
http://www.thoughtpolice.co.uk/vmware/http_download.php?file=fedora-fc6-i386.zip

FreeRadius
ftp://ftp.freeradius.org/pub/radius/old/freeradius-1.1.4.tar.gz

RadiusClient
ftp://ftp.cityline.net/pub/radiusclient/radiusclient-0.3.2.tar.gz

openssl
http://www.openssl.org/source/openssl-0.9.8e.tar.gz

Popis instalace na operační systém Fedora6 (Příloha A). Operační systém byl spuštěn v aplikaci VMware Player.

Instalace

Instalace překladače GCC - GNU Compiler Collection

```
$ yum install gcc
$ yum install gcc-c++.i386
```

Kerberos a požadované knihovny nebo aplikace

```
$ wget http://www.openssl.org/source/openssl-0.9.8e.tar.gz
$ gunzip openssl-0.9.8e.tar.gz
$ tar xvf openssl-0.9.8e.tar
$ ./config
$ make
$ su: make install

$ wget http://ftp.gnu.org/pub/gnu/ncurses/ncurses-5.5.tar.gz
$ gunzip ncurses-5.5.tar.gz
$ tar xvf ncurses-5.5.tar
$ ./configure
$ make
$ su: make install

$ yum install byacc
$ yum install flex

$ wget http://download.talinux.tal.org/pub/talinux/sources-current-unstable
      /security/libsepol/libsepol-1.14.tgz
$ tar xvf libsepol-1.14.tgz
$ make
$ su: make install
```

```
$ wget http://www.nsa.gov/selinux/archives/checkpolicy-1.34.1.tgz
$ tar xvf checkpolicy-1.34.1.tgz
$ make
$ su: make install

$ wget http://www.nsa.gov/selinux/archives/libselinux-1.34.7.tgz
$ tar xvf libselinux-1.34.7.tgz
$ make
$ su: make install

$ wget http://www.kernel.org/pub/linux/libs/pam/pre/library
      /Linux-PAM-0.99.7.1.tar.bz2
$ tar xvf Linux-PAM-0.99.7.1.tar.bz2
$ ./configure
$ make
$ su: make install (s verzí 0.99.4.0  nešlo přeložit)


$ wget http://web.mit.edu/Kerberos/dist/krb5/1.6/krb5-1.6.1-signed.tar
$ tar xvf krb5-1.6.1-signed.tar
$ gunzip krb5-1.6.1.tar.gz
$ tar xvf krb5-1.6.1.tar
$ ./configure
$ make
$ su: make install
```

IMAP server

```
$ wget ftp://ftp.cac.washington.edu/imap/old/imap-2006e.tar.Z
$ zcat imap.tar.Z | tar xvf -
$ ln -s /usr/local/ssl/lib/libcrypto.a /usr/lib/libcrypto.a
$ make slx

$ vim ~/.bashrc
   export  LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib

$ yum install xinetd.i386
```

Postfix

```
$ yum remove sendmail
$ yum install postfix
```

```
$ ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-6.6.tar.gz
$ ./configure
$ make
$ su: make install
```

stáhnout `maildrop` z: http://sourceforge.net/project/downloading.php?group_id=5404&use_mirror=kent&filename=maildrop-2.0.3.tar.bz2 (verze 2.0.4 nešla nainstalovat)

```
$ ./configure
$ make
$ su: make install
```

Pine - emailový klient

```
$ wget ftp://ftp.cac.washington.edu/pine/pine.tar.gz
$ gunzip pine.tar.gz
$ tar xvf pine.tar
$ cd pine4.64
```

Před kompilací `pine` je nutné udělat pár změn v adresáři se zdrojovými kódy aplikace `pine`

Vytvoření symbolického linku na zdrojové soubory Kerbera
`$ ln -s /CESTA_K_SYSTEMU_KERBEROS/krb5-1.6.1/src krb5`

Zkopírování původních certifikátů openssl ze zdrojových kódů do adresáře s certifikáty operačního systému

```
$su: cp -R /CESTA_KE_ZDROJOVÝM_KÓDŮM_OPENSSL/certs/*
        /usr/local/ssl/certs
```

Kompilace aplikace
`./build slx`

```
/home/user/.bashrc:
  export  LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

Dodatek B

Konfigurace postfix, imap, Kerberos, pine

Kerberos
Úprava konfiguračních souborů
`$ vim /etc/krb5.conf`

```
[logging]
  default = FILE:/var/log/krb5libs.log
  kdc = FILE:/var/log/krb5kdc.log
  admin_server = FILE:/var/log/kadmind.log

[libdefaults]
  default_realm = REALM

[realms]
  REALM = {
    kdc = domena.cz:88      - musí být jméno, které NEUKAZUJE na 127.0.0.1
                              nebo "localhost" (funguje např. 192.168.2.5)
    admin_server = domena.cz:750
    database_name = /usr/local/var/krb5kdc/principal
    admin_keytab = FILE:/usr/local/var/krb5kdc/kadm5.keytab
    acl_file = /usr/local/var/krb5kdc/kadm5.acl
    key_stash_file = /usr/local/var/krb5kdc/.k5.REALM
    kdc_ports = 750,88
 }
```

`[domain_realm]` - převádění doménových jmen na realmy, pokud nejsou explicitně zadané

```
  .domena.cz = REALM
  domena.cz = REALM
  .localhost = REALM
  localhost = REALM
```

```
[kdcdefaults]
  acl_file = /usr/local/var/krb5kdc/kadm5.acl
  admin_keytab = /usr/local/var/krb5kdc/kadm5.keytab
  v4_mode = nopreauth
```

Vytvoření souborů, ve kterých budou uloženy zašifrované klíče a tikety. Nastavení uživatelů, kteří budou mít přístup k administrátorským funkcím systému Kerberos.

```
$ touch /usr/local/var/krb5kdc/kadm5.keytab
$ touch /usr/local/var/krb5kdc/kadm5.acl
```

Vytvoření databáze Kerbera

```
$ /usr/local/sbin/kdb5_util -s create
```

(Zrušení databáze se provede příkazem)

```
($ /usr/local/sbin/kdb5_util destroy)
```

Spuštění serveru Kerbera

```
$ /usr/local/sbin/krb5kdc
```

Přidání uživatele do systému Kerberos

```
$ /usr/local/sbin/kadmin.local
  > addprinc username
```

Přidání služby do systému Kerberos

```
$ /usr/local/sbin/kadmin.local
  > addprinc -randkey imap
  > ktadd imap
```

Přihlášení uživatele

```
$ kinit username
```

Postfix - poštovní server

Úprava hostname a domainname systému:

```
$ vim /etc/hosts
  192.168.2.5  username domena.cz

$ domainname domena.cz
$ hostname username
```

Konfigurace postfixu:

```
$ vim /etc/postfix/master.cf - doručování a filtrování pošty
   maildrop  unix  -       n       n       -       -       pipe
     flags=DRhu user=vmail argv=/usr/local/bin/maildrop -d ${recipient}

$ /etc/postfix/main.cf:
    mail_owner = postfix
```

```
    myhostname = domena.cz
    inet_interfaces = localhost
    #mailbox_command = maildrop    - musí být zakomentované
    virtual_transport = maildrop
    maildrop_destination_recipient_limit = 1
```

Nastavení aliasů (uživatel, který má dostávat emaily určené pro uživatele `root`)

```
$ vim /etc/aliases root:    username

$ newaliases
```

Po změně konfigurace restart služby

```
$ /sbin/service postfix restart
```

Imap server

Imap server neobsahuje žádné konfigurační soubory. Jeho nastavení se provádí pouze pomocí parametrů při kompilaci.
Před samotnou kompilací bylo nutné provést následující změny.

```
$ vim ./Makefile
    EXTRAAUTHENTICATORS=gss
    PASSWDTYPE=gss

    lfd:
    SPECIALS=‘‘SSLINCLUDE=/usr/local/ssl/include
              SSLLIB=/usr/local/ssl/lib ...’’


$ vim ./src/osdep/unix/Makefile
    SSLINCLUDE=$(SSLDIR)/include

$ vim ./src/osdep/unix/Makefile.gss
    GSSINCLUDE=/usr/local/include/gssapi

$ ln -s /usr/local/ssl/lib/libcrypto.a /usr/lib/libcrypto.a
```

Při překladu bylo nutné zapnout podporu pro IPv6, jinak se překlad nezdařil

```
$ make lfd
   (ipv6 support: YES)
```

Bylo nutné nastavit proměnnou prostředí `LD_LIBRARY_PATH`

```
$ vim ~/.bashrc
     export  LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

Po úspěšné kompilaci serveru zbývá nastavit xinetd démona, který bude spouštět imap server. Je nutné proto vytvořit soubory `/etc/xined.d/imap` a `/etc/xined.d/imaps`, které budou mít následující obsah.

```
$ vim /etc/xined.d/imap(s)
    service imap(s)
    {
      socket_type = stream
      wait = no
      user = root
      server = /cesta_k_souboru_imapd     (/usr/sbin/imapd)
      log_on_success += HOST DURATION
      log_on_failure += HOST
      disable = no
    }
```

Konfigurace xinetd démona a imap serveru je u konce, zbývá jen restartovat xinetd démona
`$ /sbin/service xinetd restart`

Pine - emailový klient
U emailového klienta `pine` nebylo třeba děla žádné změny v konfiguraci.

Dodatek C

GSSAPI Autentizace

```
pine:  auth_gssapi_valid
           gss_import_name  // name = unknown@REALM
           gss_release_name

imap:  auth_gssapi_valid
           gss_import_name  // name = imap@REALM
           gss_release_name

imap:  auth_gssapi_server
           gss_import_name  // name = imap@REALM
           gss_acquire_cred

pine:  auth_gssapi_client
           gss_import_name  // name = imap@REALM
           gss_init_sec_context

imap:      gss_accept_sec_context

pine:      gss_release_buffer
           gss_init_sec_context

imap:      gss_release_buffer
           gss_display_name
           gss_wrap
           gss_seal

pine:      gss_unwrap
           gss_unseal
```

```
            gss_release_buffer
            gss_wrap
            gss_seal
            gss_release_buffer
            gss_delete_sec_context
            gss_release_name

imap:       gss_release_buffer
            gss_unwrap
            gss_unseal
            gss_release_buffer
            gss_release_buffer
            gss_release_name
            gss_delete_sec_context
            gss_release_cred
            gss_release_buffer
            gss_release_name
```