SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
MATERIÁLOVOTECHNOLOGICKÁ FAKULTA V TRNAVE
Katedra Aplikovanej Informatiky a Automatizácie

ELEKTRONICKÝ OBCHOD

BAKALÁRSKA PRÁCA
(web release)

JAROSLAV IMRICH

TRNAVA 2004

ABSTRAKT

Kľúčové slová: Elektronické obchodovanie. webová aplikácia. útok. ochrana
Úvodné časti práce obsahujú krátke oboznámenie s problematikou vývoja aplikácií určených pre oblasť elektronického obchodovania. Táto časť je doplnená popisom chýb, ktoré som objavil v komerčne používaných aplikáciách. Postupne sa cez analýzu aplikačných rozhraní pre aplikácie elektronického obchodovania dostávam až k popisu väčšiny známych útokov proti aplikáciám tohto druhu s webovým rozhraním. Analýza použiteľných technológií zas pomáha v utvorení celistvej predstavy o výhodách a nevýhodách jednotlivých technológií a je rozhodujúcim faktorom pre uskutočnenie správneho výberu. Zostavenie katalógu požiadaviek je teda už ovplyvnené výberom konkrétnych technológií a rozhraní. Pre zákaznícku časť obchodu som zvolil webové rozhranie používajúce skriptovací jazyk PHP a databázový server MySQL. Správcovská časť je vytvorená pomocou vývojového prostredia Borland Delphi 5. Oproti bežným prostrediam založeným na webových technológiách spočíva hlavná výhoda tohto riešenia najmä v tom, že výraznou mierou zvyšuje bezpečnosť a je používateľsky príjemnejšie. Popri vývoji zákazníckej časti som vytvoril ochranný systém webovej aplikácie a nazval som ho „Protector“. Tento systém pomáha včas identifikovať opísané útoky a zabrániť ich ničivému dopadu na aplikáciu tým, že odoprie prístup IP adrese, z ktorej útok prišiel. Samozrejme plnú kontrolu nad Protectorom má správca elektronického obchodu. Odolnosť webovej časti využívajúcej Protector som podrobil mnohým bezpečnostným testom a záznam jedného z nich je v prílohe. Práca môže teda slúžiť aj ako návod na vytvorenie bezpečnej webovej aplikácie.

ABSTRACT

Keywords: E-commerce. web application. attack. defense
First part of this work contains short introduction to the problems joined with development of applications designed to be used as an e-commerce applications. This part also includes description of errors I found in commercial applications. Step by step through the analysis of interfaces that could be possibly used to design such an application I get to description of known attacks applicable to web based applications. Analysis of common technologies helps to create a complete idea about advantages and disadvantages of each technology and it is important for making right decision. Designing of Requirements catalog is influenced by particular technologies and interfaces that have been chosen. Part of the product dedicated to customers uses web interface powered by PHP scripting language and database system MySQL. The part for Administrator was designed in Borland Delphi 5. Against conventional interfaces based on web technologies the main advantage of this solution is higher security level and it is more user-friendly. I also created defending system for web application and I called it “Protector”. It helps to identify described attacks and it avoids them to make any damage to application by banning IP address from which the attack came. Of course shop administrator has the full control over behavior of “Protector”. I made many security tests with web part of application and one of them is attached. I think this work also could be used as a guideline on creating secure web application.

OBSAH

ZOZNAM SKRATIEK A SYMBOLOV

WWW	World Wide Web
PHP	Hypertext Preprocessor
ASP	Active Server Pages
FSF	Free Software Foundation
GNU/GPL	General Public Licence
SSL	Secure Sockets Layer
POP3	Post Office Protocol
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
URL	Universal Resource Locator
HTML	HyperText Markup Language
CGI	Common Gateway Interface
SSI	Server Side Includes
JVM	Java Virtual Machine
SSJS	Server Side JavaScript
IIS	Internet Information Server
FI	Form Interpreter
DLL	Dynamic Link Library
SRBD	Systém Riadenia Bázy Dát
ODBC	Open DataBase Connectivity
SQL	Structured Query Language
BDE	Borland Database Engine
CSV	Comma Separated Values File

ÚVOD

Žiadne iné informačné médium neprešlo v posledných rokoch dvadsiateho storočia toľkými zmenami ako internet. Rozšíril sa do krajín celého sveta a postupne začína nahradzovať aj svojich najväčších konkurentov – rozhlas a televíziu. Prostredníctvom internetu dnes môžeme sledovať priame prenosy koncertov, počúvať rádio, posielať poštu či nakupovať.

V období jeho vzniku, vedeli internet používať iba počítačový experti, no k zvýšeniu jeho popularity výraznou mierou prispela služba WWW *(World Wide Web)*. Vizuálne poňatie webovských stránok ho umožňuje jednoducho používať aj ľuďom bez bližších znalostí sieťových a počítačových technológií. Masový príchod bežného používateľa bol prvým bodom zlomu, keď si význam internetu ako vplyvného informačného média uvedomili aj mnohé firmy, čo malo za následok premiestnenie podstatnej časti finančných prostriedkov určených na reklamu práve do tohto sektoru. V dnešnej dobe je už veľmi zriedkavé nájsť webovú stránku, na ktorej by nebol umiestnený reklamný banner.

1. ELEKTRONICKÉ OBCHODOVANIE

Vývoj technológií a príchod skriptovacích jazykov ako napríklad PHP *(Hypertext Preprocessor)* znamenali začiatok éry elektronického obchodovania – anglicky E-commerce. Spočiatku sa jednalo len o jednoduché aplikácie, kde si mal zákazník možnosť vybrať tovar z ponuky a jeho objednávka bola automaticky odoslaná e-mailom. V dnešnej dobe však nie je ničím neobvyklým vybraný tovar hneď aj zaplatiť, v čom napomáhajú služby elektronického bankovníctva. Ako vhodný príklad môže poslúžiť služba Tatrabanky zvaná Eliot Pay.

1.1 Typy elektronických obchodov

Prevádzkovanie elektronického obchodu a predávanie svojich produktov prostredníctvom internetu sa dnes pomaly stáva štandardom a za pár rokov bude určite neoddeliteľnou súčasťou podnikania. Ak sa v súčasnosti firma rozhodne takto sprístupniť ponúkaný tovar zákazníkom, má na výber z dvoch možností. Buď si zaplatí vývoj vlastnej aplikácie na mieru, čo je samozrejme spojené s vyššími, no jednorázovými nákladmi, alebo využije už naprogramované systémy, ktoré ponúkajú mnohé softvérové firmy.

1.1.1 Univerzálne komerčné aplikácie

Vhodným príkladom univerzálnej komerčnej aplikácie je slovenský systém Zoner Inshop3 vytvorený a prevádzkovaný ako služba serveru SlovakNet. Tento produkt je prezentovaný na webovej stránke www.inshop.sk. Svojím tvorcom je Zoner Inshop3 považovaný za komplexné riešenie, ktoré môže byť prispôsobené individuálnym požiadavkám väčšiny zákazníkov. Prevádzkovanie takéhoto typu obchodu je však sprevádzané stálymi mesačnými poplatkami (v prípade Zoner Inshop 3 je to 990 Sk mesačne) a poskytovateľ vyžaduje spojenie tejto služby aj s poskytovaním webhostingu, ktorý nie je práve najvýhodnejší.

Dovolím si tvrdiť, že nasadzovanie univerzálnych komerčných produktov so sebou nesie i bezpečnostné riziká. Ak sa útočníkovi podarí nájsť slabé miesto v ktoromkoľvek obchode využívajúcom daný systém, je viac než pravdepodobné, že tou istou chybou budú napadnuteľné i všetky ostatné obchody prevádzkované na tomto systéme. Chyba v takejto aplikácii, môže mať katastrofické dôsledky.

Aby som dokázal svoje tvrdenie, vykonal som test funkčnosti produktu Zoner Inshop3. Testu som podrobil obchodný dom www.hviezda.sk, ktorý pracuje na spomínanom systéme. Nesnažil som sa obchod napadnúť, aby som prevádzkovateľovi nespôsobil problémy, ale iba nájsť nejakú malú chybu. Moje hľadanie bolo úspešné. Po pokuse o nákup 500.000 kusov digitálnych fotoaparátov

som obdržal chybovú hlášku *„'error '8000ffff' - Catastrophic failure - /inshop/scripts/price_calculations.asp, line 210'“*. Som presvedčený, že sa jedná o extrémnu situáciu, s ktorou pri vývoji produktu nerátali, pretože chybová stránka je generovaná priamo webovým serverom spolupracujúcim s ASP *(Active Server Pages)*. Toto prekročenie veľkosti dátového typu malo za následok stratu všetkých informácií o tovare v nákupnom košíku. Takže aplikácia sa stala nepoužiteľnou. Rovnakú chybu som objavil aj v ďalších obchodoch pracujúcich na systéme Zoner Inshop3 ako napríklad jrc.inshop.sk, www.it-shop.sk alebo www.mip.sk. Som si takmer istý, že pokiaľ existujú chyby v užívateľskom prostredí, určite by sa dali nájsť aj bezpečnostné chyby, ktoré by mohli umožniť prienik do správy systému.

1.1.2GNU/GPL produkty

GNU je jedným z najúspešnejších projektov FSF *(Free Software Foundation)*, organizácie zameranej na tvorbu a šírenie voľného softvéru. Tento projekt si kladie za cieľ vytvoriť voľne šíriteľnú náhradu operačného systému unix a v súvislosti s ním vznikla i všeobecná zverejňovacia licencia GNU/GPL *(General Public Licence)*, ktorá sa vzťahuje na čoraz väčšie množstvo softvéru.

Softvérové produkty, na ktoré sa táto licencia vzťahuje majú rovnako ako komerčné produkty svoje výhody i nevýhody. Medzi nesporné výhody patrí najmä ich cena, pretože sú úplne zadarmo. Na vývoji takéhoto produktu sa podieľa množstvo programátorov. Každý priloží ruku k dielu, pomôže výsledný produkt vylepšiť. No voľne dostupné zdrojové kódy môžu byť častokrát i hrozbou pre bezpečnosť produktu. Pomáhajú totiž potenciálnemu útočníkovi obísť jednu z najzdĺhavejších častí útoku t.j. profilovanie aplikácie, keď sa pokúša zistiť ako aplikácia funguje, ako prebieha správa relácií, ako sú overovaní užívatelia atď. V prípade GNU/GPL produktu má útočník k dispozícii kompletné zdrojové kódy a je oveľa väčšia pravdepodobnosť, že nájde slabé miesto aplikácie. Teoreticky je možné, že po určitom čase sa produkt stane odolným voči väčšine útokov, prax však ukazuje, že nič nie je dokonalé a na tieto produkty sú neustále vydávané bezpečnostné záplaty a patche.

V oblasti elektronického obchodovania je bezpečnosť na prvom mieste, pretože je nutné zaručiť korektné priebehy finančných transakcií. Tiež dôležitú úlohu hrá ochrana chúlostivých osobných údajov ako napríklad čísiel kreditných kariet. V prípade softvéru publikovaného pod licenciou GNU/GPL nie je možné zaručiť tieto vlastnosti. Ak niekto nájde bezpečnostnú chybu, vždy existuje dilema či ju zverejní autorskému teamu alebo sa ju pokúsi sám zneužiť.

Istou možnosťou riešenia nastolených problémov GNU/GPL produktov by bolo ochrániť elektronické obchody využívajúce daný produkt tým, že nebude nikde zverejnený ich zoznam. Otázkou na zamyslenie však ostáva, či by sa niekto rozhodol pre svoju predajňu použiť softvér, o ktorom autori síce tvrdia, že je využívaný mnohými významnými internetovými obchodmi, no nikde nemá zverejnené referencie.

1.2 Typy aplikačných rozhraní elektronických obchodov

1.2.1Webové rozhranie

Väčšina dostupných systémov pre elektronické obchodovanie je založená na webovom rozhraní. Najväčšou výhodou tohto rozhrania je jednoznačne jednoduchosť a v neposlednom rade aj dostupnosť softvérových produktov potrebných k prevádzke, ktoré sú väčšinou šírené pod licenciou GNU/GPL. Práca s internetovým prehliadačom patrí v súčasnosti medzi základné schopnosti používateľov osobných počítačov. Preto, ak je webové rozhranie navrhnuté prehľadne, nemalo by byť vykonanie objednávky problémom ani pre človeka so základnými znalosťami. Aplikácie pracujúce na webe sú väčšinou určené pre širokú skupinu zákazníkov. Do tejto skupiny patria aj zákazníci, ktorí sa na stránku predajne dostali náhodou, napríklad kliknutím na odkaz v niektorom vyhľadávači.

1.2.2Špeciálne systémové rozhranie

Ďalšou možnosťou je aplikácia pre vybraný operačný systém navrhnutá v špecializovanom vývojovom prostredí - Microsoft Visual C++ pre operačný systém Windows, Borland Kylix pre Linux, a iné. Aby bol zákazník schopný nakupovať v takomto obchode, je potrebné, aby mu prevádzkovateľ obchodu poskytol klientskú časť aplikácie. Túto zákazník nainštaluje na svoj osobný počítač a môže prostredníctvom nej nakupovať. Toto riešenie je však menej časté a je vhodné najmä pre predajne, kde môžu nakupovať iba vybraní klienti. Príkladom môžu byť veľkoobchody, v ktorých môžu nakupovať iba súkromní podnikatelia.

1.2.3Kombinované rozhranie

Prednosti obidvoch opísaných druhov elektronických obchodov v sebe spája tretí typ pozostávajúci z klientskej časti s webovým rozhraním a správcovskej časti navrhnutej v niektorom z už spomínaných špecializovaných vývojových prostredí. Toto riešenie zachováva jednoduchosť a širokú dostupnosť klientskej časti a zároveň vďaka neexistencii webového rozhrania pre správcovskú časť významne znižuje riziko, že útočník prenikne do správy systému.

1.3 Časti aplikačných rozhraní elektronických obchodov

1.3.1 Používateľská časť

Používateľská časť elektronického obchodu je jedným z najdôležitejších prvkov, pretože reprezentuje firmu pred zákazníkom. Dojem, ktorý urobí vzhľad aplikácie na zákazníka vo výraznej miere ovplyvňuje jeho rozhodovanie pri nakupovaní. V konkurenčnom prostredí víťazí prehľadná aplikácia, v ktorej sa dá rýchlo vykonať objednávka bez zdĺhavých registrácií alebo bez zbytočného „preklikávania sa“ cez neprehľadné menu.

V dnešnej dobe, keď sa do popredia dostávajú služby bánk spojené s internetom, môže byť rozhodujúcim faktorom aj podpora rôznych druhov platby. Úlohou návrhára aplikácie elektronického obchodu je vytvoriť vhodné prostredie pre zavedenie týchto možností platby i keď prevádzkujúca firma nemusí tieto možnosti platby z rôznych príčin podporovať.

1.3.2 Správcovská časť

Správcovská časť musí poskytovať prevádzkovateľovi obchodu jednoduché nástroje pre zmenu základných vlastností obchodu, aby nemusel v prípade takýchto malých zmien zasahovať do produktu programátor. Takýmito údajmi sú napríklad názov a adresa firmy, telefónne čísla a rôzne kontaktné údaje.

Neodmysliteľnou úlohou tejto časti je tiež kompletná a podrobná správa jednotlivých položiek sortimentu, kde prevádzkovateľ môže meniť ceny alebo popisy jednotlivých tovarov, pridávať nový tovar, odoberať neaktuálny tovar. Táto časť môže byť obohatená i o štatistické informácie o jednotlivých poskytovaných produktoch, o ktoré sa môže prevádzkovateľ oprieť pri plánovaní budúcej činnosti obchodu.

Tiež bežnou časťou, ktorú správcovská časť elektronických obchodov obsahuje je prehľadová časť objednávok, či už vybavených alebo ešte nevybavených.

Ďalšie funkcie ako napríklad správa zákazníkov, kontrola a obmedzovanie prístupov alebo export a import údajov závisia najmä na konkrétnych požiadavkách zadávateľa ako i na predstavivosti a zručnosti návrhára a programátora systému.

2. BEZPEČNOSŤ ELEKTRONICKÝCH OBCHODOV

Bezpečnosť aplikácií pre elektronické obchodovanie je veľmi háklivá a diskutovaná téma, pretože tieto aplikácie obsluhujú finančné transakcie a uchovávajú chúlostivé osobné údaje ako napríklad čísla bankových účtov, kreditných kariet atď. Zabezpečenie aplikácie by preto malo tvoriť primárnu sféru záujmu už v prvotných fázach návrhu a vývoja. Mnoho návrhárov a programátorov sa však mylne domnieva, že ich produkty sú nenapadnuteľné, pretože bežia na dobre nakonfigurovaných a chránených serveroch. Pravdou však je, že i jednoduchá stránka spolupracujúca s databázou poskytuje rozhranie pre komunikáciu so serverom a teda môže umožniť útok na databázu či na server samotný. Oblasť bezpečnosti webových aplikácií je však veľmi komplexná a neodmysliteľne súvisí i s konfiguráciou jednotlivých služieb servera, no v tejto práci sa zamieram len na útoky vykonateľné proti aplikácii samotnej, ktoré by mohli viesť k získaniu dát, s ktorými aplikácia pracuje. Predpokladajme teda, že server, na ktorom bude aplikácia pracovať je odolný voči útokom.

Najčastejšími útokmi na webové aplikácie sú:

- útok na overovací mechanizmus
- útok na správu stavu relácie
- útok na súbory cookie
- útok na validáciu vstupu

Všetky ďalej popísané typy útokov majú bezprostredný vplyv na výslednú kvalitu a funkčnosť konečného produktu, a preto by každá aplikácia slúžiaca ako elektronický obchod mala byť minimálne voči týmto typom útokov odolná.

2.1 Útok na overovací mechanizmus

Úlohou overovacieho mechanizmu je vyžiadať od klienta jeho používateľský login a heslo, overiť správnosť týchto údajov a na základe výsledku overenia buď povoliť prístup k aplikácii alebo prístup odoprieť.

2.1.1 Útok na kontrolnú otázku

Najjednoduchšou technikou útoku na tento mechanizmus je poznať správne údaje. Ak poznáme meno používateľa, môžeme sa pokúsiť prihlásiť s ľubovoľným heslom. Mnohé aplikácie ponúkajú po zadaní nesprávneho hesla kontrolnú otázku, ktorá ak je správne zodpovedaná, zobrazí sa používateľské heslo. Ak útočník pozná používateľa, ktorého identitu chce získať, má väčšiu šancu, že sa mu podarí na

kontrolnú otázku odpovedať, pretože kontrolnú otázku rovnako ako aj odpoveď na ňu volí sám používateľ pri registrácii. Na tento útok netreba nijaké zvláštne vedomosti a preto už dnes väčšina aplikácií nepoužíva kontrolnú otázku, ale zasiela heslo na e-mail, ktorý používateľ tiež vyplnil pri registrácii.

2.1.2 Útok hrubou silou

Za predpokladu, že server používa na prenos overovaných údajov šifrované spojenie prostredníctvom protokolu SSL *(Secure Sockets Layer)* môžeme sniffovacie techniky vylúčiť. Ostáva teda možnosť útoku tzv. hrubou silou (z angl. brutal force attack). Pri tomto type útoku sa útočník pokúša prostredníctvom špeciálnej aplikácie generovať užívateľské mená a heslá a zasielať ich overovaciemu mechanizmu. Ak aplikácia nie je voči tomuto typu útoku programovo ošetrená je pravdepodobnosť prielomu veľmi vysoká. Veľmi vydarenou aplikáciou na vykonávanie takýchto útokov je Brutus AET2, ktorý si poradí nielen s webovými formulármi ale dokonca aj s protokolmi POP3 *(Post Office Protocol)* či FTP *(File Transfer Protocol)*.

2.2 Útok na správu stavu relácie

Primárnym dôvodom pre sledovanie stavu relácie sú aplikácie, ktoré vyžadujú overovanie používateľov. Keď je identita používateľa overená, server musí prijímať ďalšie požiadavky od daného používateľa a zároveň odmietnuť požiadavky od používateľa, ktorý ešte nie je overený.

Každá relácia má svoj vlastný jedinečný identifikátor, ktorý generuje webový server alebo sa o jeho generovanie postará sám programátor a implementuje do aplikácie príslušné algoritmy. Server odošle po prihlásení klientovi hodnotu tohto identifikátoru a ten sa ním následne preukazuje po celú dobu spojenia. Tento postup je zaužívaným štandardom a bol zavedený kvôli hlavnému nedostatku protokolu HTTP *(Hypertext Transfer Protocol)* – absencii stavovosti.

2.2.1Identifikátor ako súčasť URL

Existuje viacero spôsobov ako vymieňať hodnotu identifikátoru stavu relácie medzi klientom a serverom. Najjednoduchšou možnosťou je posielať túto hodnotu ako súčasť URL *(Universal Resource Locator)*. Hodnota identifikátoru je potom viditeľná v paneli s adresou vo webovom prehliadači, kde môže byť napríklad:

```
http://www.obchod.sk/view.php?id=8b620a4f77489d0dcaaeebe8696ec788
```

Táto možnosť je však aj najľahšie napadnuteľná, pretože stačí prepísať hodnotu identifikátoru, čím sa okamžite mení naša totožnosť. Záleží len na aplikačnej logike ako sa s touto skutočnosťou vyrovná. Takéto uchovávanie hodnoty identifikátoru nie je najvhodnejšie, preto sa v praxi viac používa uchovávanie v súboroch cookie a v hlavičkách HTTP.

2.2.2Identifikátor v súbore cookie

Cookie je malý textový súbor, ktorý je vytvorený na požiadavku webového serveru na klientskom počítači, ak mu to dovolí nastavenie prehliadača. Tento súbor slúži pre server ako úložisko dát a klient ho sprístupňuje serveru pri každom ďalšom pripojení. Podrobnejším popisom útoku na perzistentný súbor cookie sa zaoberá kapitola 2.3.

2.2.3Identifikátor ako súčasť HTTP hlavičky

Častá je aj možnosť ukladania tzv. krátkodobého (neperzistentného) cookie, kedy sa hodnoty nezapisujú na disk ale sú súčasťou HTTP hlavičiek. Na zobrazenie týchto hodnôt postačí bežný plug-in pre webový prehliadač Mozilla, avšak na ich editáciu je potrebný špeciálny program akým je napríklad Achilles.

Na prvý pohľad by sa mohlo zdať, že ukladanie hodnoty identifikátoru stavu relácie do súboru cookie alebo jej posielanie ako súčasti HTTP hlavičiek nie je o nič bezpečnejšie než prvý spomínaný spôsob, pretože sa taktiež dá táto hodnota editovať. Opak je však pravdou. Dovolím si tvrdiť, že mnohí pokročilí používatelia a dokonca ani mnohí weboví návrhári nevedia, že sa dajú zmenou HTTP hlavičiek meniť dokonca aj dáta posielané metódou POST, ktoré sú inak neviditeľné. Profesionálna aplikácia sa však nemôže spoliehať na neznalosť používateľov a potenciálnych útočníkov. Preto sa v praxi využíva databáza identifikátorov relácií, ktorá obsahuje zoznam všetkých relácií a s jej pomocou je možné dôsledne kontrolovať jednotlivé relácie.

2.3 Útok na súbory cookie

Útok na súbory cookie je úzko spätý s útokom na správu stavu relácie. Ak chce útočník vystupovať ako iný prihlásený používateľ, musí získať hodnotu jeho identifikátoru relácie. Ako som už spomenul, táto hodnota je bežne ukladaná do súborov cookie. K zachyteniu súborov cookie sa dá využiť aj už tiež spomínaný nástroj Achilles, ktorý funguje ako proxy server. Využitie tejto funkcie nie je pri každom útoku jednoduché ale tiež nie je nemožné. Môže ju využiť napríklad správca podnikovej siete.

Používaním starej verzie webového prehliadača bez nainštalovaných ochranných záplat môže používateľ sprístupniť útočníkovi cookie súbory bez toho, aby o tom vedel. Bennet Haselton a Jamie McCarthy z firmy Peacefire zverejnili skript využívajúci bezpečnostnú chybu v prehliadači Microsoft Internet Explorer. Tento skript získa všetky súbory cookie z klientského počítača. Stačí aby používateľ klikol na odkaz umiestnený na stránke. Obsah súborov cookie je potom prístupný pre operátorov webového servera [1].

2.4 Útok na validáciu vstupu

Pri útokoch na validáciu vstupu sa útočník pokúša odosielať dáta, ktorých prijatie aplikácia neočakáva. Aplikácia bežne vykoná kontrolu správnosti užívateľského vstupu, pri ktorej sa snaží zistiť či sú dáta správne. Validácia dát môže byť veľmi zložitá, pretože musí počítať jednak s rozsahom dátových typov, ktorý by sa nemal prekročiť a tiež nesmie dovoliť zadanie znakov, ktoré majú pri interpretácii programu špeciálny význam a mohli by spôsobiť napríklad vypísanie časti zdrojového kódu alebo preskočenie kontrolných procedúr.

2.4.1 Validácia vstupu na strane klienta

Validáciu vstupu je možné prevádzať buď na strene klienta alebo na strane serveru. Na strane klienta nachádza svoje uplatnenie najmä špeciálna odroda jazyka Java vytvorená firmou Netscape zvaná JavaScript. Pomocou JavaScript-u je možné vykonať kontrolu zadaných dát ešte predtým, než sa odošlú na spracovanie serveru. Chybou tohto riešenia však je, že spracovávanie skriptov napísaných v JavaScript-e sa dá vypnúť resp. veľmi ľahko obísť.

2.4.2 Validácia vstupu na strane serveru

Obecne platí, že žiadne dáta prijaté od klienta sa nemôžu považovať za dôveryhodné. Preto musí všetky hodnoty overené pomocou procedúr JavaScript-u overiť ešte raz server. Na túto skutočnosť sa často zabúda, čoho dôkazom je aj výsledok testu, ktorému som podrobil internetový obchod www.pancena.sk.

Jednoduchou úpravou URL som si bez problému objednal neexistujúci tovar. Tvorca tejto aplikácie mylne považuje všetky hodnoty prijaté od klienta za správne. Za povšimnutie však stojí, že striktne overuje správnosť zadanej e-mailovej adresy.

Správne naformátovaný „neplatný“ vstup môže byť využitý k spusteniu preplnenia vyrovnávacej pamäte, k opusteniu koreňa webového serveru, k útoku spustením vloženého kódu SQL alebo dokonca k spusteniu príkazov operačného systému [1].

3. TECHNOLÓGIE PRE TVORBU ELEKTRONICKÉHO OBCHODU

Skôr než sa pristúpi k návrhu samotnej aplikácie elektronického obchodu je nutné určiť, aké aplikačné rozhrania bude aplikácia používať. Pre každé aplikačné rozhranie, či už webové alebo systémové existuje množstvo použiteľných technológií. Správny výber konkrétnych technológií má zásadný vplyv na funkčnosť, prevádzkové náklady ako aj na výslednú cenu produktu.

3.1 Technológie webových rozhraní

V roku 1990, keď bola vytvorená dnes masovo používaná služba WWW, bol jej nosným pilierom vyznačovací jazyk HTML *(HyperText Markup Language)*, s pomocou ktorého sa dali vytvárať len statické stránky. Určité zlepšenie situácie prinieslo rozhranie CGI *(Common Gateway Interface)*, ktoré umožňuje na požiadavku klienta spustiť na webovom serveri program napísaný v ľubovoľnom programovacom jazyku (najčastejšie Perl, Shell script, C, C++), ktorý vygeneruje stránku s aktuálnymi údajmi. V rovnakom čase sa začali presadzovať aj serverové vsuvky SSI *(Server Side Includes)*, ktoré boli zapísané v bežnej statickej stránke, no pred odoslaním klientovi ich server vykonal. Takto sa dal do stránky vložiť napríklad aktuálny čas. Tieto technológie však majú spoločnú nevýhodu. Celá aplikačná logika je presunutá na server, čo pre veľké servery môže znamenať príliš veľkú záťaž. Preto sa začali hľadať riešenia ako do procesu spracovania údajov zapojiť aj klienta, no zároveň zachovať existujúce štandardy ako napríklad protokol HTTP. V roku 1996 predstavila firma Sun Microsystems svoje riešenie problému – jazyk Java. Najväčšou výhodou javy je jej platformová nezávislosť. Java aplety môžu bežať na počítači s ľubovoľným operačným systémom, pre ktorý existuje interpret jazyka tzv. JVM *(Java Virtual Machine)* [2].

3.1.1 JavaScript

V tom istom roku predstavila firma Netscape technológiu založenú na jazyku Java nazvanú JavaScript. JavaScript je programovací jazyk interpretovaný na strane klienta. Z toho vyplývajú aj jeho možnosti a obmedzenia. Keďže sa JavaScript spúšťa na počítači klienta, z pochopiteľných dôvodov nedokáže uchovávať, meniť alebo vymazávať súbory nachádzajúce sa na webovom serveri. Dokáže síce pracovať s cookie súbormi na klientskom počítači, no to pre tvorbu rozsiahlych webových aplikácií nestačí. Odhliadnuc od spomenutej nevýhody, je JavaScript pružným programovacím jazykom, s ktorým sa dá na stránky umiestniť nespočetné množstvo grafických a iných efektov [3]. Jeho primárne uplatnenie je teda pri vytváraní atraktívneho používateľskeho prostredia.

3.1.2SSJS

Úspech JavaScriptu bol tak obrovský, že firma Netscape pristúpila k jeho implementácii na strane serveru. Netscape Enterprise Server tak získal oproti konkurenčným produktom veľkú výhodu, ktorou je programovacie prostredie schopné spolupracovať s databázami a poskytujúce rovnaké možnosti ako rozhranie CGI, avšak s oveľa väčšou jednoduchosťou. Toto riešenie sa šírilo pod názvom LiveWire no dnes má výstižnejšie meno – SSJS *(Server Side JavaScript)*.

3.1.3ASP

Reakcia firmy Microsoft bola takmer okamžitá. Ich webový server IIS *(Internet Information Server)* bol obohatený o technológiu ASP. ASP je obdobou SSJS. Ako programovací jazyk možno použiť VBScipt alebo JScript, čo je klon JavaScriptu vyvorený firmou Microsoft. Konkurenčné systémy samozrejme nie sú kompatibilné. ASP používa iné značky než SSJS. Nevýhodou technológie ASP je, že rovnako ako IIS beží iba na operačnom systéme Windows.

3.1.4PHP

Mnohé nedostatky technológií SSJS a ASP odstraňuje technológia PHP. Jej najväčšou výhodou je, že je šírená ako freeware produkt a je multiplatformová. Existuje verzia pre Unix i pre Windows a používanie PHP dokonca nie je viazané na žiadny konkrétny webový server [2]. Vynikajúco beží na všetkých najznámejších serveroch, no z vlastnej skúsenosti považujem za najvýkonejšie jeho spojenie s webovým serverom Apache na platforme Unix/Linux.

Vývoj PHP sa začal v roku 1994, keď Rasmus Lerdorf vytvoril v jazyku Perl jednoduchý systém kontroly prístupu k jeho stránkam. Neskôr ho prepísal do jazyka C, pretože spúšťanie perlového skriptu zaťažovalo webový server. Tento systém začali využívať aj ďalší užívatelia serveru a tak ho autor rozšíril, doplnil o dokumentáciu a uvoľnil pod názvom Personal Home Page Tools. Po pozitívnej reakcii zo strany užívateľov vytvoril ďalší program, ktorý umožnil sprístupniť na webe databázy a nazval ho FI *(Form Interpreter)*. Svetové rozšírenie zabezpečilo až spojenie oboch systémov, čím vzniklo PHP/FI 2.0 [2]. V súčasnosti sa pripravuje už verzia PHP 5.0, ktorá má kompletne prepísané jadro a spolu s pôvodným autorom ju vyvíja programátorský team pod názvom Zend. Odporúčaný názov pre PHP je Hypertext Preprocessor.

PHP je možné skompilovať ako spustiteľný program na ľubovoľnej platforme a následne ho používať ako CGI skript. Táto možnosť zaručuje jeho použiteľnosť na ľubovoľnom webovom serveri, ktorý podporuje rozhranie CGI. Používanie PHP

týmto spôsobom však zaťažuje webový server, lebo interpret jazyka je znovu spúšťaný vždy, keď je nutné vykonať nejaký skript. Preto existuje aj druhá možnosť. Skompilovať PHP ako modul pre server apache a ten je spolu so serverom neustále zavedený v pamäti. V systéme Microsoft Windows si nájde uplatnenie kompilácia PHP na DLL *(dynamic link library)* - dynamicky linkovanú knižnicu, ktorá je taktiež neustále zavedená v pamäti.

3.2 Technológie systémových rozhraní

Pri vývoji elektronických obchodov určených pre použitie v konkrétnom operačnom systéme sa dnes využívajú najmä programovacie jazyky a vývojové prostredia štvrtej generácie, kde sa okrem priameho zápisu programového kódu používa aj vizuálne programovanie ako kreslenie dialógových okien alebo grafický návrh väzieb medzi objektmi s následným automatickým vygenerovaním príslušnej časti programu. Pri jazyku štvrtej generácie už nie je nutná znalosť určitého programovacieho jazyka, skôr je treba chápať spôsob grafického vyjadrenia danej myšlienky. Príkladom vývojových prostredí štvrtej generácie sú Microsoft Access, Borland Delphi alebo CA Visual Objects [5].

3.2.1 Borland Delphi

Vývoj programovacieho prostredia Delphi začal v roku 1983, no vizuálnym prostredím sa stalo až v roku 1995. Delphi 1 znemalo prielom v oblasti programovania pre dovtedy ešte 16-bitový systém Windows. Príchod 32-bitového operačného systému Microsoft Windows 95 však znamenal veľké zmeny. Borland na ne bleskovo zareagoval a už v roku 1996 vydal Delphi 2. V nasledujúcich verziách boli postupne pridávané ďalšie a ďalšie vylepšenia ako napríklad inteligentný dopĺňač kódu Code Insight, vylepšená podpora databáz a iné. Delphi však nie je jediným vizuálnym prostredím na trhu. Samotná firma Borland ponúka hneď ďalšie dva nástroje pre vývoj aplikácií pre Windows. Je to C++ Builder a J Builder. Samozrejme k dispozícii sú aj Visual Basic a Visual C++ od konkurenčného Microsoftu [6]. Delphi má však vďaka svojej skvelej databázovej podpore, možnosti jednoduchého vývoja aplikácií typu klient/server a veľkému množstvu voľne dostupných komponentov mnoho priaznivcov a udržuje krok s konkurenčnými produktmi.

3.2.2 Borland Kylix

Vizuálne vývojové prostredia nie sú doménou len operačného systému Windows. Produkt firmy Borland zvaný Kylix je určený pre operačné systémy Unix/Linux, presnejšie pre nimi hojne využívaný grafický server Xwindow. Keďže tento grafický server je voľne šíriteľný, poskytuje aj Borland vývojové prostredie

Kylix vo viacerých verziách. Jednou z nich je aj verzia Open, ktorá je k dispozícii zadarmo na stiahnutie z internetu. Táto verzia je však ochudobnená o technickú podporu a obsahuje len časť komponentov komerčnej verzie. Šikovný používateľ si však môže z internetu stiahnuť množstvo voľných komponentov. Keďže je prostredie Kylixu totožné s prostredím Delphi nie je problém aplikácie navrhnuté pre operačný systém Windows prekompilovať v Kylixe a používať ich pod operačným systémom Unix.

3.3 Databázové systémy

Databázu si môžeme predstaviť ako miesto na pevnom disku kam sa ukladajú a odkiaľ sa následne čítajú potrebné údaje. Tieto údaje nazvime bázou dát. Pre uľahčenie manipulácie a tiež pre umožnenie vyhľadávania v týchto údajoch, musí byť v tvorení bázy dát nejaký systém. Preto akýkoľvek prístup k báze dát neriadi používateľ ale zabezpečuje ho špeciálny program zvaný SRBD *(Systém Riadenia Bázy Dát)*. Databázový systém teda tvorí SRBD a samotná báza dát. Bežnému používateľovi sú skôr známe konkrétne názvy SRBD ako napríklad Oracle, Sybase, Informix, MS SQL Server či InterBase. Tieto databázové servery patria do skupiny komerčného softvéru, ktorého ceny sa rádovo pohybujú v desiatkach väčšinou však v stovkách tisíc korún. Pre bežného užívateľa existujú aj nemenej kvalitné freeware produkty ako napríklad MySQL alebo PostgreSQL.

Spôsob akým komunikuje SRBD s ostatnými aplikáciami je veľmi podobný komunikácii s webovým serverom. SRBD bývajú v dnešnej dobe najčastejšie spustené ako daemon systému Unix/Linux alebo služba systému Windows, čo znamená, že na určitom porte očakávajú požiadavky od klientov a na tie následne odpovedajú. Aj v tomto prípade teda funguje osvedčený model klient/server. Preto je pre SRBD v bežnej praxi zaužívaný názov databázový server.

Určitou nevýhodou je, že každý databázový server má vlastný protokol, ktorým komunikuje s klientom. Ak by mala aplikácia komunikovať s viacerými databázovými servermi, musela by poznať komunikačný protokol každého z nich. Preto firma Microsoft vytvorila rozhranie ODBC *(Open DataBase Connectivity)*, ktoré umožňuje štandardný prístup k ľubovoľným dátovým zdrojom. To slúži ako sprostredkovateľ komunikácie medzi klientskou aplikáciou a databázovým serverom. Aplikácia s rozhraním ODBC komunikuje jednotne a ODBC ovládač potom odovzdá požiadavku databázovému serveru pomocou správneho protokolu [2].

Pre zadávanie požiadaviek na databázový server aplikácie najčastejšie používajú jazyk SQL *(Structured Query Language)*. Databázové servery podporujú štandardné SQL príkazy definované normou a mnohí výrobcovia vytvorili aj vlastné

rozšírenie jazyka. Známym rozšírením je PL/SQL od Oracle. I keď je jazyk SQL veľmi jednoduchý, poskytuje všetky potrebné funkcie pre vkladanie, modifikáciu i mazanie dát.

3.3.1MySQL

MySQL je veľmi rýchly a robustný databázový server, ktorý beží pod mnohými operačnými systémami. Samozrejmosťou je jeho dostupnosť pre systémy Unix/Linux i Windows. Za svoju rýchlosť vďačí najmä tomu, že nepodporuje transakčné spracovanie dát, triggre a stored procedúry, bez čoho sa však s trochou programátorskej zručnosti dá zaobísť. Na MySQL sa vzťahuje dvojitá licencia. Môže byť používaný ako Open Source produkt s licenciou GNU/GPL, no na komerčné použitie je nutné zakúpiť licenciu, ktorej cena je oproti konkurenčným produktom neporovnateľne nižšia.

Podpora MySQL je v súčasných verziách jazyka PHP už štandardom. Niet sa čomu diviť, pretože webový server Apache s modulom jazyka PHP a MySQL ako databázovým serverom tvoria v súčasnom internete najpoužívanejšiu trojicu technológií. Všetky tieto produkty sú šírené voľne pod licenciou GNU/GPL a poskytujú minimálne rovnaké, ba v mnohých smeroch dokonca väčšie možnosti ako komerčné produkty.

4. KATALÓG POŽIADAVIEK

Kapitola bola z verzie web release vypustená.

5. DÁTOVÝ MODEL

Kapitola bola z verzie web release vypustená.

6. FUNKČNÝ MODEL

Kapitola bola z verzie web release vypustená.

7. IMPLEMENTÁCIA NAVRHNUTÉHO MODELU

Pre navrhovaný elektronický obchod som sa rozhodol použiť kombinované aplikačné rozhranie popísané v kapitole 1.2.3. Pripomeniem len, že toto riešenie pozostávajúce z klientskej časti s webovým rozhraním a správcovskej časti navrhnutej v špecializovanom vývojovom prostredí ponúka jednoduchosť a širokú dostupnosť klientskej časti a vďaka neexistencii webového rozhrania pre správcovskú časť znižuje riziko, že by sa útočníkovi podarilo preniknúť do správy systému.

Zákaznícka časť prostredia elektronického obchodu je vytvorená s využitím skriptovacieho jazyka PHP a ako databázový server som zvolil SRBD MySQL. Správcovská časť systému je vytvorená v prostredí Borland Delphi 5 klasickými postupmi využívanými pri vývoji databázových aplikácií a pripája sa na databázový server prostredníctvom ovládača ODBC, ktorý je k dispozícii na stiahnutie z www.mysql.com.

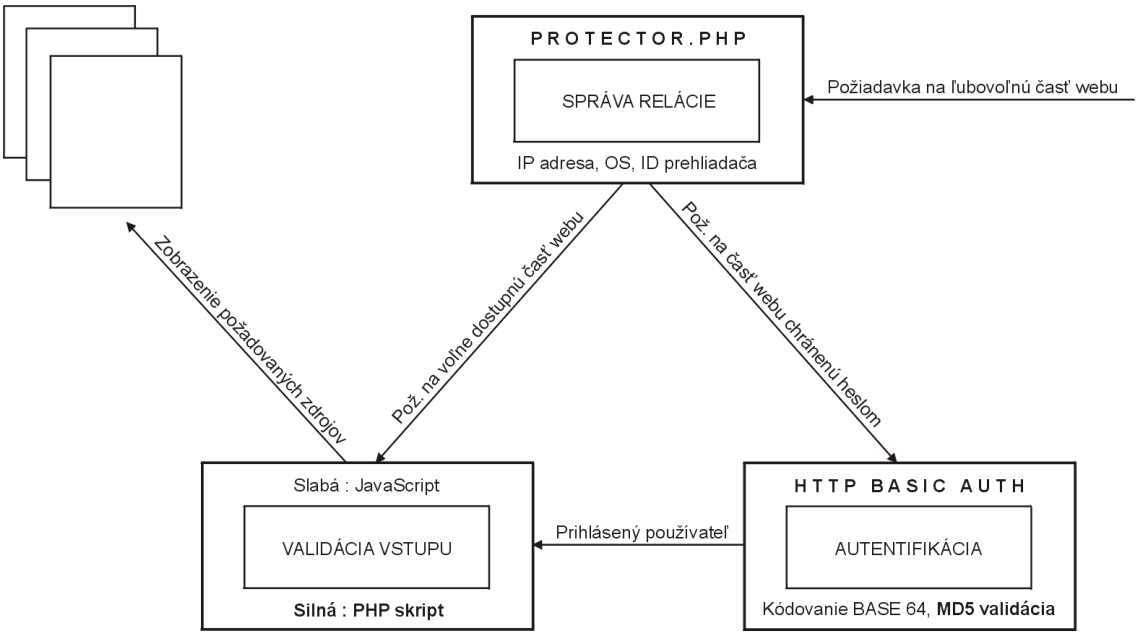
7.1 Zákaznícka časť elektronického obchodu

Vytvorená zákaznícka časť elektronického obchodu sa navonok výrazne nelíši od ostatných v súčasnosti dostupných produktov. Zákazník vkladá vybraný tovar do nákupného košíka, ktorý zobrazuje vždy aktuálne informácie o cene nákupu a poskytuje možnosť rýchlej objednávky bez nutnosti registrácie. Samozrejmosťou je aj možnosť registrácie zákazníka. Výhoda registrovaného zákazníka spočíva najmä v tom, že pri ďalšej objednávke sú automaticky vyplnené jeho osobné údaje ako napríklad meno a adresa, teda zákazník ušetrí čas potrebný na vyplnenie kontaktných údajov.

Webovú časť elektronického obchodu som navrhol s vedomím, že jeho prevádzkovateľ nemá prístup ku konfigurácii serveru, na ktorom bude používaný. To znamená, že napríklad nemôže konfigurovať firewall servera a odfiltrovať tak prípadné útoky. Preto som vytvoril vlastný ochranný systém aplikácie, ktorý má v sebe implementované funkcie na ochranu pred známymi typmi útokov proti webovým aplikáciám. Tento systém som pomenoval jednoducho podľa jeho funkcie - PROTECTOR.

7.2 Ochranný systém Protector

Protector sa dá chápať ako nejaký ochranný obal, pretože každá požiadavka na ktorúkoľvek časť webu musí prejsť najprv jeho kontrolným mechanizmom. Ak požiadavka nevyhovie niektorej z podmienok, Protector vykoná potrebné kroky. Ak sa jedná len o malé porušenie bezpečnostných pravidiel, Protector zapíše túto udalosť do databázy a pri každom ďalšom prístupe kontroluje počet takýchto udalostí zapísaných za poslednú hodinu. Správca systému môže určiť pri akom počte narušení bezpečnosti a na aký dlhý čas odoprie Protector IP adrese prístup. Ak sa jedná o závažnejšie porušenie bezpečnostných pravidiel, Protector odoprie prístup okamžite. Výhodou tohto systému je, že pracuje nad všetkými časťami zákazníckej časti elektronického obchodu, teda odopretie prístupu je globálne. Ochranné vrstvy a činnosť Protectora sú zjednodušene znázornené na obrázku 12. a podrobnejšie sú opísané v nasledujúcich kapitolách.



Obr.12 : Ochranné vrstvy a činnosť systému „Protector“

7.2.1 Správa relácie

Pri prvom prístupe k webovej časti sa vytvára s klientským prehliadačom relácia a do cookie súboru na serveri sú uložené tri reťazce, ktoré potom Protector pri každej ďalšej požiadavke kontroluje, a vďaka ktorým je schopný odhaliť útok na reláciu – tzv. session hijacking.

Prvým reťazcom je unikátny identifikačný reťazec vytvorený spojením aktuálneho času v milisekundách, ktorý uplynul od 1.1.1970, tajného reťazca

známeho iba skriptu, ktorý s ním spolupracuje a náhodne vygenerovaného čísla. Celý tento reťazec je následne upravený hashovacím algoritmom MD5 a slúži na identifikáciu relácie v celom systéme elektronického obchodu. Ak príde požiadavka na ktorúkoľvek stránku, Protector najskôr kontroluje existenciu tohto reťazca a ak ho nenájde presmeruje požiadavku na prvú stránku webu.

Prvý reťazec však nijako nechráni reláciu pred tým, aby ju nemohol zneužiť útočník a manipulovať tak s dátami skutočného vlastníka. Aby som dokázal potrebu ďalších dvoch reťazcov používaných Protectorom je nutné uviesť príklad ako taký útok na reláciu môže vyzerať. Útočník pomocou analyzátoru protokolov (snifferu) odchytí pakety určené pre obeť. Z nich získa hodnotu identifikátora relácie, ktorú si v HTTP požiadavkách a odpovediach navzájom vymieňa prehliadač obete a server. Útočník zašle serveru požiadavku na nejaký chránený zdroj a pripojí k nej získanú hodnotu identifikátoru relácie. Server nevidí rozdiel medzi požiadavkou od obete a požiadavkou od útočníka, preto útočníkovi sprístupní požadované zdroje.

Systém Protector sa snaží takémuto prípadu zabrániť existenciou druhého kontrolného reťazca pozostávajúceho z tajného reťazca a časti HTTP hlavičky prvej požiadavky obsahujúcej meno klientského prehliadača. Pri každej požiadavke v danej relácii teda protector najskôr porovná zhodu s reťazcom, ktorý bol vytvorený pri prvej požiadavke a zhodu s reťazcom vytvoreným s údajmi z aktuálnej požiadavky. Ak sa tieto dva reťazce nezhodujú znamená to, že požiadavka prišla z iného prehliadača než s akým bola relácia vytvorená a to je neomylný znak toho, že sa niekto pokúša o útok na reláciu. Odpoveď Protectoru je jednoznačná a nekompromisná – IP adrese je okamžite odmietnutý prístup na časový interval určený správcom. Nie je to 100% ochrana proti útoku na reláciu, ale aspoň donúti neodbytného útočníka používať rovnaký prehliadač a operačný systém ako používa obeť. Toto je bežná technika používaná pri programovaní bezpečných webových aplikácií. V systéme Protector som ju však doplnil ešte o tretí reťazec.

Tretí overovací reťazec je využívaný rovnako ako predchádzajúci, no pozostáva z IP adresy klienta a tajného reťazca. Tým sa útočníkovi kladie ďalšia prekážka, pretože aby mohol využívať reláciu obete, musel by mať rovnakú IP adresu ako počítač obete. Pre mnohých to môže byť neprekonateľná prekážka, avšak môže nastať aj prípad kedy útočník aj obeť pristupujú k webu cez rovnaký HTTP proxy server, teda sú navonok reprezentovaný rovnakou IP adresou. V takomto prípade sa dá proti útokom na správu relácie brániť už len jedinou, no najúčinnejšou cestou – použitím šifrovaného spojenia s využitím SSL, čo zabráni útočníkovi získať z odchytených paketov hodnotu identifikátoru relácie.

7.2.2Autentifikácia

Overenie používateľa elektronického obchodu je vykonávané pomocou autentifikačnej metódy BASIC protokolu HTTP. Keď klient vyžiada prostriedky chránené týmto overovacím mechanizmom, pošle mu server odpoveď „HTTP 401 Unauthorized“. Klientský prehliadač po prijatí tejto odpovede automaticky zobrazí dialógové okno, do ktorého musí používateľ zadať svoje prihlasovacie meno a heslo. Následne prehliadač zašle opäť rovnakú požiadavku na chránený zdroj, no do hlavičky pripojí algoritmom BASE 64 šifrované prihlasovacie údaje, ktoré vyplnil používateľ. Tie sú po prijatí upravené hashovacím algoritmom MD5, ktorý v tomto prípade nahradzuje validáciu vstupu a sú porovnané s údajmi uloženými v databáze elektronického obchodu. Ak sa prihlasovacie meno alebo heslo nezhoduje s údajmi z databázy, Protector vytvorí v databáze zápis o udalosti a po správcom nastavenom počte neúspešných pokusov môže odoprieť IP adrese prístup.

Na overovaciu metódu HTTP: BASIC sú veľmi ľahko a s relatívne dobrými výsledkami aplikovateľné útoky hrubou silou. Už spomínaný Brutus AET2 však proti implementácii s využitím Protectora nefunguje, pretože nedokáže posielať v hlavičkách požiadaviek hodnotu identifikátoru relácie. Hneď pri prvom pokuse o zaslanie hesla Brutusom nenájde Protector identifikačný reťazec pre reláciu a presmeruje požiadavku na prvú stránku webu. Brutus teda dostane zo serveru odpoveď HTTP 200 OK a vyhlási prvé heslo za správne. Pokročilý útočník by sa však pri tomto nezdare určite nezastavil a pravdepodobne by sa pokúsil použiť nástroj, ktorý by zasielal v hlavičke aj hodnotu identifikátoru relácie. Aby som otestoval aj tento prípad, vytvoril som jednoduchú aplikáciu, ktorá umožňuje vykonávať útok hrubou silou proti systémom závislých na existencii relácie. Funkčne je podobná na Brutus, no z odpovede na prvú požiadavku dokáže získať hodnotu platného identifikátoru relácie a zasielať ju v hlavičke spolu s ďalšími požiadavkami, ktoré už predstavujú útok hrubou silou. Protector však úspešne odoláva aj takýmto pokusom a po nastavenom dovolenom počte neplatných pokusov odoprie IP adrese prístup. Kompletný priebeh útoku touto aplikáciou nazvanou PSEUDO-CRACKER popisuje Príloha D: Útok na overovací mechanizmus.

7.2.3Validácia vstupu

Vytvorená webová časť elektronického obchodu je odolná aj proti útokom na validáciu vstupu. Prvú ochrannú vrstvu tvoria funkcie vytvorené pomocou JavaScript-u, ktoré nedovolia používateľovi napísať do vstupných polí formulárov nebezpečné znaky. Týmito znakmi sa rozumejú znaky so špeciálnym významom pre skripty vykonávané na strane servera. Útok na validáciu vstupu jednoducho znázorňuje nasledujúci príklad

Predstavme si, že v PHP skripte sa nachádza podmienka

```
if ($_GET[“var”]=’retazec’)
```

a ako hodnotu GET premennej *var* by útočník zaslal reťazec

```
1=1) //
```

Ak nebola vykonaná validácia vstupu, podmienka by nadobudla úplne iný význam

```
if (1=1) //=’retazec’)
```

a teda by bola vždy pravdivou!

Vykonávanie funkcií JavaScript-u sa však dá veľmi ľahko v prehliadači vypnúť. Neposkytuje teda dostatočnú ochranu proti nebezpečnému vstupu, a preto musí byť validácia vykonávaná aj v PHP skriptoch, ktoré vykonáva server. Moja implementácia elektronického obchodu v PHP preto počíta s konfiguračnou direktívou PHP *„register_globals Off“* a s ošetrením každej prijatej hodnoty od používateľa funkciou *AddSlashes()*, ktorá pridáva pred nebezpečné znaky spätné lomítko, čo zaručuje, že znak nebude mať pre kód skriptu význam. Naviac systém Protector každý významnejší pokus o útok na validáciu zapisuje do databázy a následne podobne ako pri útokoch hrubou silou po dosiahnutí limitu stanoveného správcom odoprie IP adrese prístup.

7.2.4Prienik predsa možný

Aj napriek použitiu všetkých vymenovaných techník ochrany webovej časti elektronického obchodu ostáva priestor umožňujúci prienik. Neexistuje spôsob, ako by sa aplikačne dalo zabrániť útoku pomocou odchytávania paketov. Útočník môže získať hodnotu identifikátoru platnej relácie alebo odchytí šifrované prihlasovacie údaje zákazníka, ktorých dešifrovanie, ak vie ako na to, mu potrvá pár sekúnd. Použitie šifrovanej komunikácie medzi klientom a serverom pomocou SSL však vylučuje aj túto poslednú možnosť prieniku. Ak by útočník aj zachytil časť komunikácie, nedokáže ju dešifrovať. Šifrované spojenie však nenahrádza všetky obranné techniky systému Protector. Je len poslednou chýbajúcou časťou skladačky, ktorá významne posíli bariéru chrániacu webovú e-commerce aplikáciu.

7.3Správcovská časť elektronického obchodu

Kapitola bola z verzie web release vypustená.

8. ZÁVER

Hlavnou úlohou mojej práce bolo vytvoriť funkčný elektronický obchod. Na prvý pohľad sa to nezdalo nijako extrémne náročné, no čím viac hodín som trávil prácou nad týmto projektom, tým viac som si uvedomoval všetky požiadavky, ktoré musí moja aplikácia spĺňať. Elektronický obchod je aplikáciou, ktorá vytvára rozhranie narábajúce s veľmi citlivými údajmi. Či už sa jedná o samotné údaje o vykonaných transakciách alebo o osobné údaje zákazníkov, všetky je potrebné chrániť. Starostí o ochranu navrhnutého obchodu som sa mohol jednoducho zbaviť položením si otázky „Prečo by niekto napadol môj obchod?“. Správna otázka však znie „Ako by niekto napadol môj obchod?“. Vďaka neustálemu kladeniu si tejto otázky, experimentovaniu a štúdiu literatúry som vytvoril obranný systém Protector s implementovanými ochrannými mechanizmami proti mnohým známym typom útokov, ktoré by sa dali využiť proti vytvorenému elektronickému obchodu. Protector je ľahko prenositeľný, preto každá ďalšia webová aplikácia, ktorú vytvorím, ním určite bude chránená.

9. ZOZNAM BIBLIOGRAFICKÝCH ODKAZOV

[1] SCAMBRAY, J., SHEMA, M. Hacking bez tajemství: Webové aplikace. SILNÝ, P., a kol., Brno : Computer Press, 2003, 328 s., ISBN 80-7226-769-8

[2] KOSEK, J. PHP: Tvorba interaktivních internetových aplikací. Praha : GRADA Publishing, 1998, 492 s., ISBN 80-7169-373-1

[3] ŠKULTÉTY, R. JavaScript: Programujeme internetové aplikace. Praha : Computer Press, 2001, 208 s., ISBN 80-7226-457-5

[4] KOSEK, J. HTML: Tvorba dokonalých www stránek. Praha : GRADA Publishing, 1998, 296 s., ISBN 80-7169-608-0

[5] HLAVENKA, J., a kol., Výkladový slovník výpočetní techniky a komunikací. 3.vyd, Praha : Computer Press, 1997, 452 s., ISBN 80-7226-023-5

[6] PÍSEK, S. Delphi: Začínáme programovat. 2.vyd, Praha : GRADA Publishing, 2002, 328 s., ISBN 80-247-0547-8

[7] FIELDING, R., a kol., Hypertext Transfer Protocol -- HTTP/1.1. [online] RFC2616, 1999, http://www.w3.org/Protocols/rfc2616/rfc2616.txt

[8] FRANKS, J., a kol., HTTP Authentication: Basic and Digest Access Authentication. [online] RFC2617, 1999, ftp://ftp.isi.edu/in-notes/rfc2617.txt

PRÍLOHA A : DEKLARÁCIA ZÁSOBNÍKOV DÁT

Kapitola bola z verzie web release vypustená.

PRÍLOHA B : DEKLARÁCIA DÁTOVÝCH ELEMENTOV

Kapitola bola z verzie web release vypustená.

PRÍLOHA C : DEKLARÁCIA PROCESOV A DÁTOVÝCH TOKOV

Kapitola bola z verzie web release vypustená.

PRÍLOHA D : ÚTOK NA OVEROVACÍ MECHANIZMUS

Snaha otestovať obranný mechanizmus navrhnutého elektronického obchodu ma priviedla až k vytvoreniu aplikácie PSEUDO-CRACKER. Jej jedinou úlohou je použiť útok hrubou silou na uhádnutie hesla používateľa „user“ overovaného pomocou overovacieho mechanizmu HTTP: BASIC. Znie to jednoducho, no v kapitole 2.1.2 spomínaná aplikácia Brutus nebola schopná túto úlohu splniť, pretože nedokáže k HTTP požiadavkám pripojiť hodnotu identifikátora relácie – SID. Pri prvej požiadavke na chránenú stránku *„reg.php“* presmeruje Protector požiadavku na *„index.php“*, pretože nenájde v požiadavke platnú hodnotu identifikátoru relácie. Ako vidieť na obrázku 1, PSEUDO-CRACKER túto hodnotu odchytí a pripojí ju k nasledujúcej požiadavke na stránku *„reg.php“*.

Obr.1 : Pseudo-cracker získal identifikátor relácie

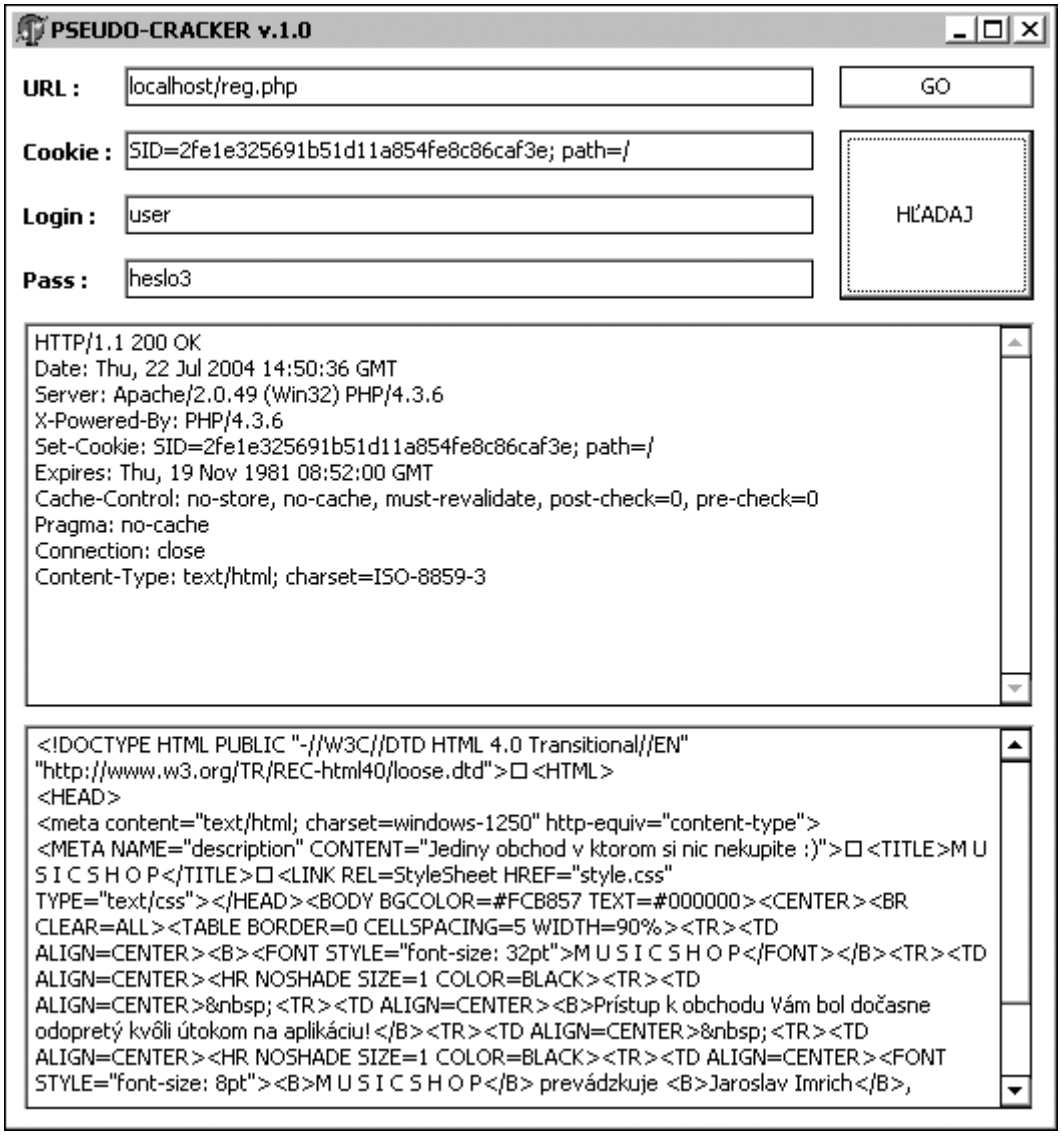
Z obrázku 2 je zrejmé, že druhú požiadavku Protector prepustil, pretože odpoveď serveru je *„HTTP/1.1 401 Authorization Required“*. PSEUDO-CRACKER ovláda platnú reláciu, môže teda pokračovať s útokom. Po kliknutí na tlačidlo *„HĽADAJ“* pripojí k požiadavke na stránku *„reg.php“* prihlasovacie údaje *user:heslo1*.

Obr.2 : Pseudo-cracker ovláda platnú reláciu

Obrázok 3 prezrádza, že prihlasovacie údaje *user:heslo1* neboli správne, pretože server opäť vrátil odpoveď *„HTTP/1.1 401 Authorization Required“*. Všetky aplikácie vykonávajúce útok hrubou silou pracujú tak, že zasielajú rôzne systematické kombinácie prihlasovacích mien a hesiel až kým nedostanú zo serveru odpoveď *„HTTP/1.1 200 OK“*. Vtedy sú posledné použité prihlasovacie údaje považované za správnu kombináciu a útok je ukončený. PSEUDO-CRACKER teda vykoná ďalší pokus a odošle požiadavku s prihlasovacími údajmi *user:heslo2*.

Obr.3 : Útok hrubou silou

Na túto požiadavku dostáva od serveru vytúženú odpoveď *„HTTP/1.1 200 OK“*, čo by malo znamenať, že prihlasovacie údaje *user:heslo2* boli správne. Po lepšom prehliadnutí tela odpovede, ktoré je zobrazené na obrázku 4 v spodnej časti okna, si však každý musí všimnúť vetu *„Prístup k obchodu Vám bol dočasne odopretý kvôli útokom na aplikáciu!“*. Znamená to len toľko, že každý neúspešný pokus o prihlásenie bol aplikačnou logikou elektronického obchodu zaznamenaný a po dosiahnutí dovoleného limitu (v tomto prípade 2) odoprel Protector prístup k skriptu *„reg.php“* a presmeroval požiadavku na stránku s týmto chybovým hlásením. Útočník môže v útoku pokračovať až po uplynutí času, ktorý taktiež určuje správca obchodu. Útok hrubou silou sa teda stáva vďaka použitým algoritmom v reálnom čase neuskutočniteľným.



Obr.4 : Protector odoprel prístup

PRÍLOHA E : INŠTALÁCIA

Kapitola bola z verzie web release vypustená.