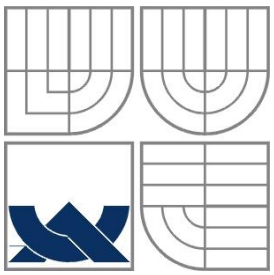
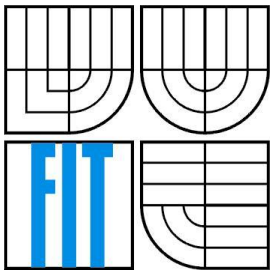


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

LICENČNÍ SERVER
LICENSE SERVER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN FRÜHBAUER
AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. Tomáš Hruška, CSc.
SUPERVISOR

BRNO 2010

Abstrakt

Cílem bakalářské práce bylo vytvořit licenční ochranu produktů projektu Lissom.
Teoretická část je určena pro seznámení uživatele s problematikou licencování a ochrany softwaru. Jsou zde popsány právní ochrany vztahující se k softwaru, způsoby licencování, druhy technologických ochran softwaru a možnosti jejich překonání. Hlavní důraz je kladen na ochranu softwaru pomocí licenčního serveru.
Praktická část se věnuje návrhu a implementaci nástroje pro správu databáze licencí, dále programu pro vytváření licenčních souborů a nakonec knihovny pro práci s těmito soubory.

Abstract

The aim of this Bachelor thesis is to create licensing protection to Lissom project products.
Theoretical part introduces reader into software licensing and protection. There are describes possibilities of legal protection, licensing and ways of breaking it. Main aim of this part is to protect software using licensing server.
Practical part is dedicated to implementation license database management tool, program creating license files and finally a library enabling using of these files.

Klíčová slova

Duševní vlastnictví, kategorie softwaru, licenční klíč, hardwarový klíč, licenční server, softwarové pirátství, licenční manažer, vendor daemon, licenční soubor

Keywords

Intellectual property, category of software, license key, hardware key, license server, software piracy, license manager, vendor daemon, license file

Citace

Frühbauer Jan: Licenční server, bakalářská práce, Brno, FIT VUT v Brně, 2010

Licenční server

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením prof. Ing. Tomáše Hrušky, CSc.
Další informace mi poskytli Ing. Boris Šuška a Ing. Karel Masařík, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Frühbauer
19.5.2010

Poděkování

Děkuji vedoucímu práce prof. Ing. Tomáši Hruškovi, CSc. a konzultantům Ing. Borisi Šuškovi a Ing. Karlu Masaříkovi, Ph.D. za vedení mé bakalářské práce, cenné rady a připomínky a za čas, který mi věnovali.

Obsah

1 Úvod

Zadání této bakalářské práce mi bylo nabídnuto od výzkumné skupiny projektu Lissom. Produkty tohoto projektu jsou totiž připravovány na komerční nasazení, a proto je zapotřebí je chránit proti nelegálnímu používání. Rád jsem toto zadání přijal, protože považuji ochranu softwaru za velmi aktuální téma dnešní doby.

Ve své práci se nejprve věnuji ochraně softwaru z právního hlediska. V další části se zaměřím na způsoby licencování softwaru a popíši jednotlivé kategorie softwaru rozdělené podle toho, jaká práva poskytuje výrobce zákazníkovi v licenci. Dále uvedu možnosti technologické ochrany softwaru a způsoby, kterými lze tyto ochrany překonat. Samostatnou část práce vyhradím pro jeden konkrétní typ ochrany softwaru - licenční server. V této části detailněji popíši jednotlivé části konceptu licenčního serveru, dále pak formát licenčního souboru, který slouží pro uchování informací o licencích, a jaké možnosti nabízí licencování pomocí tohoto typu ochrany. V závěrečné kapitole se budu věnovat praktické části bakalářské práce. Mým úkolem je vytvořit databázi licencí, uživatelské rozhraní pro správu této databáze, generátor licenčních souborů a nakonec knihovnu pro práci s těmito soubory. V závěrečné kapitole tedy popíši návrh a způsob implementace těchto částí. Hlavním požadavkem pro praktickou část bakalářské práce byla bezpečnost.

2 Softwarové právo

Ochrana softwaru spadá do ochrany duševního vlastnictví. Software lze tedy chránit pomocí patentu, obchodního tajemství a copyrightu. Zvláštní oblast ochrany softwaru představuje copyleft, který uvedl projekt GNU.

2.1 Patent

Patent je vydáván pro nový a jakkoli užitečný proces, stroj, výrobek nebo nové vylepšení. Patent dává majiteli právo využívat tento nový vynález. Bez patentu není možné vynález jakkoli využívat. Pokud je však vynález chráněn patentem, neznamená to, že se bude tajit před veřejností. Naopak je vlastník patentu povinen zveřejnit vynález. Nikdo ho sice nemůže využívat, ale každý ho může studovat. Problém nastává v tom, že patent je vydáván na určitou dobu a po této době již vlastník nemá žádný způsob, jak chránit svůj vynález [1].

2.2 Obchodní tajemství

Obchodní tajemství je informace, která má určitou ekonomickou hodnotu a která není zatím všeobecně známá a vlastník této informace nechce, aby se kdokoli k této informaci dostal. Z hlediska softwaru má ochrana obchodním tajemstvím více možností než patent. Nelze tímto způsobem však chránit čistě funkcionální programy, které mají snadno přístupné funkce. Základní rozdíl mezi patentem a obchodním tajemstvím je v tom, že majitel obchodního tajemství chce udržet svou informaci tajnou [1].

2.3 Copyright

Pomocí copyrightu lze chránit pouze díla, která jsou pevně spjata s nějakým hmatatelným médiem. Copyrightem tedy nelze chránit myšlenky, koncepty, principy či jiné nehmatatelné věci. Chrání pouze již samotné vyjádření myšlenek, ne myšlenky samotné. Není tedy možné, aby někdo například kopíroval zdrojový kód softwaru chráněného copyrightem, ale nikdo již takovémuto člověku nemůže zakázat, aby si sám napsal program, který má stejnou funkcionalitu jako chráněný program [1].

2.4 Copyleft

Copyleft byl vymyšlen v projektu GNU. Jedná se o obrácenou verzi copyrightu. Zatímco copyright nepovoluje nikomu manipulovat jakkoli se softwarem, copyleft naopak znamená, že software je naprosto svobodný. V praxi to znamená, že software může být kýmkoli modifikován nebo rozšiřován a nehrozí mu za to žádné postihy. Je zde však jedna podmínka, že jakákoli modifikovaná nebo rozšířená verze musí být také naprosto svobodná. Nikdo tedy nesmí vylepšit software chráněný copyleftem a prohlásit ho za proprietální [2].

2.5 Převod práv na zákazníka

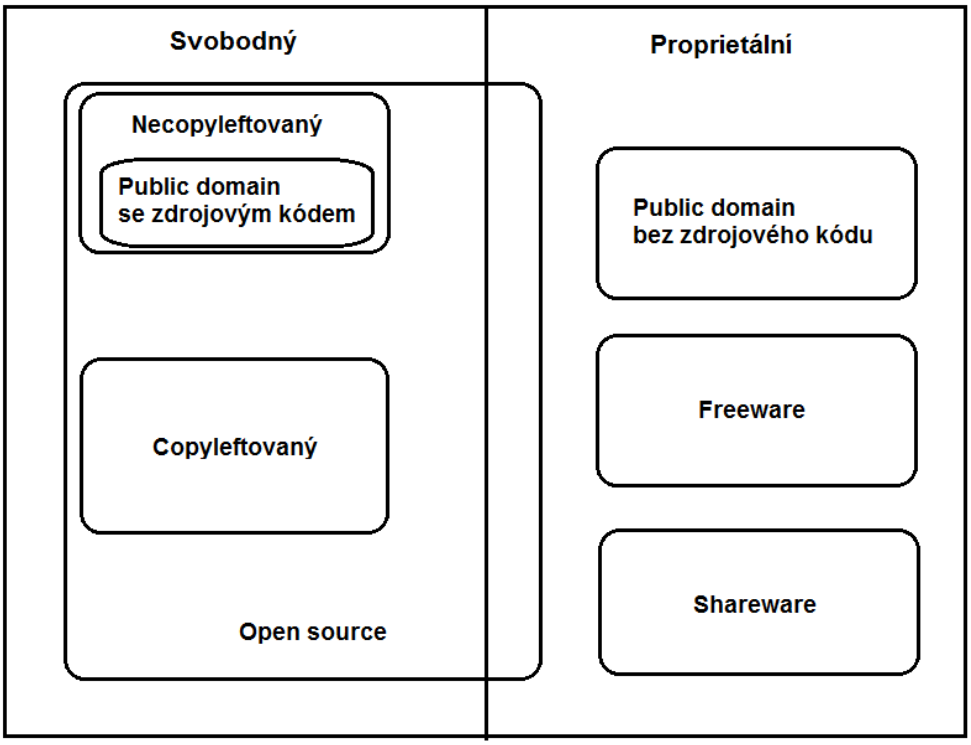
Existují dva způsoby převodu práv na duševní vlastnictví. Buď jde o převod práv, nebo přidělení licence. U převodu práv majitel chráněného majetku předá veškerá svoje práva na daný software a u přidělení licence předá práva, která sám uzná za vhodné.

2.6 Shrnutí

Software lze chránit výše zmíněnými typy ochran. Patent a obchodní tajemství jdou v podstatě proti sobě. Zatímco patent ukládá majiteli zveřejnit předmět patentování, obchodní tajemství striktně tají chráněné informace před veřejností. Proto nelze na jeden software aplikovat obě tyto ochrany společně. Copyright či copyleft chrání až samotný hmatatelný výsledek celé práce a proto nejsou v rozporu s žádnou z předešlých dvou ochran. Díky těmto ochranám vznikl základní koncept licencování softwaru. Vlastník má díky nim určitá práva a pomocí licencí může některá z těchto oprávnění převádět na zákazníka.

3 Kategorie softwaru podle způsobu licencování

Jak bylo uvedeno v předešlé kapitole, tvůrce softwaru má na svůj výrobek určitá práva. Software lze tedy dělit podle toho, jaká práva předává tvůrce svým zákazníkům. Jednotlivé kategorie softwaru a vztah mezi nimi nejlépe vystihuje tento obrázek převzatý z projektu GNU [3].



1 Vztah mezi kategoriemi softwaru

3.1 Svobodný software

Svobodný software je software, který může kdokoliv měnit a dál distribuovat. Neznamená to však, že za takovýto software nemusíme platit. Pokud má být software svobodný, musí pro něj platit čtyři svobody. První svobodou je, že uživatel může software spouštět za jakýmkoli účelem. Další je svoboda software studovat a upravovat ho pro vlastní potřebu, dále svoboda redistribuce kopie softwaru a nakonec svoboda zlepšovat software a tuto vylepšenou verzi rozšiřovat. K tomu, aby byly tyto svobody zaručeny, je nutné, aby měl uživatel přístup ke zdrojovým souborům. K takovémuto softwaru se mohou přidávat licenční pravidla, ale žádné z nich nesmí být v rozporu se čtyřmi základními svobodami. Pokud bude někdo dál distribuovat modifikovanou verzi, nemusí o její existenci nikoho informovat. Někdy se lidé mylně domnívají, že svobodný software musí být

distribuován zadarmo. Není tomu tak. I svobodný software může být zpoplatněn, ovšem takto zpoplatněný software musí dále splňovat základní podmínky svobodného softwaru. V praxi se tedy objevují i komerční svobodné softwary [3].

3.2 Open source software

Název open source software vznikl kvůli nejednoznačnosti označení svobodný software. Veřejnost se domnívala, že slovo svobodný znamená bez poplatku[3]. Software se může zařadit do této kategorie, pokud vyhovuje těmto podmínkám:

- Software musí být volně šiřitelný a nikdo k němu nesmí přidávat omezující licenční podmínky
- Zdrojový kód musí být dostupný v čitelné formě
- Software může kdokoli měnit, ale změněnou formu musí distribuovat pod stejnou licencí jako originální software
- Licence nesmí diskriminovat žádného člověka nebo skupinu lidí
- Software může být použit pro jakýkoli účel
- Při distribuci programu dostává zákazník všechna původní práva k programu
- Pokud je distribuována pouze část softwaru, musí zákazník k této části získat stejná práva, jaká platí pro originální software
- Licence open source softwaru nesmí nijak omezovat žádný jiný software a musí být technologicky nezávislá [4]

Open source software se někdy používá jako synonymum svobodného softwaru. Do jisté míry to platí, ale existují drobné rozdíly mezi těmito kategoriemi. Například svobodný software zaručuje uživateli, že nemusí svou modifikovanou verzi zveřejňovat. Ovšem u open source softwaru může být součástí licenční smlouvy povinnost tyto verze zveřejňovat. Dalším rozdílem je redistribuce základního nemodifikovaného softwaru. U svobodného softwaru nemůže nikdo po uživateli chtít, aby nějak redistribuci hlásil, kdežto u open source softwaru se toto stát může [5].

3.3 Public domain software

Do této kategorie spadají programy, které nemají vůbec žádnou ochranu autorských práv. Kdokoli tedy může tyto programy změnit na komerční, nebo je libovolně měnit. Public domain software ale nemusí mít zveřejněné zdrojové soubory, jak je tomu například u svobodného nebo open source softwaru. Tento typ softwaru se však v praxi příliš neobjevuje [3].

3.4 Copyleftovaný software

Jedná se o svobodný software, který je chráněn copyleftem. Copyleftovaný software nemá žádné omezení pro redistribuci či modifikovatelnost. A to jak pro základní verzi, tak pro verze modifikované. Není možné ke změněné verzi přidávat jakákoli omezení [3].

3.5 Necopyleftovaný svobodný software

Takovýto software je dodáván s právy na redistribuci či změnu, ale i s jistými omezeními. Některé změněné verze tohoto softwaru mohou být nesvobodné [3].

3.6 Částečně svobodný software

Částečně svobodný software má se svobodným softwarem společnou volnou distribuci, modifikaci a používání. Rozdíl je ale v tom, kdo toto může dělat. Částečně svobodný software je možné používat jen pro individuální potřeby a nesmí být používán pro výdělečné účely [3].

3.7 Proprietální software

Proprietální software je takový software, který nelze označit za svobodný. Jsou to programy, u kterých nemá uživatel přístup ke zdrojovému textu, nesmí jej dále distribuovat ani jakkoli změnit. Pokud je program potřeba změnit, tak tuto změnu může provést pouze autor. Proprietální software však nemusí být vždy placený. Do kategorie proprietálních softwarů patří i freeware [3].

3.8 Freeware

Freeware je software, který je poskytován výrobcem bezplatně. Licence na něj zároveň není nijak časově omezená. Freeware ale nejde ztotožňovat se svobodným softwarem nebo s open source softwarem. U freewaru není k dispozici zdrojový kód. Je dodávána pouze spustitelná verze programu. Uživatel také nemá oprávnění jakkoli software měnit. Mívá však povoleno software dále distribuovat [3].

3.9 Shareware

Shareware je program, který lze za určitých podmínek bezplatně využívat. Tato bezplatná verze však bývá časově nebo funkčně omezená. Až po zaplacení licenčního poplatku se uživateli

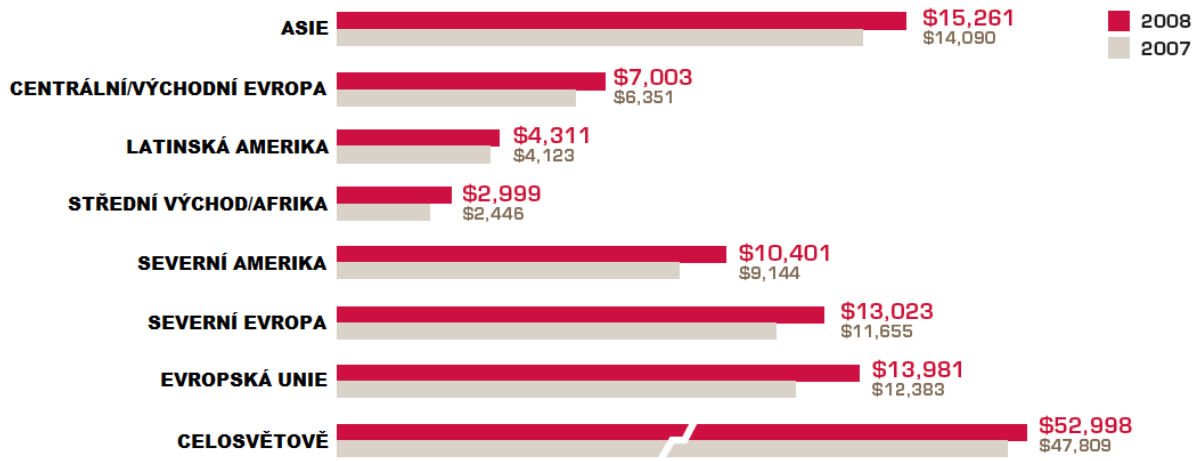
zpřístupní plná verze softwaru. Shareware může být většinou uživatelem dále distribuován, ale nemůže být modifikován. Proto také uživatel nemívá přístup ke zdrojovým textům [3].

3.10 Komerční software

Komerční software se někdy mylně spojuje pouze s kategorií proprietálního softwaru. Komerčním se může stát software z kterékoli z výše uvedených kategorií tím, že se zpoplatní jeho distribuce. Právě pro tuto kategorii softwaru se používá řada ochranných prvků proti nezákonnému používání [3].

4 Ochrana softwaru

Výrobci komerčního softwaru ztrácí každý rok značnou sumu peněz (viz graf 1) kvůli neautorizovanému využívání jejich výrobků. Proto je ochrana softwaru jejich prvořadým zájmem.



Graf 1: Ztráty v důsledku softwarového pirátství v milionech amerických dolarů [7]

Výrobci chrání svůj software nejčastěji licenčními klíči, omezeními, hardwarovými klíči a licenčními servery. Zvláštní kapitolou je ochrana nosičů softwaru.

4.1 Licenční klíč

Ochrana licenčním klíčem je dnes velmi často používaná. Licenční klíč musí uživatel zadat při spuštění nebo instalaci softwaru. Bez správného licenčního klíče nedojde k úspěšnému spuštění. Licenční klíče lze získat několika způsoby a jejich ověřování se také může lišit) [8].

4.1.1 Neměnný licenční klíč

Toto je asi nejsnáze prolomitelná ochrana softwaru. Útočníkovi v tomto případě stačí zjistit tento klíč a může software distribuovat dále s tímto klíčem. Tvůrce softwaru sice může klíč velmi dobře zašifrovat, ale i tak je pouze otázkou času, kdy se někomu klíč podaří dešifrovat) [8].

4.1.2 Licenční klíč vygenerovaný ze zadaných údajů

Tento způsob ochrany je mnohem účinnější než předešlý a je také častěji používaný. Licenční klíč se mění podle toho, pro jakého uživatele je poskytován. Uživatel je při koupi softwaru nucen zadat nějaké osobní údaje a z nich se vygeneruje licenční klíč, který se potom použije při instalaci.

4.1.3 Licenční klíč vygenerovaný z identifikace hardwaru

Při generování klíče se použije nějaká jednoznačná identifikace počítače, na který je software instalován. Jako identifikace může být použita například MAC adresa síťové karty nebo sériové číslo pevného disku. Tvůrce programu navíc může útočníkovi ztížit práci tím, že danou identifikaci dobře zašifruje, takže není poznat, která identifikace počítače je v generování použita) [8].

4.1.4 Licenční klíč kontrolovaný přes internet

Kontrolovat licenční klíče přes internet je velmi silné zabezpečení. Při každém spuštění softwaru proběhne kontrola správnosti klíče. Lze tímto způsobem snadno zjistit, jestli více uživatelů nepoužívá jeden licenční klíč. Takový klíč je pak uložen v databázi zakázaných klíčů a pomocí něj již není možné spuštění aplikace. Nevýhodou této ochrany je, že zákazník musí mít neustálý přístup k internetu. Proto je tato ochrana využívána u softwaru, který pro svůj běh vyžaduje připojení k internetu (tím pádem by si ho zákazník bez připojení nekoupil) [8].

4.2 Omezení práce se softwarem

Prodejce softwaru uveřejní tzv. trial verzi softwaru, která je nějakým způsobem omezena. Tato verze bývá většinou volně k dostání bez jakéhokoli poplatku. Uživatel si tedy může software vyzkoušet v omezené míře, aniž by si musel kupovat licenci. Omezení bývají různá. Buď jde o časové omezení, takže po určité době se aplikace již nespustí, nebo je dán maximální možný počet spuštění trial verze nebo je tato verze funkčně omezená - to znamená, že nefunguje nějaká důležitá funkce, která je nutná pro využívání programu (např. ukládání vytvořené práce). Software nebo programy, které jsou chráněny těmito omezeními, se liší podle toho, jak lze omezení zrušit.

4.2.1 Zrušení omezení zadáním licenčního klíče

Když softwaru vyprší časové omezení nebo omezení počtu spuštění, je uživatel nucen zadat licenční klíč, který mu zpřístupní plnohodnotnou verzi softwaru. Klíč většinou není nutné zadávat až při vypršení omezené verze, ale kdykoliv před tím. V případě funkčně omezeného softwaru je zadání klíče pouze na uživateli. Pro licenční klíče nutné pro rušení omezení platí to samé, co již bylo napsáno v předchozí kapitole o licenčních klíčích) [8].

4.2.2 Zrušení omezení nahráním registračního souboru

Jedná se o podobnou možnost jako u rušení omezení licenčním klíčem. Rozdíl je v tom, že zákazník nemusí zadávat licenční klíč, ale aplikace ho vyzve k vložení registračního souboru.

V tomto souboru mohou být různé informace. Nejjednodušší je registrační soubor, ve kterém je uložen licenční klíč. Další možností je vložit do registračního souboru společně s klíčem i kód pro zpřístupnění omezených funkcí programu. V každém případě je potřeba registrační soubor velmi dobře zašifrovat a chránit ho tak proti neoprávněnému přístupu) [8].

4.2.3 Koupě plné verze

V některých případech nelze trial verzi změnit na plnohodnotnou. Proto je zákazník nucen pořídit si plnou verzi softwaru a ne pouze licenční klíč či registrační soubor. V případě funkčního omezení to znamená, že nedostupné funkce nejsou v programu pouze zamčeny, ale že v něm nejsou vůbec implementovány. U časového omezení nebo omezení počtu spuštění se tato varianta příliš nepoužívá, protože program je ve své podstatě plně funkční a je proto zbytečné mít dvě verze programu) [8].

4.3 Hardwarový klíč

Ochrana hardwarovým klíčem je velmi silná a velmi těžko prolomitelná. K softwaru je přidán i hardware, bez kterého nemůže být používán. Jednak se aplikace neustále při běhu dotazuje na přítomnost tohoto klíče a jednak může potřebovat klíč pro dešifrování částí programu.

4.3.1 Hardwarový odemykací klíč

Tento druh klíče v sobě obsahuje malou paměť typu ROM, ve které je uloženo heslo. Aplikace se pak cyklicky během svého běhu ptá na toto heslo. Jestliže je hardwarový klíč vytažen při běhu aplikace, tak se aplikace ukončí. Heslo pak může být použito i pro dešifrování zašifrovaných částí programu, což ještě zvýší ochranu softwaru) [8].

4.3.2 Obálkový hardwarový klíč

Obálkový hardwarový klíč neobsahuje pouze heslo, jak je tomu u odvykacího klíče. Obálkový klíč funguje jako dešifrovací stroj. Na vstup dostává zašifrovaný program a na výstupu pak má dešifrovaný strojový kód programu) [8].

4.4 Licenční server

Při ochraně licenčním serverem dostane zákazník vygenerovaný licenční soubor se seznamem zakoupených licencí, které jsou chráněny digitálním podpisem. Při spuštění softwaru je licenčním

serverem kontrolován licenční soubor a podle něj buď povolen start aplikace, nebo zamítnut. Více podrobností o této ochraně najdete v kapitole 6.

4.5 Ochrana nosiče softwaru

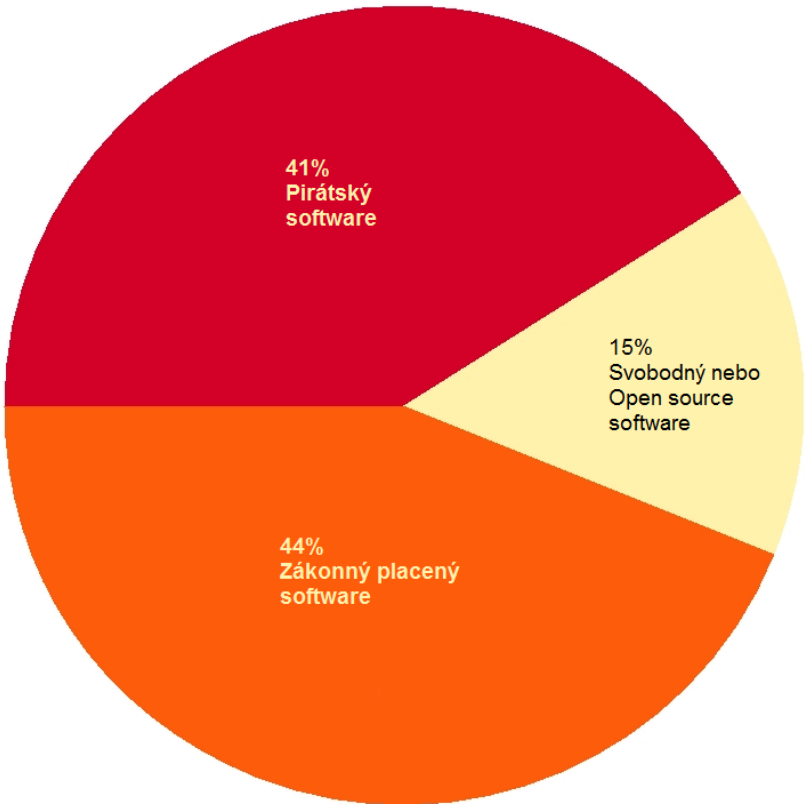
Každý prodejce softwaru se snaží, aby nikdo jeho produkt nekopíroval a neprodával dál. Z tohoto důvodu se snaží ochraňovat i samotné přenosové médium softwaru. Programy od některých výrobců mají v sobě zabudovanou funkci, která zjišťuje, jestli je v mechanice skutečně přítomné originální CD nebo DVD. Tento disk je pak chráněn digitálním podpisem. Používanou ochranou je úmyslné vložení chyb na nosič. Na originální nosič je při výrobě zaznamenáno několik chyb. Program potom kontroluje, zda jsou dané chyby opravdu přítomné na médiu. Programy na kopírování CD nebo DVD totiž takovéto chyby přeskakují a do kopie je nezapisují) [9].

4.6 Shrnutí

V této kapitole jsem popsal nejběžnější způsoby ochran softwaru. Může se samozřejmě používat kombinace jednotlivých ochran. Ochrana nosiče softwaru je dnes už naprosto běžná u všech druhů softwaru, které se prodávají na pevném médiu. Výrobci softwaru také dodávají software spolu s licenčním klíčem, který je nutno zadat při instalaci. Pro ochranu softwaru po nainstalování se používá hardwarový klíč, licenční server nebo kombinace těchto ochran (v tom případě je licenční soubor licenčního serveru pevně spjat s hardwarovým klíčem).

5 Porušování ochrany softwaru

V dnešní době je otázka porušování autorských práv velmi aktuální. Jak vyplývá z grafu 2, 41% používaného softwaru ve světě je nezákonná. Nákup nového softwaru není levnou záležitostí, a proto hodně lidí raději volí nezákonnou cestu. Nekoupí si originální software od výrobce, ale najdou si raději tzv. prolomenou verzi softwaru bez jakýchkoli ochranných prvků. Tyto úpravy softwaru dělají crackeři. To jsou lidé, kteří z různých důvodů překonávají ochrany softwaru. Nejčastěji to dělají kvůli finančnímu zisku, slávě nebo pouze pro zábavu[6]. Pro takovéto porušování ochrany se vžil název softwarové pirátství a pro software zbavený ochrany pirátský software.



Graf 2: Zastoupení kategorií softwaru) [7]

5.1 Nejčastější formy softwarového pirátství

Softwarové pirátství zabírá všechny možné způsoby obcházení nebo porušování autorských práv. Zde je několik nejčastějších způsobů:

- Kopírování programů uživateli pro jiné osoby

- Používání programů na více počítačích než na kolik má uživatel licence (toto se týká hlavně firem)
- Nelegální prodej softwaru
- Nelegální pronájem nebo půjčování softwaru
- Komerční využívání softwaru, u kterého je toto zakázané [10], [11].

5.2 Cracking

Pomocí crackingu se obchází ochrany licencovaného softwaru. Cracker studuje software a hledá mezery v zabezpečení licence. Studium nejčastěji provádí metodou zpětného inženýrství. Na software použije disassembler, který převede spustitelnou verzi softwaru na symbolický zápis assembleru. Tento kód je velmi nečitelný, ale zkušený programátor z něj může získat informace o chování programu. Těchto informací pak využije při prolomení ochrany [6]. Nyní následuje seznam ochran a způsoby jejich prolomení.

5.2.1 Licenční klíč

Prolomení ochrany licenčním klíčem je pro crackery asi nejjednodušší. Pokud má software pouze jeden správný licenční klíč, což je velmi slabá ochrana, tak crackerům stačí z programu zjistit pouze toto číslo a zveřejnit ho. Další možností je obejití kontroly licenčního klíče. Cracker obejde kontrolu tak, že změní podmínku u vyhodnocování správnosti klíče, takže místo toho, aby se software spouštěl pouze se správným klíčem, se spouštím se všemi klíči kromě správného. Ovšem tomu se dá předejít zakódováním určité části kódu správným klíčem. Zadání nesprávného klíče tudíž vede ke špatnému dekódování programu a jeho následné nefunkčnosti. Pokud se použije generování klíčů z osobních údajů zákazníka nebo z hardwarové identifikace, má cracker těžší práci. Nestačí zjistit pouze jeden klíč, ale musí přijít na to, z jakých informací se generuje licenční klíč a jaký je algoritmus generování. Když se licenční klíč kontroluje on-line na internetu, tak není žádný jednoduchý způsob, jak tuto ochranu obejít. Pokud cracker najde klíč, který server přijme, tak ho stejně může použít pouze pro jeden počítač a nemůže ho nabízet dále [8].

5.2.2 Omezení softwaru

Pokud je software chráněn časovým omezením, je nejčastějším útokem na tuto ochranu prodloužení doby, za kterou software přestane fungovat. Při omezeném počtu spuštění je situace obdobná, cracker jen přepíše počet spuštění. Jestliže je však software funkčně omezen, musí cracker přijít na to, jak funkce odblokovat. To se liší podle způsobu zrušení omezení. Při použití licenčního

klíče se použijí stejné postupy, jaké byly uvedeny výše. Jestliže je omezení rušeno registračním souborem, ve kterém je napsán zašifrovaný kód pro dekódování nepřístupných funkcí, musí cracker tento kód dešifrovat a určit jak se pomocí něj zpřístupňují omezené funkce [8].

5.2.3 Hardwarový klíč

Hardwarový klíč je asi nejhůře prolomitelná ochrana. Při ochraně hardwarovým odemykacím klíčem je nutné nejprve odstranit ze softwaru funkce, které se neustále ptají na přítomnost odemykacího klíče. Tím ovšem crackerovi práce nekončí, protože v tomto klíči je navíc obsaženo heslo pro dekódování určitých částí programu. Takže cracker musí toto heslo zjistit a zajistit pomocí něj správné dekódování programu. Obálkový hardwarový klíč je pro crackera ještě těžší oříšek. Musí totiž zjistit, jakým způsobem klíč dešifruje software. Klíče jsou vyráběny na vysoké technologické úrovni a je proto velmi nesnadné správně určit co vlastně dělají. Někdy crackerovi nezbývá nic jiného než si vytvořit napodobeninu klíče, což se ovšem mnohdy ani nevyplatí [8].

5.2.4 Licenční server

Prolomení ochrany licenčním serverem jde dělat více způsoby. Jednak je možné upravit licencovanou aplikaci tak, že se při jejím spuštění nevolá licenční server a tudíž nedojde ke kontrole licenčního souboru. Další možností je podvržení licenčního serveru jiným serverem vytvořeným crackerem. Zde je ovšem potřeba, aby cracker přesně znal komunikační protokol mezi serverem a aplikací a jelikož celá komunikace bývá silně šifrována, není to také nic jednoduchého. Jako poslední možnost je zmanipulovat licenční soubor. Licenční soubor je sice pro člověka dobře čitelný a snadno z něj jde získat informace, ale změna dat v souboru je velmi složitá, protože každý řádek je chráněn digitálním podpisem. Cracker by tedy musel znát soukromý klíč dodavatele softwaru, aby mohl zakódovat upravená data v souboru.

6 Kontrola licencí licenčním serverem

Ochrana licenčním serverem je způsob ochrany softwaru, který byl vybrán pro projekt Lissom. Návrh serveru vychází z komerčního produktu FLEXnet [12].

Celý koncept licencování zahrnuje tyto části:

- Licencovaná aplikace
- Licenční server (složený z licenčního manažera a vendor daemona)
- Licenční soubor

Licenční manažer vlastně není nic jiného než prostředník mezi aplikací a vendor daemonem, který se stará o ověření licenčního souboru. Z toho důvodu je možné a v praxi i využívané mít jeden licenční manažer pro více vendor daemonů. Důležité je pouze dodržení komunikačního protokolu.

6.1 Licencovaná aplikace

Licencovaná aplikace musí před samotným spuštěním zjistit, zda je dostupná licence, která dovoluje její spuštění. Musí v sobě tedy mít zabudovaný mechanizmus ověřování. Každá aplikace musí mít přístup k příslušnému licenčnímu souboru. Pokud je licence aplikace vázaná (viz kapitola 6.6) a není omezena počtem souběžně spuštěných instancí aplikace, proběhne ověření pouze v rámci aplikace a není potřeba volat licenční manažer. V opačném případě aplikace určí z licenčního souboru umístění licenčního manažera, se kterým se spojí a zašle mu svou identifikaci. Potom jen čeká na rozhodnutí, zda je licence platná a zda se může spustit.

6.2 Licenční manažer

Licenční manažer je jednotka, která přímo komunikuje s aplikací, která je licencována. Po ustavení spojení zašle aplikace manažeru svoji identifikaci a manažer pomocí svých licenčních souborů určí, ke kterému vendor daemonu aplikace náleží. Dále spustí daný daemon a předá mu licenční soubor a identifikaci aplikace, která žádá o povolení ke spuštění. Rozhodnutí démona potom přepošle aplikaci. Protože činnost prováděná manažerem je stejná pro všechny aplikace od všech prodejců, tak v praxi bývá manažer dodáván třetí stranou.

6.3 Vendor daemon

Vendor daemon je program, který je dodáván vždy výrobcem licencované aplikace. Kontroluje totiž plnění firemní licenční politiky a ověřuje digitální podpis licencí. Daemon kontroluje

licenční soubor, který mu zaslal licenční manažer. Určí, zda nebylo se souborem neoprávněně manipulováno a následně z něj zjistí, jaká licence platí pro danou aplikaci. Jestliže má licence omezený počet souběžně spuštěných instancí aplikace, tak si zjistí z registru právě používaných licencí, který si uchovává, jestli už není počet vyčerpán. Podle zjištěných informací pak rozhodne, zda se aplikace může spustit. Registr právě používaných licencí může spravovat i licenční manažer.

6.4 Umístění licenčního serveru

Licenční server se nachází vždy u zákazníka a jsou v podstatě dvě možnosti, kam ho může instalovat. Při užití vázané počítané licence může být licenční server na stejném počítači, jako aplikace. Pokud však vyžaduje mít aplikaci na více počítačích, umisťuje se licenční server na nějaký server v zákazníkově síti, které je dostupný ze všech stanic využívajících aplikaci. Tento server pak musí být ale neustále v provozu, jinak by uživatelé nemohli spouštět aplikaci. Dále se také využívá ochrana redundantními servery. V tom případě jsou v síti spuštěny tři licenční servery. Servery si mezi sebou periodicky zasílají zprávy, kterými dávají najevo, že jsou aktivní. Při havárii jednoho z nich tak zbylé servery brzy zjistí jeho nefunkčnost. Pro příjem požadavků od licencovaných aplikací musí být aktivní dva servery. Jeden je označen jako primární a k tomu jsou směřovány všechny požadavky. Pokud by se primární server stal z jakéhokoli důvodu nedostupným, tak ho nahradí druhý. Uživatelé pak mají větší jistotu, že budou moci využívat licencované aplikace. Nevýhodou je jednak nutnost mít tři fyzické servery a pak také to, že při jednom aktivním serveru už není služba k dispozici.

6.5 Licenční soubor

Licenční soubor je základním informačním zdrojem v licencování. Licencovaná aplikace z něj získává umístění manažera a manažer umístění démona, který z něj čte informace o jednotlivých licencích.

6.5.1 Formát licenčního souboru

Licenční soubor FLEXnet má přesně danou syntaxi. Skládá se z informací o licenčním serveru, vendor daemonu a nakonec ze samotných licencí využívaných daným zákazníkem.

6.5.1.1 Informace o licenčním serveru

Prvním řádkem licenčního souboru jsou informace o licenčním serveru (nebo o licenčních serverech, pokud jsou použity redundantní servery) v tomto formátu:

SERVER host hostid [port] [PRIMARY_IS_MASTER] [HEARTBEAT_INTERVAL=seconds]

[12]

Povinné parametry:

- host – doménové jméno nebo IP adresa licenčního serveru
- hostid – jednoznačná identifikace stanice, na které je licenční server spuštěn. Nejčastěji to bývá MAC adresa nebo sériové číslo disku

Volitelné parametry:

- port – TCP/IP port, na kterém licenční server naslouchá.
- PRIMARY_IS_MASTER – tento parametr se zadává u primárního serveru, pokud se použijí redundantní servery. Když primární server spadne, tak začne vyřizovat požadavky sekundární server. Pokud je tento parametr zadán, tak se při opětovném zprovoznění primárního serveru předá vyřizování zpátky na primární. Jestliže ne, tak sekundární bude požadavky vyřizovat dál.
- HEARTBEAT_INTERVAL – určuje, kolik sekund čeká server při redundantní konfiguraci na periodicky zasílanou zprávu, než se sám ukončí.

6.5.1.2 Informace o vendor daemonu

Další položkou v licenčním souboru jsou informace o vendor daemonu ve formátu:

VENDOR vendor [vendor_daemon_path] [[OPTIONS=]options_file_path] [[PORT=]port]

[12]

Místo klíčového slova VENDOR se v dřívějších verzích licenčních souborů užívalo slovo DAEMON.

Povinné parametry:

- vendor – název vendor daemona

Volitelné parametry:

- vendor_daemon_path - cesta k daemonu. Pokud není zadána, tak licenční manažer hledá démona v předem nastavených umístěních.
- OPTIONS - cesta k souboru s uživatelovým nastavením
- PORT - TCP/IP port daemona

6.5.1.3 Vynucení volání serveru

Jestliže chce prodejce z jakéhokoli důvodu vynutit volání licenčního serveru pro každé ověřování, tedy i pro vázané licence, které se jinak ověřují pouze aplikací, může zadat pod informace o manažeru

a daemonu na samostatný řádek parametr USE_SERVER. Tím se veškeré ověřování převede na server.

6.5.1.4 Informace o licencích

Po těchto položkách, které by se dali označit za hlavičku licenčního souboru, následují řádky, které obsahují samotné licence zákazníka:

{FEATURE|INCREMENT} feature vendor feat_version exp_date num_lic SIGN=sign
[optional_attributes]

[12]

Mohou být uvozeny klíčovými slovy FEATURE nebo INCREMENT. Klíčovým slovem FEATURE začíná licence, která je základní pro danou licencovanou aplikaci. V licenčním souboru může být pro jednu aplikaci pouze jedna licence typu FEATURE. Pokud by se v souboru objevila i jiná, tak se bere v potaz pouze první z nich, ostatní se zahazují. Jestliže se však chce k aplikaci přidat další licence, použije se právě již zmiňovaný typ INCREMENT. Pro jednu aplikaci může být neomezený počet licencí tohoto typu.

Povinné parametry:

- feature - jednoznačné pojmenování aplikace výrobcem
- vendor - název vendor démona, který spravuje danou aplikaci
- feat_version - Verze aplikace
- exp_date - datum vypršení licence. Pokud je licence časově neomezená, tak se zapíše do data rok 0, nebo je místo data zapsán řetězec „permanent“.
- num_lic - udává, kolik instancí dané aplikace může mít zákazník spuštěných najednou. Jestliže zákazník nemá omezený počet, je do souboru zapsán počet 0, nebo řetězec „uncounted“. V tomto případě je však nutné, aby byl u tohoto řádku souboru napsán parametr hostid (viz dále).
- SIGN - každý řádek licence musí obsahovat digitální podpis, který kontroluje daemon.

Dále může licence obsahovat volitelné parametry, které se zapisují stylem klíčové_slovo=hodnota.

Volitelné parametry zadávané prodejcem:

- HOSTID – tento parametr je povinný pouze u licencí s neomezeným počtem. Hostid jednoznačně identifikuje počítač, na kterém se bude aplikace spouštět. Pokud je dovoleno používat licenci na více počítačích, může být na jednom řádku licence specifikováno více hostid. Další možností je zápis hostid jako IP adresu sítě či podsítě, ve které se licencovaná aplikace bude spouštět.
- BORROW – parametr se zadává při výpůjčce dané licence a udává na kolik hodin je půjčená licence platná.

- DUP_GROUP - parametr udává skupinu, pro kterou je povoleno tzv. duplikování licence. To znamená, že když daná skupina spustí více instancí aplikace, z počtu licencí na danou aplikaci se vezme pouze jedna. Duplikovat se může pro uživatele, počítač, displej nebo pro výrobcem definovanou skupinu.
- FLOAT_OK – povoluje mobilní licencování dané aplikace
- HOST_BASED – udává počet stanic, na kterých může být licence používána. Identifikace těchto stanic musí být uložena v uživatelském souboru nastavení. Počet stanic nesmí přesáhnout počet licencí parametru num_lic.
- ISSUED – datum vydání licence
- ISSUER – vydavatel licence
- NOTICE – poznámky týkající se duševního vlastnictví
- OVERDRAFT – povoluje zákazníkovi krátkodobé využití více licencí, než kolik má zaplaceno. Parametr udává maximální počet licencí nad limit.
- PLATFORMS – seznam podporovaných platforem
- SN – sériové číslo licence
- START – datum, od kterého se může licence využívat
- SUPERSEDE – parametr obsahuje seznam názvů aplikací. Pokud v souboru existuje řádek pro nějakou ze zadaných aplikací, který má starší datum vydání než řádek s touto licencí, pak se při ověřování licencí bude hledět pouze na řádek s tímto parametrem a ne na řádky se starší licencí.
- TS_OK – parametr musí být zadán, jestliže chce zákazník přistupovat k licencované aplikaci přes Windows Terminal Server. Bez tohoto parametru by licenční server nepovolil spuštění aplikace.
- USER_BASED – udává počet uživatelů, kteří mohou využívat licenci. Identifikace uživatelů musí být uložena v uživatelském souboru nastavení.
- VENDOR_STRING – jakýkoli řetězec definovaný prodejcem

Volitelné parametry zadávané uživatelem:

- asset_info – dodatečné informace od administrátora licencí pro asset management
- dist_info – dodatečné informace od poskytovatele licencí
- sort – určuje řazení v licenčním souboru. Řadí se od nejnižšího čísla po největší.
- user_info – dodatečné informace od administrátora licencí
- vendor_info – dodatečné informace od prodejce softwaru

6.5.1.5 Informace o balíčcích

Pokud chce prodejce softwaru vydat licenci na nějaký balíček aplikací, může k tomu využít tento řádek, kterým definuje balíček:

*PACKAGE package vendor [pkg_version] COMPONENTS=pkg_list *
[OPTIONS=SUITE] [OPTIONS=SUITE_RESERVED]
[SUPERSEDE[="p1 p2 ..."] ISSUED=date] SIGN=pkg_sign

[12]

K tomuto řádku musí existovat také řádek s licencí, ve které musí být název aplikace nastaven na jméno balíčku.

Povinné parametry:

- package – název balíčku
- vendor – název vendor daemona
- COMPONENTS – seznam názvů všech aplikací v balíčku
- SIGN - každý řádek licence musí obsahovat digitální podpis, který kontroluje daemon.

Volitelné parametry:

- pkg_version – verze balíčku
- OPTIONS=SUITE – pokud je tento parametr zadán, tak při povolení spuštění jedné aplikace z balíčku se nespotřebuje pouze jedna licence dané aplikace, ale jedna licence celého balíčku. V opačném případě se licence spotřebovávají pro každou aplikaci zvlášť.
- OPTIONS= SUITE_RESERVED – pokud uživatel spustí jednu z aplikací v balíčku a tento parametr je nastavený, tak se mu automaticky zarezervují licence i na ostatní aplikace v balíčku.
- SUPERSEDE – obsahuje seznam názvů balíčků. Při ověřování se nahradí balíčky ze seznamu, které mají starší datum vydání, řádkem s tímto parametrem
- ISSUED – datum vydání

6.5.1.6 Informace o upgradech

V licenčním souboru se může zaznamenat přechod licencované aplikace na novější verzi pomocí řádku s touto syntaxí:

*UPGRADE feature vendor from_feat_version to_feat_version exp_date num_lic *
[options ...] SIGN=sign

[12]

Všechny parametry kromě from*_feat_version* a *to_feat_version* jsou shodné jako u řádku s licencí. Tyto dva parametry obsahují čísla staré a nové verze aplikace. Řádek UPGRADE však nepřidává nové licence. Počet licencí uvedený v tomto řádku neoznačuje počet nových licencí, ale počet licencí převedených ze staré na novou verzi.

6.5.1.7 Další informace o syntaxi licenčního souboru

Licenční soubor může obsahovat také komentáře, které jsou uvozeny znakem ‚#'. Komentář je platný na jeden řádek. Pokud je potřeba jeden řádek licence rozdělit na dva nebo více řádků, používá se pro rozdělení znak '\'.

Licenční soubor má dále specifikované doporučené pořadí. Pro jednu aplikaci se zapisují nejdříve řádky typu FEATURE a až potom řádky typu INCREMENT. Pokud je pro jednu aplikaci více řádků typu FEATURE, tak ten, který se má použít, musí být uveden jako první, protože ostatní řádky tohoto typu se nezpracovávají. Další doporučení je zapisovat vázané nepočítané licence před plovoucími licencemi. Je to z toho důvodu, že plovoucí licence by mohli být vyčerpány uživatelem, pro kterého je nastavena vázaná licence a tím by došlo ke znemožnění spuštění aplikace uživatelům, kteří používají plovoucí licence.

6.5.2 Podpis licenčního souboru

Licenční soubor se nachází u zákazníka a zákazník k němu má volný přístup. Je proto nesmírně důležité tento soubor dobře chránit proti neoprávněným změnám. Ochrana souboru spočívá v podepsání každého řádku obsahující licenci digitálním podpisem. Do podpisu musí být zahrnuty všechny parametry licence, které nesmí zákazník měnit, a k tomu ještě parametr hostid serveru (případně tři hostid serverů pro redundantní konfiguraci).

6.5.3 Umístění licenčního souboru

Každý prodejce softwaru má přesně specifikované umístění, kde se musí nacházet licenční soubor. Pokud zákazník využívá licenci pouze na jednom počítači, tak se soubor uloží na prodejcem specifikované místo na disku. Pokud však chce zákazník využívat licence na více počítačích, jsou dvě možnosti, jak umístit licenční soubor. Buď se nakopíruje do všech počítačů využívajících licenci na specifikované místo, nebo se uloží na nějaké místo v síti, ze kterého je licenční soubor dostupný všem počítačům.

6.6 Způsoby licencování pomocí licenčního serveru

6.6.1 Plovoucí licence

Plovoucí licence je určená pro použití v celé síti zákazníka. Není vázána na žádný počítač. Každý v síti, kdo má nainstalovanou licencovanou aplikaci a má přístup k licenčnímu serveru, může aplikaci používat. Plovoucí licence ale musí mít specifikovaný maximální možné současné používání

licence. Proto je nutné mít neustále aktivní licenční server, aby vedl záznam o právě používaných licencích. U těchto licencí není možné mít nepočítané licence.

6.6.2 Vázané licence

Vázané licence jsou vždy pevně spjaty s určitým počítačem nebo se skupinou počítačů, s hardwarovým klíčem nebo s uživatelem. Tyto licence mohou být jak počítané tak nepočítané. Jelikož jsou tyto licence vázané na počítač (počítače), tak je nutné mít u každé licence v licenčním souboru zadané hostid.

6.6.3 Časově omezené licence

Prodejce softwaru má možnost časově omezit platnost licence. Zároveň to může být výhodnější i pro zákazníka, pokud ví, že produkt bude potřebovat jen po omezenou dobu a nemusí si tedy kupovat neomezenou licenci. Časové omezení licence je uvedené u každé licence jako datum expirace.

6.6.4 Trial licence

Těmito licencemi dává prodejce možnost zákazníkovi vyzkoušet si bezplatně funkčnost jeho produktu na omezenou dobu. V podstatě se jedná o speciální případ časově omezené licence.

6.6.5 Platba za používání

Zákazník v tomto případě nepředplatí licenci, ale za určitý časový úsek licenční server vygeneruje soubor se záznamy využívání licence a podle něj prodejce určí poplatek pro zákazníka. Licenční server si musí uchovávat veškeré informace o spouštěných aplikacích [13].

6.6.6 Token licence

Tento způsob licencování je vhodný pro zákazníky, kteří dopředu přesně nevědí, kolik licencí od daných produktů budou potřebovat. Nekoupí si tedy určitý počet licencí, ale určitý počet tokenů, které pak licenční server přiděluje aplikacím, které žádají o spuštění. Existují dva druhy tokenů – konkurentní a konzumptivní.

Využívání konkurentních tokenů se velmi podobá plovoucím licencím. Rozdíl je v tom, že server nepřiděluje licence, ale tokeny. Každá aplikace má daný počet tokenů, které potřebuje pro svůj běh. Po ukončení pak aplikace tokeny vrátí serveru.

Konzumptivní tokeny aplikace po svém ukončení nevrací. Tento typ licencování se hodí zejména pro transakční aplikace, kdy na každou transakci spotřebuje jeden či více tokenů. Tokeny se také mohou spotřebovávat podle doby běhu aplikace, takže např. jedna hodina běhu dané aplikace spotřebuje jeden token.

Výhodné je, že nejsou speciální tokeny pro každou licencovanou aplikaci, ale více aplikací může využívat stejné tokeny [14].

6.6.7 Půjčování licencí

Pokud počítač, na kterém se využívá licencovaná aplikace, nemá nepřetržitý přístup k licenčnímu serveru, může si zákazník na dobu, kdy není připojen k serveru, licence půjčit. K tomu se používá parametr BORROW v licenčním souboru. Prodejce vydá zákazníkovi licenci s tímto parametrem. Zákazník jen uvede datum vrácení půjčené licence a licenční server zapíše na zákazníkův počítač informace o půjčce, které bude aplikace kontrolovat při spuštění. Licenční server zapůjčí licenci na danou aplikaci a označí ji jako využívanou, tudíž nikdo jiný nemůže v době výpůjčky tuto licenci využít.

6.6.8 Licence na balíček produktů

Pokud prodejce softwaru nabízí balíček produktů, není třeba vystavovat licence na každý produkt v balíčku zvlášť, ale může dát jednu licenci na celý balíček. Hlavní výhodou je, že se výrazně zkrátí počet záznamů v licenčním souboru. Zapíše se do něj totiž jenom popis balíčku (řádek PACKAGE) a popis licence dané zákazníkovi k němu.

6.7 Počítané a nepočítané licence

Nepočítané licence jsou takové licence, které nejsou nijak omezeny počtem současně spuštěných instancí jedné aplikace. Takovéto licence mají v licenčním souboru počet licencí nastaven na 0. Pokud jsou v souboru všechny licence nastavené na nepočítané, tak ani není potřeba mít licenční server. Ale licenční server může přijímat požadavky i z aplikace, která nemá omezený počet, pokud je v licenčním souboru zapsán řádek USE_SERVER, kterým se dává najevo, že veškeré ověřování se musí provádět na licenčním serveru. U počítaných licencí je však přítomnost licenčního serveru nutná, protože musí být přehled o právě spuštěných programech. Vendor daemon hlídá maximální počet spuštěných aplikací a dalším nepovolí spuštění. Existují i techniky pro dočasné povolení určitého počtu licencí nad limit. Dobu dočasného přetečení licencí určuje firemní licenční politika.

7 Popis praktické části práce

Jako praktickou část této práce jsem vytvořil návrh databáze licencí a k této databázi uživatelské rozhraní, dále pak program, který ze záznamů v databázi generuje licenční soubor pro daného zákazníka a nakonec knihovnu pro práci s licenčními soubory. Ve všech částech práce jsem se nechal inspirovat komerčně využívaným licenčním serverem FLEXnet.

V licenčním serveru pro projekt Lissom nebudou využívány všechny způsoby licencování popsané v kapitole 6. Ze všech způsobů licencování, které může server poskytovat, budou implementovány pouze plovoucí, vázané a časově omezené licence. Pro tyto způsoby je potřeba mít u licencí zapsány pouze povinné parametry a volitelné parametry hostid a duplikace. Ostatní parametry nebudou nevyužívány, a proto jsem je ani nezahrnul do návrhu.

7.1 Databáze licencí

7.1.1 Návrh databáze licencí

Při návrhu databáze jsem se snažil, aby výsledná databáze nebyla příliš rozsáhlá, aby v ní byly pouze skutečně potřebné informace.

Základem celé databáze je tabulka obsahující jednotlivé produkty a tabulka k nim náležejících modulů, ke kterým si zákazník nakupuje licence. Původně jsem vytvořil návrh bez tabulky produktů. Zákazníci si tedy nakupovali pouze moduly a nijak jsem neřešil jejich začlenění do produktů. To sice z hlediska funkčnosti distribuce licenčních souborů z databáze ničemu nevadilo, ale výsledná databáze byla díky tomu nepřehledná. Proto jsem nakonec zvolil variantu členění na produkty a moduly. Výsledné uspořádání je logičtější a lépe se v něm orientuje. Je tak možné vidět pro daného zákazníka, jaké si nakoupil produkty a jaké moduly z nabídky si vybral, což je mnohem lepší, než když bylo možné u zákazníka zjistit pouze všechny moduly ze všech produktů dohromady.

Další základní částí databáze je tabulka zákazníků a jejich kontaktních osob. Tabulka zákazníků je navržena pro právnické osoby, ale pokud by zákazníkem byla i fyzická osoba, může být vynechán záznam obsahující IČO a místo názvu firmy se vyplní jméno zákazníka. Jeden zákazník pak může mít v podstatě neomezený počet kontaktních osob, které vyčlení ze své firmy na správu licencí.

Každá licence zakoupená zákazníkem je uložená v databázi s náležitými parametry. Tyto parametry jsou shodné s parametry v licenčním souboru.

Jako poslední tabulka je v databázi seznam administrátorů, kteří mohou přistupovat k uživatelskému rozhraní databáze.

Propojení jednotlivých tabulek do databáze ukazuje ER diagram.

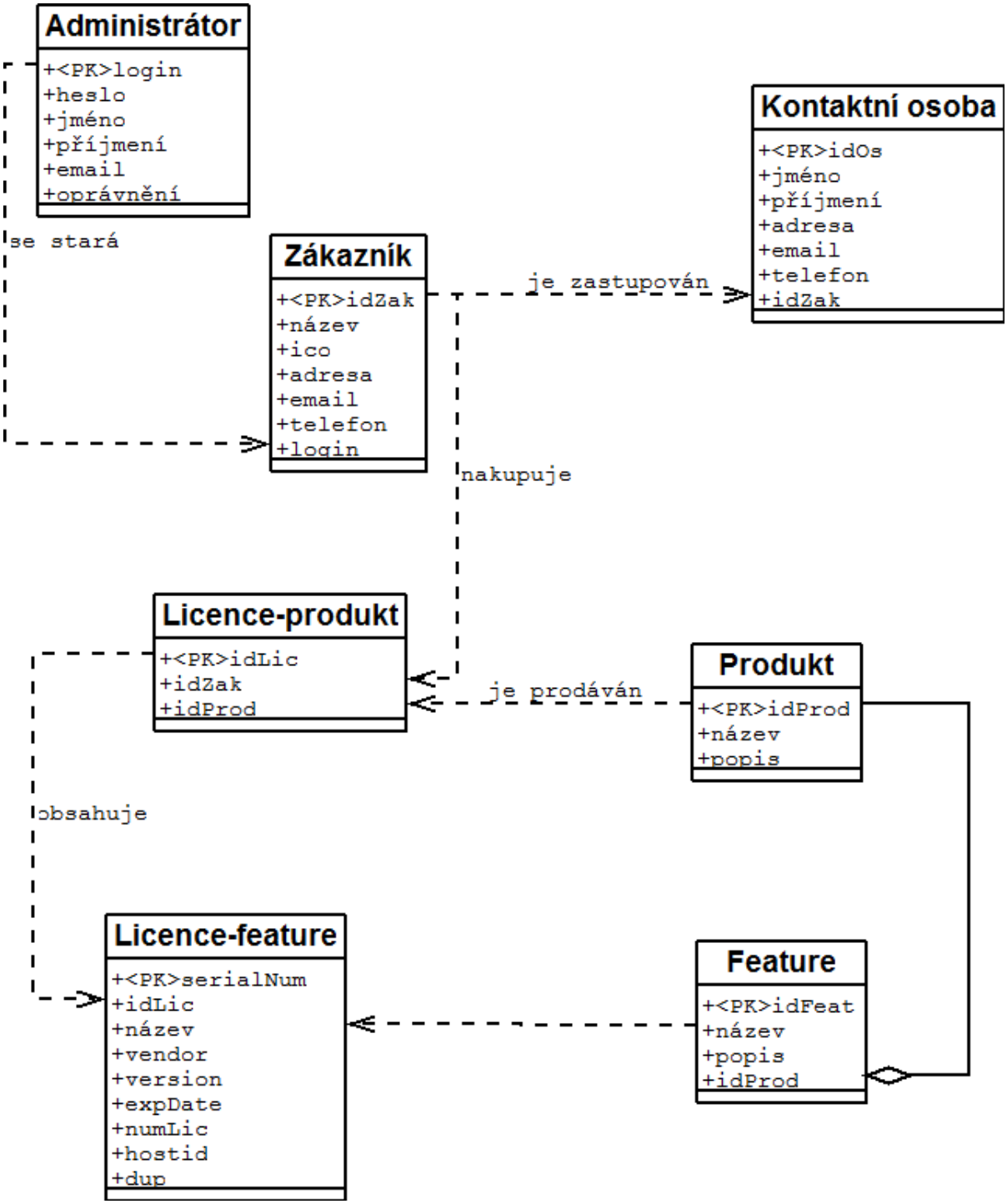


Diagram 1: ER diagram

7.1.2 Implementace nástroje pro správu databáze

Nad databází jsem vytvořil uživatelské rozhraní formou webových stránek. Jako implementační jazyk jsem vybral PHP. Tento skriptovací jazyk je využíván právě pro tvorbu dynamických webových stránek. Začleňuje se přímo do značkovacího jazyka HTML. PHP skript je většinou prováděn na serveru a ke koncovému uživateli se dostává pouze výsledek skriptu. PHP je

vhodnou volbou také proto, že v sobě obsahuje knihovnu pro práci s databázovým systémem MySQL, který jsem využil pro uložení databáze [15].

7.1.2.1 Správa administrátorů

Nástroj pro správu databáze je přístupný pouze oprávněným administrátorům. Nelze tedy k webové aplikaci přistoupit bez znalosti přístupového jména a hesla. Do aplikace jsem zakomponoval i pomůcku pro zapomenuté heslo. Jestliže administrátor zapomene heslo, stačí na přihlašovací stránce kliknout na odkaz *Zapomenuté heslo*. Objeví se formulář, do kterého administrátor zadá svoje přihlašovací jméno a e-mail. Pokud je kombinace e-mailu a jména správná, tak se mu na zadaný e-mail zašlou přihlašovací údaje. Přihlášený administrátor má samozřejmě možnost měnit svoje údaje. Může si změnit heslo, jméno, příjmení a e-mail, ale v žádném případě si nemůže měnit přihlašovací jméno a jeho oprávnění může měnit pouze superadministrátor. Tím se dostávám k rozlišení dvou stupňů oprávnění – administrátor a superadministrátor. Administrátor má práva na přidávání a úpravu produktů a modulů, zákazníků a jejich kontaktních osob a přidávání licencí k zákazníkovi, kterého mají pod svou správou. Superadministrátor má navíc právo přidat nového administrátora, měnit u zákazníka administrátora, který ho spravuje, a také měnit oprávnění jiných administrátorů, ale ne svoje vlastní (pokud by byl jediný superadministrátor a změnil si oprávnění na administrátora, tak by se už nikdo jiný nemohl stát superadministrátorem).

Při vkládání nového administrátora jsou zadávané hodnoty kontrolovány. U loginu se kontroluje, zda se skládá pouze ze znaků anglické abecedy a zda se tento login již v databázi nevyskytuje. Druhý parametr, který podléhá kontrole, je emailová adresa. U ní se kontroluje přítomnost znaku '@' a za ním následující domény. U jména a příjmení se kontroluje pouze jejich přítomnost.

7.1.2.2 Správa zákazníků

Správa zákazníků je ve webové aplikaci umístěna v sekci *Zákazníci*. Zde se každému administrátorovi zobrazí seznam zákazníků, u kterých je uveden jako správce, a operace, které může s daným záznamem provádět.

Administrátor může přidávat nového zákazníka. Hodnoty jednotlivých položek jsou opět kontrolovány. Emailová adresa naprosto shodným způsobem jako u kontroly administrátorů, IČO musí mít pevnou délku 8 číslic a poštovní směrovací číslo 5 číslic, telefonní číslo musí začínat mezinárodní předvolbou. Při ukládání nového zákazníka se generuje automaticky jeho identifikační číslo v databázi a jako správce tohoto zákazníka je uveden administrátor, který informace o tomto zákazníkovi ukládal. Superadministrátor může později změnit správce. K tomu slouží odkaz *Změnit správce zákazníků* v sekci *Zákazníci*. Zde se objeví tabulka všech zákazníků a k nim přidělených

administrátorů. U každého ze zákazníků může vybrat nového správce ze seznamu všech administrátorů v databázi.

Jednotlivé záznamy o zákaznících se dají také upravovat. K tomu slouží stejný formulář, jako pro zadání nového zákazníka, pouze jsou v něm předvyplněny aktuální hodnoty všech parametrů.

Správce může u každého svého zákazníka přidávat a upravovat informace o kontaktních osobách. Formulář pro tyto akce je typickým formulářem pro fyzickou osobu.

7.1.2.3 Správa produktů

V sekci *Produkty* je zobrazen seznam produktů v databázi. U každého produktu může administrátor měnit pouze popis produktu. Dále může administrátor přidávat nové produkty pouhým zadáním názvu produktu a jeho popisu.

Ke každému produktu pak administrátor přidává jednotlivé moduly, které k němu náleží. Modul má stejné parametry jako produkt a lze u něj měnit také pouze popis.

7.1.2.4 Správa licencí

Asi nejdůležitější funkcí celého nástroje pro správu databáze je přidávání licencí k zákazníkovi. Administrátor nejprve musí vybrat, který produkt si chce zákazník koupit. Poté pro každý zakoupený produkt může vybírat z nabídky modulů pro tento produkt. U každého modulu musí zadat informace, které jsou pro licencování nejdůležitější, protože budou zapsány v licenčním souboru. Zákazník si může samozřejmě zakoupit více licencí na jeden modul. Datum expirace zadává administrátor ve formátu čitelném pro člověka – den.měsíc.rok. Protože formát data v databázích MySQL je rok-měsíc-den, tak jsem vytvořil funkci pro převod těchto dvou formátů. Zadané informace o licencích na moduly už se nemohou měnit. Lze pouze k modulům přidávat nové licence.

7.2 Generátor licenčních souborů

Generátor licenčních souborů má za úkol získat data z databáze licencí, tyto data zapsat podle přesně dané syntaxe do souboru a podepsat je digitálním podpisem. Jako implementační jazyk byl vybrán jazyk C.

7.2.1 Hlavička souboru

V hlavičce souboru jsou informace, které nejsou umístěny v databázi. Jsou to informace o licenčním manažeru a vendor daemonu, které musí dodat zákazník. Administrátor licencí tedy musí od zákazníka tyto údaje získat a až poté může být vygenerován licenční soubor. Informace potřebné pro vytvoření hlavičky jsou generátoru předkládány jako parametry programu. Generátor musí být schopen rozeznat, zda má všechny potřebné parametry. Musí tedy rozlišovat povinné a nepovinné

údaje. Případnou nepřítomnost důležitých parametrů pak musí ohlásit a nepřipustit vytvoření nekorektního licenčního souboru.

7.2.2 Práce s databází

Generátor musí nejprve získat data z databáze pro zákazníka, kterému se licenční soubor generuje. Program se tedy musí připojit k databázi a vybrat z ní správné údaje pro zapsání do souboru. To znamená získat seznam všech produktů, které si zákazník koupil a následně ke každému z těchto produktů seznam licencí na moduly.

7.2.3 Tvorba řádku licence

Data z databáze je potřeba složit do jednoho řádku podle formátu FLEXnet. Volitelné argumenty u licence FLEXnet jsou vždy uvozeny svým klíčovým slovem pro identifikaci, protože není předem známo, zda se v souboru objeví a případně v jakém pořadí. V rámci generátoru je však potřeba rozlišovat pouze dva volitelné parametry uvozené klíčovými slovy HOSTID a DUP. Nakonec je ještě potřeba podepsat celý řádek digitálním podpisem.

7.2.4 Digitální podpis

Jak již bylo zmíněno, každý řádek licenčního souboru musí být chráněn proti přepsání digitálním podpisem.

Nejprve je nutné určit přesně syntaxi kódovaného řetězce. Do řetězce musí být zapsány všechny informace z daného řádku a k tomu hostid licenčního serveru. Nabízí se tedy vytvořit řetězec podle formátu licenčního řádku doplněného o hostid serveru. Formát tedy vypadá takto:

typ_licence modul vendor verze datum_exp počet_licencí [HOSTID=hostid] [DUP=duplikace]\ hostid_serveru

Dále bylo potřeba vybrat vhodný podepisovací algoritmus. Zde jsem se opět nechal inspirovat produktem FLEXnet, který licenční soubory kóduje pomocí kryptografie nad eliptickými křivkami. Tento způsob kryptografie je využívaný hlavně kvůli jeho rychlosti a také kvůli krátké délce klíče, která je potřebná pro bezpečné zašifrování. Bezpečnost tohoto kryptosystému je postavena na řešení úlohy diskrétního logaritmu, pro který zatím neexistuje žádný algoritmus se subexponenciální výpočetní složitostí. Pro podpis jsem použil algoritmus ECDSA, který je postaven právě na kryptografii nad eliptickými křivkami. Více o kryptografii nad eliptickými křivkami a algoritmu ECDSA můžete najít zde [16]. Tento algoritmus je implementován v otevřené knihovně OpenSSL [19], kterou jsem při implementaci programu využil.

Jelikož kryptografie nad eliptickými křivkami je asymetrická, musí se nejprve vytvořit pár šifrovacích klíčů – soukromý, který se použije při podpisu řetězce, a veřejný, kterým se podpis ověřuje. Klíče jsem vytvořil nad 239-bitovou eliptickou křivkou [17]. Dvojici těchto klíčů jsem vygenerovat také pomocí knihovny OpenSSL a do programu jsem je zanesl jako řetězcové konstanty. V generátoru je potřebný pouze soukromý klíč, protože se v něm podpis vytváří, ale neověřuje.

Algoritmus ECDSA dále požaduje, aby podepisovaný řetězec nebyl podepisován přímo, ale aby z něj byl nejprve vytvořen hashovaní funkcí otisk a teprve ten je podepsán. Jako hashovaní funkci jsem vybral SHA. Tato funkce převádí řetězec dlouhý maximálně 2^{64} bitů na otisk o délce 20 bytů [18].

7.2.5 Implementace generátoru

Generátor jsem implementoval podle následujícího vývojového diagramu.

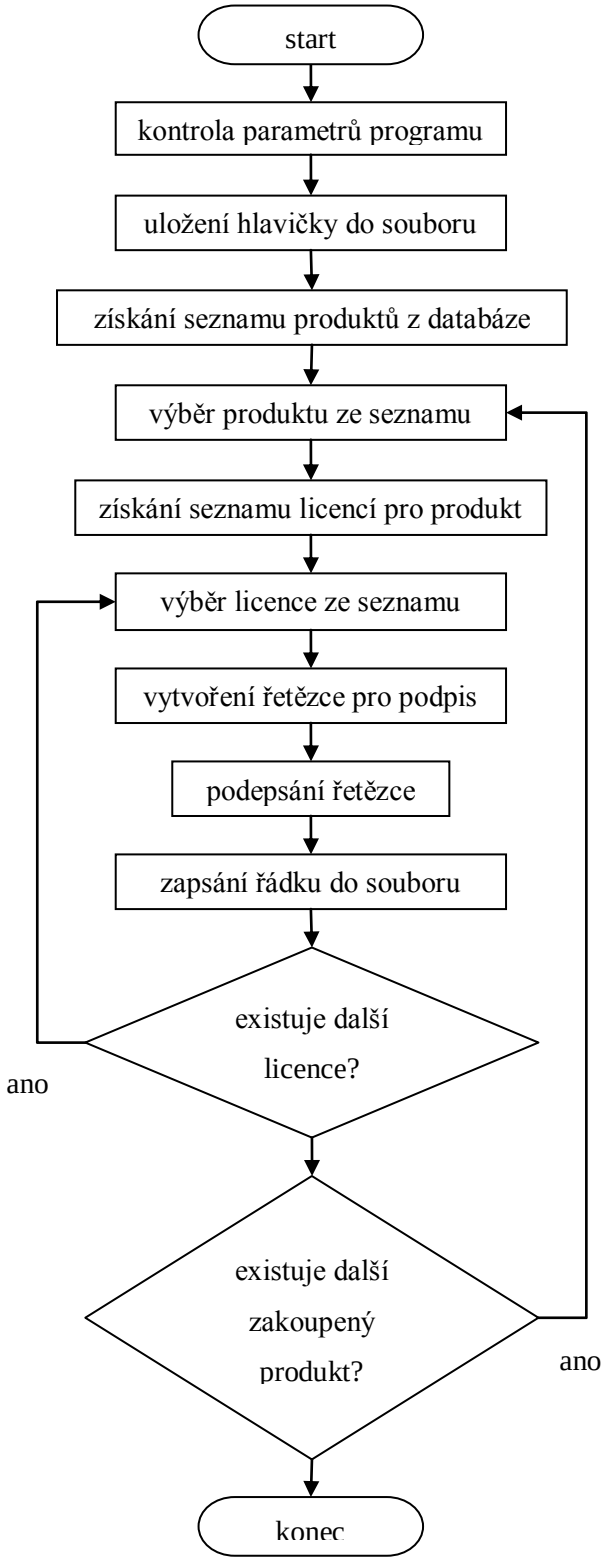


Diagram 2 : Vývojový diagram generátoru licenčních souborů

7.2.5.1 Přístup k databázi

Pro přístup k datům v databázi jsem použil knihovnu *mysql.h*. Z této knihovny jsem využil funkce pro připojení k databázi *mysql_real_connect* a pro získání dat z databáze pomocí dotazu *SELECT mysql_store_result*. Tato funkce vrací seznam záznamů vybraných dotazem a knihovna také nabízí funkci *mysql_fetch_row* pro procházení tohoto seznamu.

7.2.5.2 Funkce pro kryptografii

Jak jsem již zmínil v návrhu generátoru, pro vytvoření podpisu jsem použil knihovnu OpenSSL. Pomocí této knihovny jsem vytvořil strukturu, která obsahuje primární klíč. Tato struktura musí být k dispozici jako parametr funkce pro vytvoření podpisu *ECDSA_do_sign*. Tato funkce má jako další parametr otisk podepisovaného řetězce. Ten se vytvoří pomocí další funkce knihovny *SHA1*. Výstupem funkce *ECDSA_do_sign* je struktura obsahující dvě čísla podpisu.

7.2.5.3 Parametry programu

Nejprve proběhne kontrola parametrů programu, které musí být při spuštění programu zadány takto:

*./generátor -i customer_id -s server_host -sh server_hostid [-sport server_port] -v vendor *
[-vp vendor_path] [-vo optional_file_path] [-vport vendor_port]

Na pořadí parametrů nezáleží, není pevně dáno. Zkontroluje se přítomnost všech povinný parametrů a následně se zapíše hlavička do licenčního souboru. Kontrola i zápis se provádí v rámci funkce *parametry* s prototypem:

*int parametry (int pocet, char **param,FILE *f,int *id, int *hostid);,*

kde *pocet* je počet parametrů programu, *param* je pole těchto parametrů, *f* je ukazatel na licenční soubor pro zápis, *id* je identifikace zákazníka a *hostid* je pořadí parametru hostid serveru v poli parametrů. Parametry *pocet* a *param* jsou vstupní a *f, id* a *hostid* výstupní. Identifikace zákazníka je dále potřebná pro vybrání dat z databáze a hostid serveru se bude používat při podepisování souboru, proto musí být dostupné i mimo tuto funkci. Funkce vrací 0 při korektním provedení, jinak vrací příslušný chybový kód.

7.2.5.4 Vytvoření řádku

Tvorba řádku se provádí ve funkci *vytvor_radek* s prototypem:

*int vytvor_radek (MYSQL_ROW row,my_ulonglong rows,char **feat,unsigned long *len,char *hostid);,*

kde *row* je struktura obsahující jeden záznam o licenci z databáze, *rows* je počet položek záznamu, *feat* je vytvářený řádek, *len* je pole obsahující velikosti položek záznamu a *hostid* je hostid serveru. Parametry *row, rows, len* a *hostid* jsou vstupní a *feat* je výstupní parametr. Funkce nejprve vytvoří řetězec pro podpis, získá jeho otisk a z tohoto otisku vytvoří podpis. Z řetězce následně vymaže hostid serveru a místo něj do řetězce uloží podpis - dvě čísla v hexadecimálním tvaru převedená na řetězec. Tím je řádek vytvořen. Funkce vrací 0 při korektním provedení, jinak vrací příslušný chybový kód.

7.2.5.5 Zápis řádku do souboru

O zápis řádku do souboru se stará funkce *zapis_radek* s tímto prototypem:

*int zapis_radek(char **feat,FILE *f);,*

kde vstupní parametr *feat* je zapisovaný řádek a výstupní parametr *f* je ukazatel na soubor, do kterého se bude zapisovat. Pokud je řádek delší než 80 znaků, tak ho funkce rozdělí na části s maximální velikostí 80 znaků, které oddělí znakem '\' a odřádkováním. Výsledný licenční soubor tak bude přehlednější. Takto upravený řetězec pak zapíše do souboru. Funkce vrací 0 při korektním provedení, jinak vrací příslušný chybový kód.

7.3 Knihovna pro práci s licenčními soubory

Tuto knihovnu bude pro svou činnost potřebovat licencovaná aplikace, licenční manažer i vendor daemon. Knihovna pro práci s licenčními soubory musí implementovat funkce pro ověření podpisů souboru a funkce pro získání informací ze souboru. Implementačním jazykem je stejně jako u generátoru licencí jazyk C.

7.3.1 Ověření podpisů

Pro ověření podpisů v souboru je nejprve nutné vytvořit řetězec, který se podepisoval. Ten se vytvoří podle stejné syntaxe, která byla popsána v návrhu generátoru. Stejně tak jako v generátoru se vytvoří otisk pomocí hashovaní funkce a ověří se, jestli tento otisk koresponduje s digitálním podpisem. Tento postup se musí aplikovat na všechny řádky s licencemi. Pro ověření podpisu musí být k dispozici veřejný klíč.

7.3.2 Získání informací ze souboru

Při získávání informací ze souboru je nejprve nutné získat kompletní řádek informací pro server, daemon nebo licenci. Pokud jsou tyto informace rozděleny do více řádků, musí se složit dohromady. Následně se řádek musí rozdělit na jednotlivé položky a tyto položky se musí uložit do přehledné struktury. Pro uložení dat z licenčního souboru jsem navrhl tyto struktury:

Manažer	Vendor daemon	Licence na modul	Seznam licencí

struct SERVER{
char *host;
char *hostid;
int port;
char err;
}
	struct VENDOR{
	char *nazev;
	char *path;
	char *opt_path;
	int port;
	char err;
	}
		struct FEATURE{
		char *typ;
		char *nazev;
		char *vendor;
		char *verze;
		DATE *exp;
		int pocet;
		char *hostid;
		char *dup;
		char err;
		}
			struct LICENSE{
			int pocet;
			struct FEATURE *feature;
			char err;
			}

Parametr err v každé struktuře udává pozici případné nekorektní informace ze souboru. Pro uložení data expirace ze souboru jsem navrhl vlastní strukturu DATE, která obsahuje tři čísla reprezentující den, měsíc a rok. Implementační jazyk C sice nabízí strukturu pro uložení data, ale je v ní zbytečně mnoho položek (mimo data i čas, den v týdnu, …). Navíc se do této struktury nezapisuje přímo rok, ale rozdíl daného roku od roku 1900. To se nehodí pro uložení data ze souboru, protože ten může obsahovat rok 0 pro neomezenou licenci.

7.3.3 Implementace knihovny

7.3.3.1 Funkce pro práci se strukturami

V knihovně se nachází funkce pro inicializaci všech struktur a pro uvolnění struktur z paměti. Při inicializaci struktur se všechny ukazatele nastaví na hodnotu NULL a čísla na hodnotu 0.

7.3.3.2 Ukládání dat ze souboru do struktur

Nejprve je nutné získat informace ze souboru. K tomu slouží funkce *get_line*:

*int get_line(FILE *f,char **line);,*

která nenačítá jeden řádek ze souboru *f*, ale jeden záznam ze souboru do řetězce *line*. Jestliže je tedy např. záznam o serveru rozdělen v souboru na dva řádky, tato funkce ho vrátí jako jeden. Funkce vrací 0 pro korektní provedení, když narazí na konec souboru, tak EOF, jinak příslušný chybový kód. Dále jsou v knihovně tři funkce (*parse_server_line*, *parse_vendor_line*, *parse_feature_line*) na rozdělení parametrů v řetězci pro server, vendor daemon a licenci na modul. V každé funkci se ukládají data ze záznamu do příslušné struktury, ověřuje se jejich správnost a případně nepřítomnost povinných parametrů. V těchto funkcích se už ale nezkoumá správnost podpisů. Nakonec je pro zjednodušení práce uživateli knihovny implementována funkce pro načtení všech licencí na moduly do struktury LICENSE.

7.3.3.3 Ověření podpisů v souboru

O ověření podpisů v souboru se stará tato funkce:

*int check_file(char *key, FILE *f);*

kde *key* je veřejný klíč nutný pro ověření pravosti záznamů a *f* je kontrolovaný soubor. V této funkci jsou využity výše zmíněné funkce pro získání dat o serveru a licencích. Pro každý záznam o licenci se vytvoří řádek pro podpis a z něj se vytvoří otisk hashovaní funkcí SHA. Následně se porovná, zda daný otisk souhlasí s podpisem tohoto záznamu. K tomuto je opět použita knihovna OpenSLL a její funkce pro SHA a šifrování ECDSA.

7.3.3.4 Funkce pro práci s datem

Jelikož jsem vytvořil vlastní strukturu pro uchování data, implementoval jsem i funkce pro práci s ní. Jednak je to funkce pro inicializaci struktury *init_date*, která vynuluje všechny položky ve struktuře, dále funkce pro získání aktuálního lokálního data *get_act_date* a nakonec funkce pro převod data na řetězec a naopak *date_to_str* a *str_to_date*. Tento řetězec má formát rok-měsíc-den.

Závěr

Hlavním účelem této práce bylo vytvořit nástroje pro správu a distribuci licencí. Tyto nástroje budou používány v projektu Lissom. Knihovna pro práci s licenčními soubory pak bude použita v současně probíhajícím projektu, kterým je tvorba licenčního manažera a vendor daemona a zakomponování mechanizmu ověřování do aplikací licencovaných projektem Lissom.

K databázi licencí i k nástroji na jeho správu nemusí mít přístup nikdo jiný než administrátor prodejce. Z hlediska bezpečnosti by bylo nejlepší možností umístit server s databází do takové části sítě, která je bezpečně oddělena od vnější veřejné sítě. Do nástroje pro správu databáze jsem navíc zakomponoval ověřování identifikace uživatele, takže vstup je možný pouze s platným uživatelským jménem a heslem.

Stejný důraz jako na umístění databáze musí být kladen i na generátor licencí. V něm je totiž obsažen soukromý klíč, který slouží pro vytvoření podpisu licenčních souborů. Metodou zpětného inženýrství by bylo možné tento klíč z programu získat. Útočník by potom mohl generovat vlastní korektní licenční soubory a tím by se ztratil smysl celého ověřování platných licencí.

Knihovna pro práci s licenčními soubory neobsahuje žádné informace, které by mohly vést k prolomení ochrany licenčním serverem. Pracuje se sice v ní s veřejným klíčem, ale ten je do funkcí dodáván vendor daemonem. Navíc při dnešních technických možnostech nelze z veřejného klíče určit klíč soukromý.

Tato práce má široké možnosti pro budoucí vývoj. Do celého konceptu licencování mohou být přidány další funkce, které by zřejmě uvítal hlavně zákazník. Je to například půjčování licencí nebo platba za používání licence. Tyto nadstavby by ovšem vedli k modifikaci návrhu databáze a licenčních souborů, protože je potřeba pro ně ukládat speciální parametry k licencím. A změny v návrhu by samozřejmě zapříčinily nutnost změny implementace všech nástrojů.

Práce na bakalářské práci byla opravdu zajímavá. Všechny body zadání jsem splnil. Dozvěděl jsem se nové informace hlavně o licencování softwaru z pohledu prodejce. Dříve jsem se s licencováním setkával pouze z druhé strany, tedy z pohledu zákazníka.

Literatura

[1] H. Ward Classen: *A Practical Guide to Software Licensing for Licensees and Licensors, Second Edition: Model Forms and Annotations,* American Bar Association, 2008, 978-1590318492

[2] Gnu.org [online]. [cit. 2010-04-20]. What is Copyleft?. Dostupné z WWW: <http://www.gnu.org/copyleft/>.

[3] *Gnu.org* [online]. [cit. 2010-04-20]. Categories of free and non-free software. Dostupné z WWW: <http://www.gnu.org/philosophy/categories.html>

[4] *Opensource.org* [online]. [cit. 2010-04-20]. The Open Source Definition. Dostupné z WWW: <http://www.opensource.org/docs/osd>

[5] AUJEZDSKÝ, Josef. *www.root.cz* [online]. [cit. 2010-04-20]. Licence. Dostupné z WWW: <http://www.root.cz/specialy/licence/>.

[6] ZEMÁNEK, Jakub. *www.builder.cz* [online]. 07.05. 2001 [cit. 2010-04-22]. Ochrana software. Dostupné z WWW: <http://www.builder.cz/art/asembler/crack_uvod.html>.

[7] *Sixth Annual Global Software Piracy Study*: Business Software Alliance , 2009. 21 s. Dostupné z WWW: <http://global.bsa.org/globalpiracy2008/studies/globalpiracy2008.pdf>.

[8] *Warez.cz* [online]. 2009 [cit. 2010-04-24]. Jak se chrání software. Dostupné z WWW: <http://www.warez.cz/clanky/kategorie/cracking/>.

[9] NOVOSÁD, Jiří. *Fi.muni.cz* [online]. [cit. 2010-04-26]. Ochrany proti kopírování CD. Dostupné z WWW: <http://www.fi.muni.cz/usr/jkucera/pv109/2003/xnovosad.htm>.

[10] *Amsoft.cz* [online]. 2006 [cit. 2010-04-06]. *Co* je to softwarové pirátství?. Dostupné z WWW: <http://www.amsoft.cz/antipiracy/pirat.html>.

[11] JANSA, Lukáš. *www.pravoit.cz* [online]. 13.12.2006 [cit. 2010-04-06]. Softwarové pirátství ve firmě. Dostupné z WWW: <http://www.pravoit.cz/article/softwarove-piratstvi-ve-firme>.

[12] *FLEXnet licensing : End user guide*. [s.l.] : Macrovison, 2006. 168 s. Dostupné z WWW: <http://www.minitab.com/uploadedFiles/Shared_Resources/Documents/License_Management/flexnet_licensing_end_user_guide.pdf>.

[13] *LM-X End Users Guide*. [s.l.] : X-Formation, 2010. 32 s. Dostupné z WWW: <http://www.x-formation.com/lm-x_license_manager/enduser.pdf>.

[14] WENDT, Cris. *Flexerasoftware.com* [online]. 7.9.2009 [cit. 2010-05-01]. More on Token Based License Models. Dostupné z WWW: <http://blogs.flexerasoftware.com/ecm/2009/07/more-on-token-based-license-models.html>.

[15] *Cs.wikipedia.org* [online]. 2004 [cit. 2010-05-03]. PHP. Dostupné z WWW: <http://cs.wikipedia.org/wiki/PHP>.

[16] *SEC 1 : Elliptic Curve Cryptography*. [s.l.] : Certicom, 21.5.2009. 138 s. Dostupné z WWW: <http://www.secg.org/download/aid-780/sec1-v2.pdf>.

[17] *SEC 2 : Recommended Elliptic Curve Domain Parameters*. [s.l.] : Certicom, 27.1.2010. 33 s. Dostupné z WWW: <http://www.secg.org/download/aid-784/sec2-v2.pdf>.

[18] *Secure hash standard*. [s.l.] : National Institute of Standards and Technology, 17.5.1995. Dostupné z WWW: <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.

[19] *OpenSSL : The Open Source toolkit for SSL/TLS* [online]. c1999 [cit. 2010-04-15]. Dostupné z WWW: <http://www.openssl.org/>.

Seznam příloh

Příloha 1. CD

Soubory uložené na CD podle složek:

- Generátor
  - generator.c - zdrojový kód programu pro generování licenčních souborů
  - makefile - soubor pro kompilaci souboru generator.c
  - Dokumentace-generator.pdf - programová dokumentace ke generátoru
  - Manual-generator.pdf - návod na použití generátoru
- Knihovna
  - license.c - zdrojový kód knihovny pro práci s licenčními soubory
  - license.h - hlavičkový soubor k souboru license.c
  - check.c - zdrojový kód ukázkového programu pro vyzkoušení funkčnosti knihovny
  - makefile - soubor pro kompilaci ukázkového programu
  - Dokumentace-knihovna.pdf - programová dokumentace knihovny
- Správa
  - *.php,*.inc - zdrojové kódy nástroje pro správu licencí
  - Dokumentace-databaze.pdf - programová dokumentace k nástroji pro správu databáze
  - Manual-databaze.pdf - návod na zprovoznění nástroje pro správu databáze a jeho používání
- Technicka-zprava.pdf - text bakalářské práce