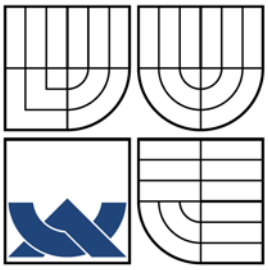
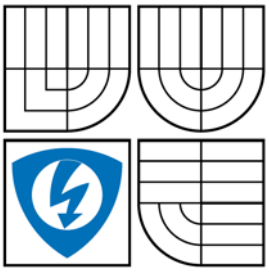


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

BEZPEČNÝ PŘÍSTUP DO WEBOVÉHO ROZHRANÍ

SECURE ACCESS TO WEB INTERFACE

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. MILAN KAZIK
AUTHOR

VEDOUCÍ PRÁCE — ING. ONDŘEJ RÁŠO
SUPERVISOR

BRNO 2009

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

1 ÚVOD

Web hacking je jednou z najelegantnejších techník, akú hacking pozná. Hlavným dôvodom na toto tvrdenie je to, že útočník k úspešnému napadnutiu systému vo väčšine prípadov nepotrebuje nič viac ako webový prehliadač a samozrejme odborné znalosti webových technológii, hlavne webového serveru. Bezpečnosť webových aplikácií stojí a padá na jeho bezpečnosti. Akokoľvek zabezpečená aplikácia je zraniteľná, pokiaľ je zraniteľný server, na ktorom beží. Rovnaké tvrdenie platí i o serveroch databázových.

V porovnaní s programátorom webovej aplikácie je útočník oveľa nápaditejší. Je nebezpečné myslieť si, že je niečo neprelomiteľné len preto, že ma ako programátora nenapadá ako to prelomiť. Uľahčovanie si práce na úkor kvality a lenivosť sú hlavnou príčinou, prečo je také veľké množstvo webových aplikácii na webe bezpečnostne zraniteľných. Asi 90 % webových útokov je pritom spôsobených nesprávnym ošetrením vstupov [2].

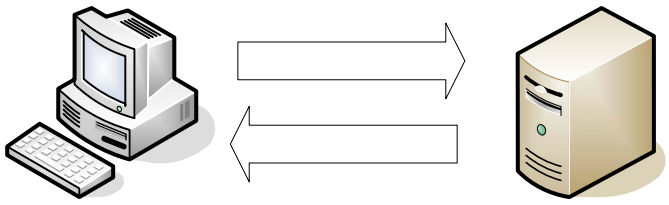
V tejto diplomovej práci sú popísané najčastejšie útoky na webové rozhranie. Jej cieľom je navrhnúť a realizovať rozhranie také, aby bolo voči týmto útokom imúnne.

V prvej kapitole je popísaný základ protokolu HTTP, webového a databázového serveru. Ich slabiny a silné stránky vzhľadom k bezpečnosti. Ďalšia kapitola sa zaoberá vstupmi a výstupmi z webovej aplikácie, ktoré sú vstupnými bodmi pre potenciálne útoky. Neskôr sú tieto útoky podrobne rozobrané a je tiež navrhnutá cesta k ochrane. Kapitola o zaznamenávaní udalostí sa zaoberá informovaním užívateľa i webmastera o potenciálne nebezpečných situáciách, ktoré môžu nastať. V praktickom riešení je podrobne rozobraná štruktúra a fungovanie projektu. V poslednej kapitole je reč o certifikátoch a zabezpečení pomocou nich.

2 ZÁKLAD WEBOVÝCH APLIKÁCIÍ

2.1 HYPERTEXT TRANSFER PROTOCOL (HTTP)

Za veľkým rozmachom internetu stojí protokol HTTP. Štandard je popísaný v RFC1945. HTTP je bezstavový protokol typu výzva-odpoveď. Komunikácia prebieha medzi prehliadačom a webovým serverom. Iniciátorom spojenia je vždy prehliadač. HTTP je teda protokolom typu klient-server (viz Obr. 1 [2]).



Obr. 1 Model klient-server používaný na webe

Prvá oficiálna špecifikácia protokolu HTTP sa označuje HTTP/0.9. V súčasnosti najpoužívanejšia verzia je 1.0. Obsahuje metódy ako GET, HEAD a POST. V roku 2001 bola vyvinutá aj verzia 1.1. Tá je obohatená o metódy CONNECT, DELETE či OPTIONS. Tie ale z hľadiska bezpečnosti nehrajú príliš veľkú úlohu.

HTTP požiadavka

Typická HTTP požiadavka má tvar [2]:

GET / HTTP / 1.0

Host: www.example.com

*Accept: text/html, text/plain, image/**

Accept-Language: en

User-Agent: Mozilla/5.0

Prvý riadok sa nazýva Request-Line (riadok s požiadavkou). Z neho je možné zistiť, o akú metódu ide. Pri požiadavku typu GET sú parametre zakódované ako

časť URL. Pri požiadavku typu POST sú dáta posielané ako súčasť hlavičky. Nie sú teda viditeľné v adrese URL.

HTTP Odpoveď

Je odpoveď serveru na HTTP požiadavku. Je možné ju rozdeliť na tri časti:

- kód odpovedi (200 OK, 301 Moved Permanently, 404 Not Found,…)
- pole hlavičky
- dáta

V tabuľke 1 je uvedený zoznam hlavičiek protokolu HTTP/1.0. HTTP/1.1 obsahuje ešte zopár ďalších, ale tie nie sú pre oblasť bezpečnosti nejak zvlášť dôležité.

Tab. 1: Zoznam hlavičiek protokolu HTTP/1.0

Pole hlavičky	**Popis**
Allow	Obsahuje metódy podporované požadovanou položkou
Authorization	Obsahuje autorizačné informácie pre HTTP autentizáciu
Content-Encoding	spôsob kódovania dát
Content-Length	dĺžka dát v bytoch (v desiatkovej sústave)
Content-Type	typ zasielaných dát;dôležité pre správne zobrazovanie na strane klienta
Date	aktuálny dátum a čas na serveri
Expires	časový rozsah platnosti dát
From	emailová adresa osoby zodpovednej za dáta (zriedkavo používané)
Last-Modified	Datum a čas poslednej úpravy
Location	umiestnenie požadovanej položky
Pragma	volitelné správanie na požiadavky (napr. no-cache)
Referer	umožňuje klientovi určiť adresu predchádzajúcej stránky
Server	software bežiaci na serveri (napr. Server: Microsoft-IIS/5.0)
User-Agent	doplnkové informácie o klientovi (napr. User-Agent: Mozilla/5.0)
WWW-Authenticate	odpoveď na kód 401 Unauthorized - výzva k dohodnutiu sa na autorizácii

Z hľadiska bezpečnosti sú dôležité sú najmä hlavičky Referer, Server a User-Agent.

Hlavička Referer – odosieľateľ

Táto hlavička obsahuje URL dokumentu, z ktorého požiadavka pochádza. Typickým príkladom použitia je kliknutie na odkaz.

Po spustení tejto stránky sa prehliadač pripojí na server, aby stiahol požadovaný dokument. V požiadavke na konkrétny PHP skript (v tomto prípade registration.php) odošle prehliadač hlavičku Referer, ktorá vyzerá tak ako na obrázku 2.

```
GET /registration.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/x-shockwave-flash, application/vnd.ms-excel,
application/vnd.ms-powerpoint, application/msword, */*
Referer: http://127.0.0.1/index.php
Accept-Language: sk
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
1.1.4322)
Host: 127.0.0.1
Connection: Keep-Alive
Cookie: WMdata=63659150-63659150-1238057179wm:0wm?0;
PHPSESSID=0viqm6im0defpscjv9vq7tnl55
```

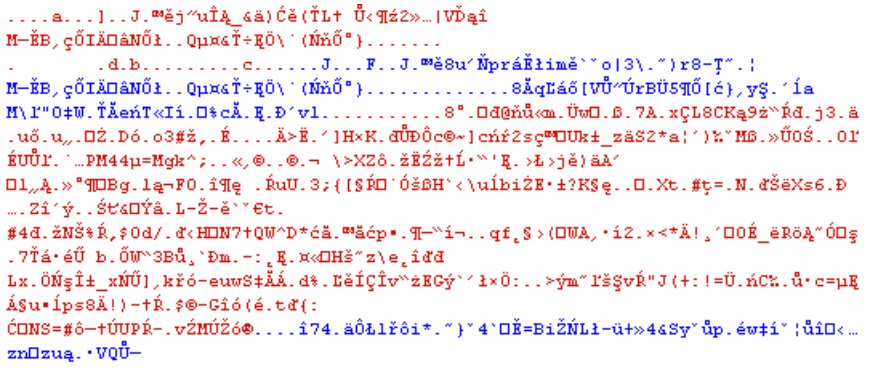
Obr. 2: Hlavička odosielaná prehliadačom po kliknutí na odkaz

Bezpečnostný problém spočíva v tom, že mnoho serverov kontroluje práve hlavičku Referer, aby sa uistili, že požiadavka prichádza práve z danej stránky. Kedže tieto hlavičky pochádzajú zo strany klienta, môže ich tento kedykoľvek upraviť a server tak oklamať (viz kapitola 3.1)

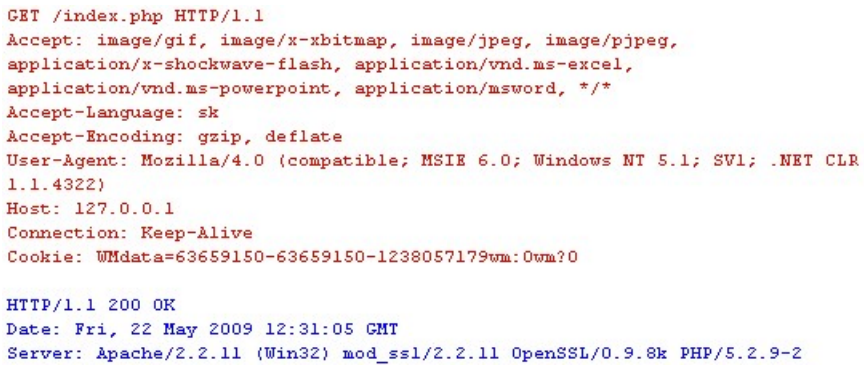
2.1.1 Protokol HTTPS

Protokol HTTPS je vlastne klasický protokol HTTP použitý cez šifrovací kanál. Ten vytvára protokol Secure Socket Layer (SSL) alebo jeho nástupcu Transport Layer Security (TLS). Je dôležité vedieť, že šifrovanie chráni len komunikáciu medzi klientom a serverom. Informácie o tom, že ak bude namiesto protokolu HTTP použitý HTTPS, je o bezpečnosť aplikácie postarané, sa nezakladajú na pravde. Útočník môže stále napadnúť klienta i server, ale bude

preňho ťažké zaútočiť na komunikačný kanál medzi nimi. Faktom je, že sa týmto spôsobom vyhneme útoku Man In The Middle (MITM) (viz kapitola 4.3). Dáta prenášané nebezpečnými sieťami budú nečitateľné pre kohokoľvek, kto sa k dátam dostane. Je tým teda vyriešená iba časť bezpečnostnej otázky. Rozdiel medzi protokolmi HTTP a HTTPS je jednoznačne vidieť z odchytených paketov programom CommView[12].



Obr. 3: Dáta protokolu HTTPS získané pomocou programu CommView



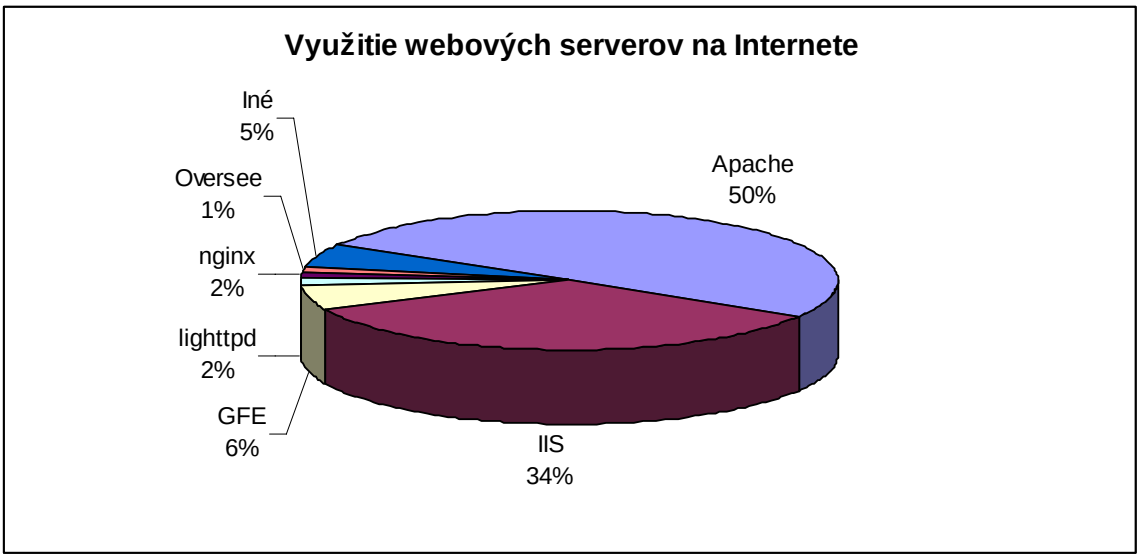
Obr. 4: Dáta protokolu HTTP získané pomocou programu CommView

2.2 WEBOVÉ A DATABÁZOVÉ SERVERY

Webový server je počítač, ktorý je zodpovedný za vybavovanie HTTP požiadavkov od klientov, napríklad odoslanie webovej stránky.

Spoločnosť Netcraft každý mesiac zverejňuje štatistiku o využívaní webových serverov [3]. Na prvých dvoch miestach sa dlhodobo umiestňujú Apache od rovnomenného výrobcu a Internet Information Services (IIS) od Microsoftu.

Spolu tvoria približne 84% z celkového počtu webových serverov, ktoré sa na internete používajú, viz obr. 5



Obr. 5: Využitie webových serverov vo svete za obdobie október 2008

2.2.1 Apache

Najpoužívanejším webovým serverom v prostredí Internetu je Apache. Jeho vlastnosti sú s bezpečnosťou úzko previazané. Server Apache umožňuje použitie:

- virtuálnych hostiteľov
- Server Side Includes
- Dynamického obsahu s CGI
- obslúh
- premenného prostredia

Na serveri Apache bola vyvinutá aj táto diplomová práca. Aktuálna verzia systému bola v čase tvorby 2.2.11. Boli na ňu tiež aplikované najnovšie bezpečnostné záplaty. Podrobná analýza dôležitých nastavení serveru je rozobraná v kap. 2.3

2.2.2 IIS

Najväčšou bezpečnostnou slabinou serveru IIS je aplikácia Internet Server Application Programming Interface (ISAPI). Vďaka ISAPI možno rozširovať

možnosti webového serveru vlastnými programami. Tým vytvára množstvo vstupných bodov, a tým zvyšuje pravdepodobnosť vniknutia do systému. Riešením problému je buď spoľahnúť sa na bezpečnostné záplaty Microsoftu alebo odstrániť nepotrebné rozšírenia zo systému. Druhé riešenie je pravdepodobne rozumnejšie.

2.2.3 Iné bezpečnostné slabiny webových serverov

Štandardné uložené procedúry sú vytvorené na serveri z dôvodu uľahčovania práce programátorom. Zlepšujú tiež výkon celého systému. Môžu však byť prostriedkom, ktorý môže spôsobiť vniknutie útočníka na server. Preto je v záujme bezpečnosti, tieto po inštalácii odstrániť.

Napríklad uložená procedúra xp_dirtree umožňuje získať adresárový strom.

2.2.4 Databázové servery

Základom každej webovej aplikácie je databáza, nástroj pre ukladanie a získavanie dát, využiteľných užívateľmi. Pri útoku na databázový server ide útočníkovi o získanie dát, prípadne vyvolanie chýb, aby sa dozvedel podrobné informácie o nastavení. Niektoré databázy sa rovnako ako webové servery potýkajú s bezpečnostným problémom štandardných uložených procedúr. Naštastie sa tieto procedúry dajú jednoducho odstrániť.

Projekt diplomovej práce má uloženú databázu v MySQL databáze verzie 5.1.33. Hlavná databáza má názov „diplomovapraca“. Na prístup k nej je definovaný jedinečný užívateľ, ktorého práva sú len také, aké sú nevyhnutne potrebné k správnemu fungovaniu systému.

	Používateľ	Hostiteľ	Heslo	Globálne práva [1]	Prideliť	
☐	Akýkoľvek	%	--	USAGE	Nie	
☐	diplomovapraca	127.0.0.1	Áno	SELECT, INSERT, UPDATE, CREATE, ALTER	Nie	

Obr. 6: Užívateľ „diplomovapraca“ a jeho práva

Takto definovať práva však nie je možné pokiaľ máme webovú aplikáciu umiestnenú na komerčnom serveri hostingovej spoločnosti. Väčšina týchto serverov je ale nastavená tak, aby k žiadnej bezpečnostnej trhline nedošlo.

V databáze diplomovej práce sa nachádzajú tri tabuľky:

- temp_uzivatelia
- transakcie
- uzivatelia

Tabuľka temp_užívatelia slúži k dočasnému ukladaniu užívateľov pred ich aktivovaním účtu (viz kapitola 6). Do tabuľky transakcie sa ukladá jedinečný identifikátor, pomocou ktorého je zabezpečená ochrana proti viacnásobnému spracovaniu údajov z formuláru (viz kapitola 3.1.3). A konečne najdôležitejšou tabuľkou je tabuľka uzivatelia. Tam sú uložené všetky dôležité polia: meno, heslo v šifrovanej podobe, emailová adresa, identifikátor relácie session, IP adresa, lasttime(posledný čas prihlásenia), lastloginfail (čas posledného neúspešného prihlásenia) a počet zostávajúcich pokusov o prihlásenie. Podrobný popis významu a funkcie jednotlivých polí nájdete v kapitole 6.

2.3 DÔLEŽITÉ NASTAVENIA WEB SERVERU APACHE

Všetky pre bezpečnosť dôležité nastavenia serveru Apache boli v systéme nastavené už ako predvolené, takže nebolo nutné upravovať ich. Nižšie sú popísané najdôležitejšie z nich.

2.3.1 Vylistovanie adresára

Funkcia sa zapína príkazom *Options +Indexes*. Ak je táto funkcia na serveri zapnutá, útočníkovi umožní zobrazenie kompletnej štruktúry adresára. Ukážka obsahu adresára /obr, kde sú umiestnené obrázky vyzerá ako na obrázku 7. Pri vypnutej možnosti sa zobrazí chybová stránka 403 Forbidden – nepovolený prístup.

Obr. 7: Vylistovanie adresára

2.3.2 Register globals

Ide o nastavenie PHP, ktoré určuje ako má PHP interpret spracovávať vstupné premenné. Predvolené nastavenie od verzie PHP 4.2 je vypnuté (OFF) [13]. Od verzie 6 dokonca nebude táto funkcia vôbec obsiahnutá.

Zapnuté (ON) - PHP nekontroluje pôvod premennej, t.j. nezáleží či premenná pochádza z formuláru (metódou POST či GET) alebo ide o lokálnu premennú.

Vypnuté (OFF) – bezpečnejšie riešenie

Pokiaľ je skript napísaný pre správnu funkciu s register_globals = OFF, bude fungovať aj direktívou zapnutou (ON). Naopak to už však neplatí.

Moja webová aplikácia je napísaná tak, aby vyhovovala register_globals nastavenou na OFF.

Názorný príklad - pole formulára ‘email‘, ktoré posielame metódou POST:

<form action=forgot_pass.php method=post name=resetPass>

<input type=text name=email>

Pokiaľ je direktíva zapnutá, pole email prijmeme ako $email. Ak je ale vypnutá, musíme použiť príkaz: *$_POST['email']*.

2.3.3 Magic_quotes_gpc

Magic_quotes_gpc (GPC = GET/POST/COOKIE) je veľmi dôležitá konfiguračná direktíva, ktorá je dôležitá pre upravenie SQL príkazov predtým, ako budú uložené do databázy. Ak je zapnutá, znaky:

Jednoduché úvodzovky (‘), dvojité úvodzovky (“) a spätné lomítko (\) sú „escapované“ (doplnené na začiatku o 1 spätné lomítko). Reťazec ‘meno’ bude nahradený reťazcom \'meno\'. Rovnaký výsledok zaručí použitie PHP funkcie addslashes()

Výhoda tejto direktívy je tá, že ak programátor na nejaký neošetrený reťazec zabudne, nápravu vykoná sama. Osobne ale preferujem mať túto direktívu vypnutú. Pokiaľ je direktíva zapnutá a na reťazec je aplikovaná funkcia addslashes() budú pred každý escapovaný znak použité hneď tri spätné lomítka (jeden pridá funkcia addslashes() a dva direktíva magic_quotes_gpc). Potom je nutné použiť opačnú funkciu stripslashes(), ktorá nadbytočné lomítka odstráni.[14]

Aby bola webová aplikácia dostatočne univerzálna, je do nej zakomponovaná funkcia **get_magic_quotes_gpc()**, ktorá kontroluje, či je táto direktíva zapnutá alebo nie. V prípade vypnutej direktívy vracia 0, inak vracia 1.[13]

V diplomovej práci som pracoval s vypnutou direktívou. Ošetrenie nepovolených znakov som riešil PHP funkciou **mysql_real_escape_string()**. (viz kapitola 4.4)

3 OŠETRENIE VSTUPU

Aby bolo možné ochrániť webovú aplikáciu pred útočníkmi, je nutné pochopiť ich myslenie. Len vtedy je možné úspešne sa ubrániť. Na druhú stranu to, že programátora nenapadá žiadny spôsob, ako by mohla byť aplikácia napadnuteľná ešte neznamená, že inteligentný útočník do nej neprenikne. Účely útokov môžu byť rôzne. Od upravenia vzhľadu stránok až po získanie dôležitých a citlivých informácii (napr. užívateľské heslá, čísla kreditných kariet a pod.). Pre programátora je teda nevyhnutné ošetriť všetky cesty, ktoré do aplikácie vedú.

3.1 VSTUP UŽÍVATEĽA

Vstup užívateľa je miesto, odkiaľ prichádzajú do systému dáta. Vstupom môže byť adresa URL vrátane parametrov, formulárové prvky, atď. Bola by chyba, ak by programátor uvažoval tak, že užívateľ bude napríklad do formulárových polí zadávať vždy platné údaje. Ak užívateľ do poľa rok narodenia zadá písmeno a formulár odošle, je na svete neplatný vstup.

Čo je možné považovať za neplatný vstup? Neplatný vstup je taký, ktorý obsahuje nepovolené znaky alebo kombinácie znakov, ktoré webový server identifikuje ako znaky riadiace. To znamená, že užívateľ môže zadaním vhodného reťazca ovládať správanie serveru. Môže napríklad prehliadať konfiguračné súbory, spúšťať na servery rôzne skripty alebo získavať či meniť tabuľky v databázi. Takáto činnosť je samozrejme neprípustná, a preto je dôležité jej za každú cenu zabrániť. Rozlišujeme dva druhy vstupov, ktoré je potrebné ošetriť.

Sú to[1]:

- Vstup tvorený na strane klienta
- Vstup tvorený na strane serveru

3.1.1 Vstup tvorený na strane klienta

Pochádza zo vstupných textových polí typu text, password alebo textarea. Užívateľom vytvorený vstup nemusí byť vždy platný. A to kvôli chybám, ktoré sa v ňom vyskytnú. Viac o ošetrení klientského vstupu nájdete v kapitole 3.1.4

3.1.2 Vstup tvorený na strane serveru

Ostatné vstupy formulárov – dáta zo skrytých polí, položky zoznamu, súbory cookie, hlavičky HTTP atď.

Rozdiel oproti predchádzajúcemu typu spočíva v tom, že tento vstup za normálnych okolností neplatný nikdy nebude. Ak sa stane, že je neplatný, existuje dôvodné podozrenie, že niekto manipuluje s hodnotami, ktoré sú za normálnych okolností mimo jeho dosah. V prípade, že sa tak stane, je vhodné túto udalosť zaznamenať (viz kapitola 5).

3.1.3 Formulár a prihlasovanie užívateľa

Existujú nasledujúce pravidlá pre formuláre, ktorých je potrebné sa držať[11]:

1. Formulár spracovať pred akýmkoľvek výstupom
2. V prípade úspešného odoslania formuláru (napr. uloženie do databáze) zabezpečiť, aby sa pri obnovení stránky nevykonalo uloženie znovu.
3. Hodnoty do databáze vkladať ošetrené
4. Pri spracovávaní formuláru vykonávať kontrolu aj na strane klienta (JavaScript) – tzv. user friendly aplikácia
5. Pri vkladaní hodnoty z formuláru upraviť túto hodnotu (odstrániť HTML príkazy, upraviť špeciálne znaky, atď)

Viacnásobné odosielanie údajov z formuláru

Problém posielania tzv. postdata informácii prehliadačom môže v neošetrenej aplikácii narobiť množstvo problémov. Takáto situácia nastáva vtedy, ak dáta formulára zasiela i spracováva jeden skript. Prehliadač pri obnovení stránky zašle rovnaké dáta opäť a ak nie je aplikácia na podobné správanie pripravená, vykoná sa činnosť ako keby boli dáta štandardným spôsobom poslané vo formulári.

Na riešenie problému sa najčastejšie používa presmerovanie alebo použitie náhodných reťazcov, pomocou ktorých je každé odoslanie dát jednoznačne identifikované. Pristúpil som k riešeniu druhým spôsobom.

Na odoslanie identifikátora (trans_id) je použité skryté pole formuláru (hidden)

$dataid = mt_rand();

<input type=hidden name=trans_id value=$dataid>

Na druhej strane je použité ukladanie náhodných reťazcov do tabuľky v databaze (transakcie). Na začiatku skriptov je kontrolované, či sa daný identifikátor v tabuľke nachádza. Ak nie, skript pokračuje vo svojej činnosti a reťazec do tabuľky použitých identifikátorov pridá.

"INSERT INTO transakcie(trans_id) VALUES ('$trans_id')"

Ak sa už reťazec v tabuľke nachádza, rozhodovacia podmienka na začiatku skriptu nebude splnená a skript sa predčasne ukončí. Týmto je znemožnené, aby boli dáta z formuláru použité znovu. Problém tohto riešenia spočíva v rýchlom narastaní počtu identifikátorov trans_id v databázi. Tento problém sa dá riešiť pravidelným odstraňovaním použitých reťazcov (napr. použitím cronu).

Prihlasovanie užívateľa

Pri prihlasovaní je dôležité vybrať vhodný typ autorizácie a teda spôsob predávania údajov od klienta k serveru. Pri kombinácii Apache, PHP a MySQL je na výber niekoľko možností:

- HTTP autentifikácia

  Nebezpečná, pretože heslá sa predávajú pri každej požiadavke v hlavičke v čistom texte. Výhodou je jednoduchosť použitia. Prehliadač posúva po prihlásení užívateľské meno a heslo automaticky bez ďalších zásahov.

- Session autentifikácia

  Nepriame predávanie s pomocou session. Nie je nutné neustále posielať meno a heslo, len jednoznačne určitý reťazec. Nevýhodou je zložitejšie programovanie. V diplomovej práci je použitá práve session autentifikácia. Ostatné doplnkové kontroly sú vysvetlené nižšie.

Spôsob prihlasovania pri session autentifikácii

Po úspešnom prihlásení užívateľa je vygenerovaný náhodný reťazec (session_login_string), ktorý sa uloží do databázi pri položke daného užívateľa a zároveň sa zapíše aj do session premennej. Do databázi bude tiež dobré uložiť čas posledného prístupu, aby bolo možné po určitom čase nečinnosti užívateľa odhlásiť, a tiež IP adresu, z ktorej prihlásenie prebehlo, kvôli zvýšeniu bezpečnosti. Na začiatku skriptov stránok, kde je nutné, aby bol užívateľ prihlásený sa potom session identifikátor kontroluje a na základe toho je užívateľovi umožnené na stránku pristúpiť.

Príkladom v diplomovej práci je stránka určená na zmenu hesla change_pass.php. Kontrola session potom vyzerá takto:

if(isset($_SESSION["session_login_string"]))

Ak je podmienka splnená (session je nastavené), skript pokračuje na stránku. Ak podmienka splnená nie je, zobrazí sa chybové hlásenie ako na obr. 8.

Obr. 8: Chybové hlásenie o neprihlásenom užívateľovi

3.1.4 Ošetrenie vstupého poľa formuláru

Každé pole formuláru je nevyhnutné na strane programovacieho jazyka ošetriť tak, aby sme útočníkovi znemožnili využiť to. Klientské overovanie nie je

vhodné z hľadiska toho , že sa plne nachádza v moci klienta a môže teda byť jednoducho obídené.

Z hľadiska bezpečnosti dôležité kontroly sú najmä:

- definovať povolené znaky pre prihlasovacie meno
- definovať validitu emailovej adresy

Pre potrebu malej aplikácie postačuje, aby boli v užívateľskom mene povolené len veľké a malé znaky abecedy, číslice a znak podtržítka.

Na kontrolu je vhodná PHP funkcia *ereg()*, ktorá je určená na kontrolu regulárnych výrazov. Pomocou regulárnych výrazov je možné vytvoriť masku a pomocou nej potom analyzovať ľubovoľný reťazec[13].

Konkrétny príklad kontroly užívateľského mena:

EregI("^[_a-z0-9]+$", $meno)

Použité je kľúčové slovo eregi (rovnaká funkcia ako ereg() akurát bez rozlišovania veľkých a malých písmen). Znak „^“ značí začiatok regulárneho výrazu. V lomených zátvorkách je definovaná množina možných znakov. Znak „+“ definuje neobmedzený počet opakovaní týchto znakov a konečne znak „$“ regulárny výraz ukončuje. Regulárny výraz je porovnávaný s užívateľským meno uloženým v premennej $meno. Pokiaľ užívateľské meno vyhovuje, funkcia eregi() vracia pravdu (TRUE).

Kontrola emailovej adresy má rovnaký princíp ako predchádzajúca kontrola užívateľského mena. Je však podstatne zložitejšia. Premenná $atom definuje povolené znaky pred znakom „@“, premenná $domain za ním. V premennej $regex je uložený výsledný regulárny výraz, ktorý sa potom porovnáva s emailovou adresou z formulára ($email).[22]

$atom = '[-a-z0-9!#$%&+/=?^_`{|}~]';*

$domain = '([a-z]([-a-z0-9][a-z0-9]+)?)';*

$regex = '^' . $atom . '+(.' . $atom . '+)' . '@' . '(' . $domain . '{1,63}.)+' . $domain . '{2,63}$';*

eregi($regex, $email)

Ďalšie formulárové polia v tejto diplomovej práci, ktorých ošetrenie už bolo viac či menej jednoduché boli:

- kontrola naplnenia poľa
- kontrola zhody hesiel (pri registrácii)
- obmedzenie dĺžky poľa

3.1.5 Slepá dôvera v serverový vstup

Táto časť sa týka najmä skrytých polí angl. hidden. Akonáhle sú určité dáta posielané klientovi, stráca server kontrolu nad ich validitou, pretože na druhej strane môže dôjsť k ich úprave. Problém sa netýka iba skrytých polí, ale tiež klientských skriptov ako je napr. JavaScript. Použitie JavaScriptu užívateľovi pomôže vyplniť formulár dátami, ktoré sú validné. V žiadnom prípade ale nemôže slúžiť ako jediná kontrola validity. V prehliadači je možné javascript vypnúť a tým je celá ochrana obídená. Pretože ide o vstup zo strany klienta (viz kapitola 3.1.1) nie je tento vstup dôveryhodný.

3.2 HESLÁ

Bezpečnosť webových aplikácií stojí a padá na použitých heslách. Pomocou nich si užívatelia chránia osobné a niekedy aj veľmi citlivé údaje na serveroch. Príliš krátke heslá tvorené malým množstvom znakov sú ľahko odhaliteľné najjednoduchším spôsobom odhaľovania hesiel, tzv. hrubou silou. Lámanie hesiel hrubou silou je technika, pri ktorej sú skúmané všetky permutácie znakov. Ďalším spôsobom ako rýchlo odhaliť heslo je tzv. slovníkový útok. Pri ňom sa postupne skúšajú slová zo slovníka a ich príbuzné tvary. Na to, aby bolo heslo bezpečné, je potrebné aby spĺňalo nasledovné kritéria [8]:

- Obsahovalo aspoň 8 znakov
- Neobsahovalo úplné slová

- Neobsahovalo užívateľské meno, skutočné meno ani meno spoločnosti
- Obsahovalo znaky zo všetkých 4 skupín znakov (veľké písmená, malé písmená, číslice a špeciálne znaky)

Pre uľahčenie voľby hesla užívateľovi podľa práve posledného bodu vyššie uvedeného zoznamu, je na stránke registrácie naprogramovaná nápoveda v podobe JavaScriptu.

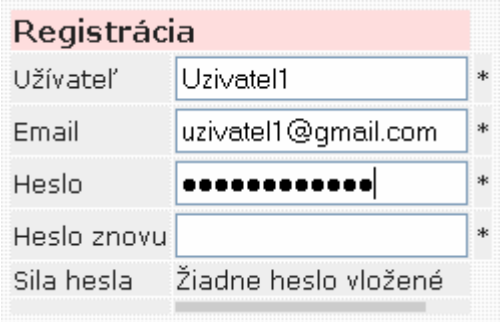
Jednoduchý skript kontroluje silu užívateľovho hesla podľa jednoduchého algoritmu. Dôležitá je „konečná cena“ hesla, ktorá sa určí podľa toho, koľko bodov dané heslo spĺňa. Definovaných je 5 kategórií:

- Heslo je dlhšie ako 6 znakov
- Heslo obsahuje znaky z malej (a-z) aj veľkej abecedy (A-Z)
- Heslo obsahuje minimálne jednu číslicu (0-9)
- Heslo obsahuje minimálne jeden špeciálny znak (!,@,#,$,%,^,&,*,?,_,~,-,atď)
- Heslo je dlhšie ako 9 znakov

Obr. 9: Ukážka funkcie skriptu - použité heslo: Qasdf59*wq

Ak dané heslo spĺňa všetky podmienky, je skriptom označené ako „veľmi silné“. Dôvod umiestnenia tejto nápovedy na moju stránku je jediný. Upozorňovať užívateľov, aby používali bezpečné heslá. Vyhnú sa tým nebezpečenstvu zlomenia hesla hrubou silou či slovníkovým útokom.

Ak je JavaScript v prehliadači vypnutý, z hľadiska bezpečnosti aplikácie sa nič nedeje, pretože validita formulárových polí je ošetrená na úrovni jazyka PHP.



Obr. 10: Vypnutý JavaScript v prehliadači

3.2.1 Bezpečná správa hesla

Keďže si nikdy nemôžeme byť istý, či sa niekto nedostane k tabuľke v databázi, kde sú heslá uložené, je rozumné tieto ukladať v šifrovanej podobe. A práve na túto činnosť sa používa tzv. hashovanie hesiel. Ide o techniku, pri ktorej sa z hesla v otvorenom texte, získa reťazec náhodných znakov. Hashovanie je jednocestná funkcia. Logicky teda z hashu nie je možné spätne odvodiť heslo. Na hashovanie hesiel sa používa niekoľko algoritmov, z ktorých je veľmi rozšírený Secure Hash Algorithm (SHA)[9].

SHA1 používa dĺžku hashu 160 bitov (v php funkcii sha1() je to 40 bitov). Bol vyvinutý americkou bezpečnostnou agentúrou (NSA) ako nástupca funkcie Message-Digest algorithm 5 (MD5), v ktorej boli odhalené bezpečnostné trhliny. MySQL funkcia bola pridaná v MySQL 4.0.2.

"Solenie" hesiel

Ak dvaja užívatelia zadajú rovnaké heslo, majú logicky aj rovnaký haš. Útočník, ktorý získa tabuľku databázy s uloženými heslami potom môže porovnávať haše a tak heslo odhadnúť. Znižuje sa tým bezpečnosť uložených hesiel. Prakticky sa to rieši tak, že hashované nie je len heslo, ale jeho kombinácia s reťazcom, ktorý je náhodne vygenerovaný pri prvom ukladaní do databázy. Tejto činnosti sa hovorí "solenie" hesiel z angl. salt.

Na generovanie hesiel som v diplomovej práci použil nasledovné:

$token = "g8KLm4";

```
$nove_heslo = sha1(sha1($meno.$heslo.$token));
$nove_heslo = substr($nove_heslo,0,32);
```

Dopredu som si určil náhodný token. Do hesla je pri každom hašovaní zakomponované užívateľské meno spojené s heslo z formulára a tokenom. Tento reťazec je následne dvakrát zhašovaný funkciou sha1() a následne kvôli zmäteniu nepriateľa je tento hash skrátený na 32 znakov. Táto dĺžka nebola zvolená náhodne. Rovnako dlhý hash je totiž výstupom PHP funkcie md5(). Keďže skrátenie hesla nebolo príliš radikálne, nemusíme sa báť, že by došlo ku kolízii hashov, t.j. dve heslá by mali rovnaký hash.

4 ÚTOKY A OBRANA

Aby bolo možné pochopiť hrozby a dôsledky webového útoku, je potrebné pochopiť špecifikáciu HTTP protokolu (viz kapitola 2.1). Nestačí použitie firewallu pretože webové útoky sú vedené cez TCP porty 80 a 443 a tie firewall prepustiť musí. Inak by totiž nebola žiadna komunikácia možná.

4.1 RELÁCIA UŽÍVATEĽA

Pri práci s reláciou (session) si klient a server neustále medzi sebou vymieňajú session identificator (SID). Ide o náhodne vygenerovaný token, podľa ktorého je server schopný určiť, či požiadavka skutočne prišla od konkrétneho návštevníka. Ten kto pozná SID, má prístup k celej relácii.

Základné pravidlá pre tvorbu SID[7]:

- unikátnosť – pri každej návšteve je potrebné vygenerovať nový identifikátor
- náhodnosť – SID musí byť náhodné a nezávislé (napríklad na IP adrese prípadne aktuálnom čase)
- neodvoditeľnosť – rad po sebe vygenerovaných SID nesmie ukazovať žiadnu pravidelnosť
- dostatočná dĺžka – zabráni sa tak útoku hrubou silou
- expiračnosť – po určitom čase SID obnoviť

Generovanie identifikátoru:

$session_login_string = md5(uniqid(mt_rand()) . $ip);

Z náhodne vygenerovaného čísla je generovaný reťazec, ktorý je spojený s IP adresou užívateľa a následne je celý zhašovaný funkciou md5(). Dĺžka identifikátora je potom 32 znakov.

Je dôležité, aby bol identifikátor relácie pravidelne obmieňaný. Pri odhlásení užívateľa je relácia zrušená a následne pri prihlásení je vygenerované ďalšie session id. K bezpečnosti vplýva aj fakt, že webová aplikácia je nastavená tak, aby po 15 minútach nečinnosti užívateľa odhlásila. Na to je určená položka databáze lasttime.

Do nej je ukladaný čas prihlásenia. Funkciou date_sub() je potom overované, či prihlásenie prekročilo definovaný čas.

DATE_SUB(now(),INTERVAL 15 MINUTE)

Toto je jeden zo spôsobov, ako je zabezpečené pravidelné regenerovanie identifikátora relácie. Jeden z hlavných bezpečnostných problémov – krádež relácie, je popísaný v kapitole 4.2.1

4.2 ÚTOK TYPU CROSS SITE SCRIPTING (XSS)

Tento útok sa prvýkrat objavil v roku 2000[10]. Útok spočíva v oklamaní webového servera takým spôsobom, že sám potom odosiela užívateľovi škodlivý kód HTML. Vo väčšine prípadov ide o skript. Na obsahu skriptu potom záleží, aká činnosť bude vykonaná (krádež informácií o relácii, zmena obsahu webových stránok, presmerovanie formulárov, atď). Príčina možného napadnutia systému týmto útokom je nesprávne filtrovanie výstupu webovej aplikácie. Pretože XSS je problém predávania dát, je nutné ošetrovať dáta práve v okamihu ich predávania.

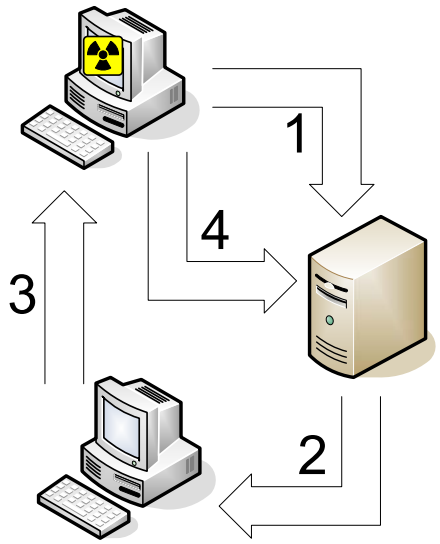
4.2.1 Krádež relácie pomocou XSS

Najjednoduchší spôsob krádeže relácie pomocou XSS je zobrazený na Obr. 5

Postup:

1. Útočník donúti pôvodnú webovú stránku, s ktorou má obeť aktívnu reláciu, aby obsahovala JavaScript pre krádež súboru cookie.
2. Webový prehliadač obete získa z tejto stránky skript a spustí ho. Skript potom automaticky presmeruje prehliadač na webový server útočníka, pričom so sebou ako časť adresy URL berie i súbor cookie s identifikátorom relácie.
3. Po obdržaní požiadavky získa aplikácia útočníka z adresy URL potrebný súbor cookie a ako odpoveď vytvorí stránku obsahujúcu skript, ktorý presmeruje prehliadač späť na pôvodnú stránku.
4. Webový prehliadač obete načíta danú stránku z webového serveru útočníka. Následne dôjde k spusteniu nového skriptu pre presmerovanie.

5. Útočník vloží ukradnutý súbor cookie do svojho vlastného prehliadača a spojí sa s pôvodným webovým serverom pod identitou obete.



Obr. 11: Krádež relácie pomocou XSS

XSS je problémom metaznakov. Teda pre vyriešenie tohto problému je teda nutné zaistiť, aby metaznaky stratili svoj riadiaci význam. Riešením je kódovanie HTML.

4.2.2 Kódovanie HTML

Pri kódovaní HTML je potrebné riadiť sa jednoduchým algoritmom zobrazeným v Tab. 2

Tab. 2: Dôležité metaznaky pri XSS

Znak	Zakódovaný znak
& (ampersand)	&amp;
" (dvojité úvodzovky)	&quot;
< (menší ako)	&lt;
> (väčší ako)	&gt;
‘ (jednoduché úvodzovky)	&#39;

Zakódovanie týchto piatich znakov ale vždy nerieši problém. Práve naopak, niekedy ďalšie problémy pridáva. Ak napríklad do príspevku je potrebné vložiť znak

< (menší ako) napríklad pri vyjadrovaní matematicke nerovnosti. Znak je zobrazovaný ako <. Preto je veľmi komplikované zamedziť všetkým možnostiam vloženia skriptu do stránky.

V PHP je pre kódovanie znakov vytvorená funkcia **htmlspecialchars().** Tá vykonáva kódovanie namiesto programátora.

Ideálnym príkladom na nutnosť použitia tejto funkcie je vypisovanie chybových správ. Napríklad chybová hláška uložená do premennej môže obsahovať riadici znak jazyka HTML. Reťazec ENT_QUOTES znamená, že sa zakódujú jednoduché aj dvojité úvodzovky.

*$sprava="Vaša e-mailová adresa je v nesprávnom tvare!
";*

echo htmlspecialchars(htmlspecialchars($sprava, ENT_QUOTES));

Výstupom takého kódu bude:

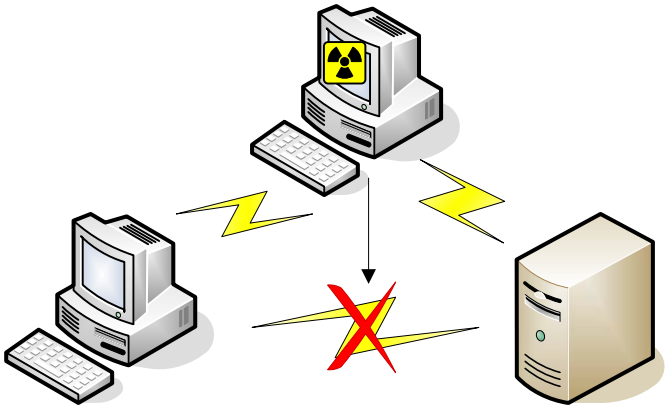
Vaša e-mailová adresa je v nesprávnom tvare!

Podobnou funkciou ako htmlspecialchars() je htmlentities(). Tá navyše dokáže kódovať aj iné špeciálne znaky. Výstupom v tomto prípade bude:

Vaša e-mailová adresa je v nesprávnom tvare!

4.3 ÚTOK TYPU MAN IN THE MIDDLE (MITM)

Ako už napovedá samotný názov tento typ útoku spočíva v tom, že útočník sa nachádza na ceste medzi užívateľom a webovým serverom, odchytáva komunikáciu a upravuje ju. Najjednoduchšie je realizácia tohto útoku pri bezdrôtovom prenose dát, kedy sú informácie posielané všetkými smermi vzduchom a teda nie je žiadny problém, aby ich ktokoľvek zachytil. Ochrana proti útoku MITM je jednoduchá - šifrovaná komunikácia napríklad pomocou SSL (viz kapitola 7)



Obr. 12: Útok typu MITM

4.4 ÚTOK TYPU SQL INJECTION

Popis problému SQL injection sa prvýkrát objavil v roku 1998. Pomocou tohto útoku je útočník schopný meniť alebo zadávať požiadavky, ktoré sa odosielajú do databáze prostredníctvom vstupov do webovej aplikácie. Samotný útok začína v okamihu, keď program vytvára dotazy založené na reťazcoch pochádzajúcich od klienta a posiela ich na databázový server bez toho, aby ošetril znaky, ktoré server považuje za špeciálne.

SQL požiadavka, ktorá je neošetrená proti útoku SQL injection, a teda vykoná neštandardnú udalosť môže vyzerať napríklad takto[2]:

$mysql_query = "SELECT * FROM Uziv WHERE UzivMeno='".$_POST["UzivMeno"]."' AND Heslo='".$_POST["Heslo"]."'"

Útočník do poľa pre užívateľské meno zadá napríklad:

honza' --

Konečná požiadavka potom vyzerá nasledovne:

*SELECT * FROM Uziv WHERE UzivMeno='honza' --' AND Heslo=''*

Útočník po zadaní vyššie uvedenej SQL požiadavky dostane od databázového serveru ako výstup údaje týkajúce sa užívateľa Honzu, vrátane hesla. Pokiaľ sú heslá v databáze uložené v nešifrovanej podobe (viz kapitola 3.2) získa pekný úlovok.

V tejto diplomovej práci je ochrana proti SQL injection riešená funkciou **mysql_real_escape_string()**. Táto funkcia pridá spätné lomítko (\) ku všetkým znakom uvedeným v Tab. 3. Funkcia je schopná brať do úvahy aj použitú znakovú sadu. Vyžaduje teda vo svojich parametroch aj identifikátor spojenia k databázi. Pracuje od PHP verzie 4.3, čo je v dnešnom svete už štandard.

Tab. 3: Znaky, ktoré „escapuje“ funkcia mysql_real_escape_string()

Znak	Význam
“ ”	medzera
“\t”	tabulátor
“\n”	nový riadok
“\r”	carriage return (tlač)
“\0”	nulový byte
“\x0B”	vertikálny tabulátor

Funkcia vyzerá napríklad takto:

```
$new_string = mysql_real_escape_string($string, $db);
```

Premenná $db obsahuje funkciu mysql_connect (pripojenie k databázi). Túto funkciu je nevyhnutné použiť na miestach, kde sa dáta ukladajú do databázi. Teda pred funkciou mysql_query().

4.5 ÚTOK TYPU TRÓJSKY KÔŇ

Útok je známy tiež pod názvom Cross-site Request Forgery (CSRF). Problém útoku, ktorý sa nazýva trójsky kôň [5] je to, že útočník dokáže prinútiť iných užívateľov, aby vznášali na web také požiadavky, ktoré by inak nevzniesli, poslaním emailu prípadne kliknutím na odkaz.

Riešením je použitie tzv. lístkového systému. Základom systému sú náhodné čísla, ktoré nie je možné uhádnuť. Systém funguje tak, že pri vykonávaní akcie, ktorá má určitý účinok sa vygeneruje náhodný reťazec a ten sa priradí k danej akcii.

Príklad [2]:

<input type=”hidden” name=”listok” value=”uV54j87vgt$aqEF4”/>

Pokiaľ je ponukou odkaz, pripojí sa lístok k parametrom adresy URL:

Áno, súhlasím

Ku každému vytvorenému lístku sa priradí názov akcie, ktorú vykonáva a uloží sa do fondu lístkov v relácii užívateľa.

Na podobnom princípe funguje v diplomovej práci ochrana proti viacnásobnému odoslaniu formulára popísaný v kapitole 3.1.3.

4.6 ÚTOK TYPU DENIAL OF SERVICE (DOS)

Slovenský názov je “odmietnutie služby”. Jedným zo spôsobov ako zabrániť útokom hrubou silou je, zablokovať prístup k účtu na určitý čas pri definovanom počte neúspešných pokusov o prihlásenie. Tu sa ale dostávame do konfliktu práve s DoS. Útočníkovi niekedy môže stačiť, keď užívateľovi zablokuje prístup k účtu (napríklad registrácia na termín skúšky v školskom informačnom systéme). K tomu nepotrebuje nič iné, len poznať užívateľské heslo. Preto by sa na podobných weboch mal používať iný spôsob obrany.

V diplomovej práci táto ochrana použitá nie je. Nakoľko som uprednostnil ochranu proti útokom hrubou silou (blokovaním účtu po 5 neúspešných prihláseniach)

4.7 ÚTOK POMOCOU PREPLNENIA VYROVNÁVACEJ PAMÄTE

Jeden z najsofistikovanejších, najzložitejších a zároveň najnebezpečnejších útokov na web je založený na zraniteľnosti práce s pamäťou webového serveru.

Nebezpečnosť spočíva v tom, že útočník je schopný po úspešnom vykonaní plne ovládnuť server a spustiť na ňom akýkoľvek príkaz.

K preplneniu pamäti dochádza vtedy, keď sú dáta zapisované do pamäte väčšej, ako množstvo pamäte, ktorá je pre operáciu vyhradená. Keď je volaná nejaká funkcia, uložia sa do zásobníka hodnoty registrov, ich parametre a premenné spolu s návratovou adresou. Útok funguje vtedy, ak je návratová hodnota prepísaná vhodným reťazcom. Tak je možné výrazne pozmeniť chod programu.

Pre úspešné vykonanie útoku je ale potrebné mať príslušné znalosti systému a preto je vykonávaný málokedy. Jeho následky však bývajú drtivé. Problémom je, že sa proti nemu bráni len veľmi ťažko, pretože funguje v dôsledku chyby na komerčnom softvéri.

5 ZAZNAMENÁVANIE A HLÁSENIE UDALOSTÍ

Platí pravidlo, že webový server by mal vracať len nevyhnutné informácie o svojej činnosti a zložení. Tým sa predíde tomu, aby potenciálny útočník dostal potrebné informácie o nastavení a štruktúre celého systému, ktoré mu umožnia preniknúť dovnútra.

Bežný užívateľ umiestňuje svoje webové aplikácie na servery firiem, ktoré ponúkajú hosting. Nemá teda webový ani databázový server priamo pod kontrolou a musí sa spoľahnúť, že služby, ktoré daná spoločnosť na svojom servery poskytuje, budú dostatočne na zaistenie bezpečnosti aplikácie.

Väčšina kvalitných webhostingových spoločností umožňujú veľkú škálu možností ako si servery prispôsobiť vlastným potrebám. Ja som webovú aplikáciu tejto diplomovej práce umiestnil na servery spoločnosti Websupport [21]

5.1 VLASTNÝ SYSTEM HLÁSENÍ

Logovanie prihlasovaní môže administrátorom dopomôcť pri odhaľovaní pokusov o uhádnutie hesla užívateľa. Pri útoku hrubou silou sa útočník skúša tisícky hesiel za sekundu. V logovacom súbore je teda zaznamenaných veľké množstvo údajov za krátky čas. Administrátor potom hneď pozná, že sa deje niečo neobvyklé a vie na vzniknutú situáciu reagovať napr. dočasným zablokovaním účtu a informovaním užívateľa.

Keďže v mojom prípade ide o jednoduchý projekt bude postačovať, aby sa pri každom pokuse o prihlásenie (stlačenie tlačitka Prihlásiť) do súboru zaznamenávali použité užívateľské meno, IP adresa a aktuálny čas akcie. Z tohoto súboru bude jednoduché identifikovať, čo, kedy a kým bolo zadávané do poľa určeného pre užívateľské meno. Obdobne by bolo možné nastaviť logovanie aj na ostatné polia.

Pre túto činnosť slúži jednoduchý skript logovanie_udalosti.php. Ten získa obsah premennej z formulárového poľa “meno“, aktuálny čas a IP adresu, z ktorej sa užívateľ prihlasuje. Otvorí súbor prihlasovanie.txt pomocou PHP funkcie fopen() na zápis a premenné do tohto súboru zapíše pomocou funkcie fputs(). Ako oddeľovač údajov sú použité pomlčky.[interval.cz]

```
May 24, 2009, 12:21:29 am-----User1-----87.244.203.134
May 24, 2009, 12:20:50 am-----Milan' -------87.244.203.134
May 24, 2009, 12:20:15 am-----Milan-----87.244.203.134
```

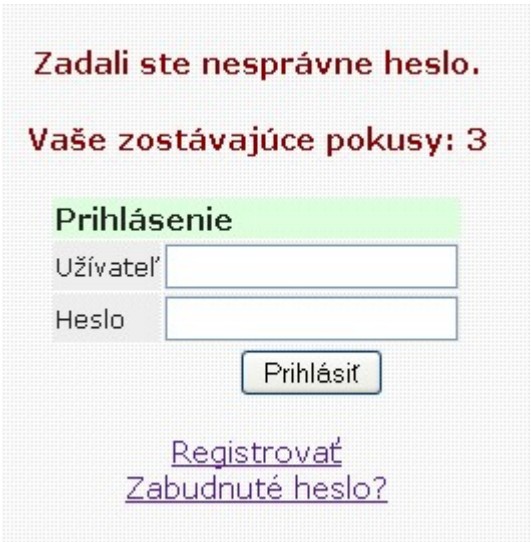
Obr. 13: Výpis z logu

Logovanie vyššie uvedených údajov je len informatívne. Oveľa dôležitejšie je efektívne zabrániť v útoku na heslo hrubou silou. Na to slúži časť kódu, kedy síce užívateľ zadal správne užívateľské meno, ale heslo už nie.

5.1.1 Princíp ochrany proti útoku na heslo hrubou silou

Pri každom užívateľovi sú do databázi pridané ďalšie 3 položky – lastloginfail, pokusy a locked. Do firstloginfail je ukladaný dátum, kedy prišlo k poslednému neúspešnému pokusu o prihlásenie. Do položky pokusy je pri vytvorení užívateľa zapísaná hodnota 5. Tá značí koľko pokusov užívateľ má na to, aby sa úspešne prihlásil. Po piatom neúspešnom pokuse sa účet zablokuje. Automaticky je odblokovaný po 15 minútach. Na to slúži tretia položka locked, ktorá je nastavená v prípade zamknutého účtu na 1. Na začiatku prihlasovania sa teda okrem mena a hesla, kontroluje aj nastavenie položky locked v databázi. Automatické odblokovanie účtu je riešené pomocou funkcie DATE_SUB(), ktorá berie údaje z lastloginfail.

lastloginfail <= DATE_SUB(now(),INTERVAL 15 MINUTE)



Obr. 14: Hlásenie o neúspešnom prihlásení

Ak je už účet zablokovaný, zobrazí sa hlásenie ako na obr. 15.

Obr. 15: Dočasne zablokovaný účet

V prípade, že užívateľ zadal neplatné dáta, mal by byť slušne upozornený na spôsobenú chybu a zobraziť tiež popis ako túto chybu napraviť. Na túto činnosť nie je príliš vhodné použiť JavaScript vzhľadom na skutočnosť, že ide o vstup zo strany klienta. Je teda potrebné ošetrenie na úrovni programovacieho jazyka. Tieto hlásenia sú rozobrané v kap. 3.1.4.

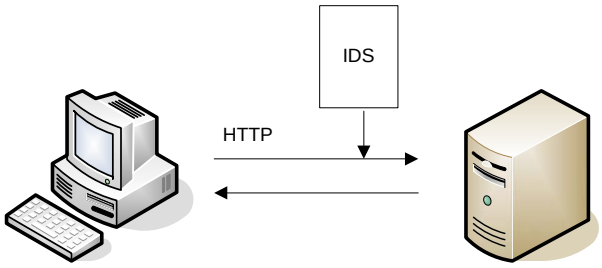
5.2 DETEKTORY PRIENIKU

Detektory prieniku sú zariadenia, ktoré monitorujú webovú komunikáciu a na základe definovaných pravidiel upozorňujú na možné útoky v reálnom čase. Skladajú sa z troch zložiek – sledovacej, analytickej a výstražnej. Sledovacia časť zbiera dáta, v tomto prípade sieťovú prevádzku. Analytická časť vykonáva rozbor zachytenej komunikácie a rozhoduje, či ide o bežnú činnosť alebo pokus o útok. Nakoniec výstražná časť na typ vzniknutej udalosti podľa naprogramovaných pravidiel reaguje. Reakcia môže byť pasívna (upozornenie v administrátorskej konzole prípadne zaznamenanie do protokolu) alebo aktívna (napr. Priamy príkaz na firewall, ktorý komunikáciu zablokuje)

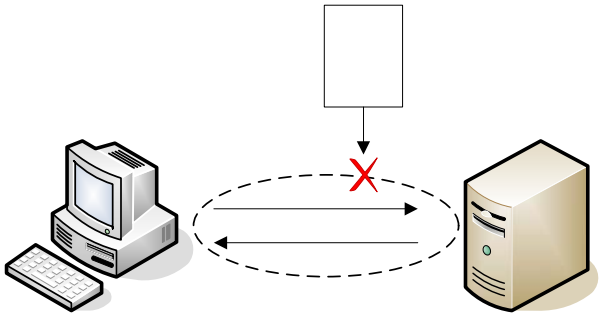
Detektory delíme na dva typy:

1. Sieťové
2. Hostiteľské

Sieťové detektory monitorujú len sieťový segment, zatiaľ čo hostiteľské detektory môžu sledovať aj ďalšie parametre systému (napr. bežiace procesy, prístupy k súborovému systému a pod.). Najväčším nepriateľom IDS je šifrovaná komunikácia cez SSL. Detektor nie je schopný rozšifrovať prenášané dáta a tie prechádzajú cez neho akoby ani neexistoval. Riešením je tzv. obrátená proxy, ktorá prichádzajúce dáta pred detektorom dešifruje.



Obr. 16: IDS v HTTP komunikácii



Obr. 17: IDS v HTTPS komunikácii

Presnosť detektorov

Presnosť je základným problémom detektorov prieniku. Existujú dva problematické prípady [1]:

- planý poplach
- falošný kľud

K planému poplachu dôjde vtedy, ak je nejaká činnosť označená za útok napriek tomu že o útok nejde. Na druhú stranu falošný kľud znamená, že útok prebehne bez

toho, aby si ho IDS všimol. Čo sa týka stupňa nebezpečnosti, je pri falošnom kľude výrazne vyššia ako pri planom poplachu.

6 PRAKTICKÉ RIEŠENIE

Prvá stránka na ktorú sa užívateľ štandardným spôsobom dostane, je úvodná stránka s prihlasovacím oknom index.php. Tu sa po zadaní prihlasovacieho mena a hesla do pripraveného formulára, môže prihlásiť. Pokiaľ ešte nebol zaregistrovaný, je to možné po kliknutí na odkaz „Registrovať“. Poslednou možnosťou je „reset“ hesla v prípade, že ho zabudne.

Obr. 18: Úvodná stránka

Pri prvom načítaní úvodnej stránky, pokračuje skript tzv. nainkludovaním hlavičky (header.php). V hlavičke sa nachádza nevyhnutné pripojenie k databázi „diplomovapraca“ v MySQL a tiež skript login.php.

Ako už názov napovedá, sú v ňom príkazy potrebné k prihláseniu užívateľa. Rozhodovacím kritériom je zistené, či bol odoslaný formulár (sú naplnené premenné meno a heslo) alebo sa potrebné údaje „vytiahnu“ zo session. Taktiež sa zistí IP adresa užívateľa, ktorá sa pravidelne kontroluje, aby v priebehu spojenia nedošlo k jej zmene.

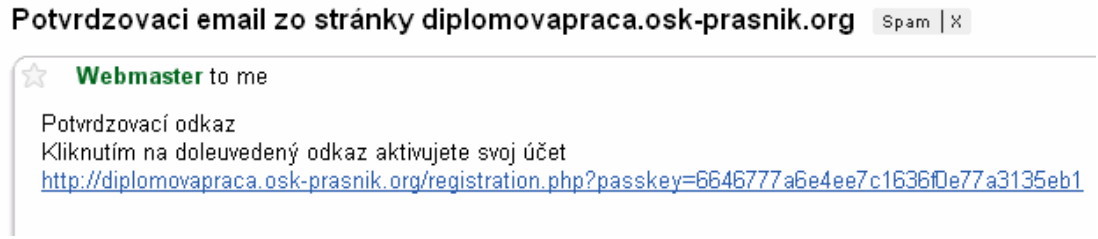
6.1 REGISTRÁCIA

Ak užívateľ ešte nemá vytvorené heslo, musí sa najprv prihlásiť. Pri registrácii musí zadať meno, emailovú adresu a heslo. Na informatívnu kontrolu sily hesla je do stránky vložený JavaScript.

Obr. 19: Registrácia užívateľa

Všetky formulárové polia sú ošetrené na úrovni programovacieho jazyka. Ak ich všetky užívateľ úspešne splní, je mu zaslaný potvrdzovací email s náhodne vygenerovaným potvrdzovacím kódom.

$confirm_code=md5(uniqid(rand()));



Obr. 20: Potvrdzovací email po registrácii

Ak naň klikne, zobrazí sa mu stránka o úspešnej registrácii. Pri prvom zadaní všetkých potrebných údajov pri registrácii sa dáta spolu s potvrdzovacím kódom zapíšu do tabuľky temp_uzivatelia. Keď je účet overený, dáta sa z tejto tabuľky vymažú a zapíšu sa do tabuľky uzivatelia. Teraz sa môže užívateľ prihlásiť.

6.2 SPÔSOB PRIHLASOVANIA

Prihlasovanie prebieha v skripte first_login.php. Na začiatku sa kontroluje, či účet nie je zablokovaný (po piatich neúspešných pokusoch o prihlásenie). Ak účet blokovaný nie je, prípadne vypršal čas blokovania, skontroluje meno a heslo oproti tabuľke v databázi. Vygeneruje session id reťazec, a spolu s IP adresou a aktuálnym časom, ich zapíše do tabuľku uzivatelia. V skripte load.php potom naplní premenné a užívateľovi všetko na hlavnú stránku vypíše.

Obr. 21: Prihlásený užívateľ

Ak nebolo prihlásenie úspešné, ale uživateľské meno bolo zadané správne, príde na radu časť skriptu, ktorá sa venuje ochrane proti útokom na heslo hrubou silou. (viz kapitola 5.1.1)

6.3 ODHLÁSENIE

K odhláseniu môže dôjsť dvojitým spôsobom:

- kliknutím na odkaz
- po vypršaní limitu (15 minút)

Odhlásenie po kliknutí na odkaz zabezpečuje skript logout.php. Ten najprv zmení session identifikátor v databáze a potom aj ostatné premenné na náhodné reťazce. Následne potom zničí session (príkazom session_destroy()).

Ak je užívateľ viac ako 15 minút nečinný, je odhlásený. Má to na svedomí funkcia logged.php, kde prestane platiť nasledujúca podmienka:

lasttime >= DATE_SUB(now(),INTERVAL 15 MINUTE)

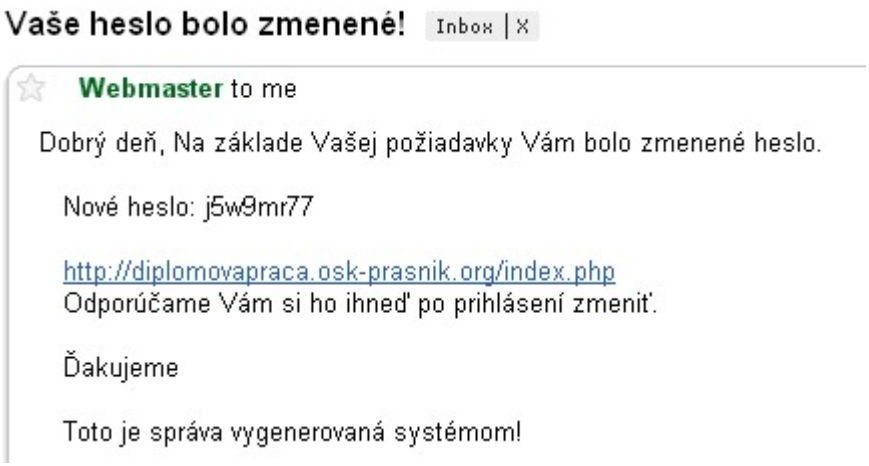
Obr. 22: Timeout spojenia

6.4 ZABUDNUTÉ HESLO

Ak užívateľ zabudne svoje heslo, klikne na hlavnej stránke na odkaz „Zabudnuté heslo“. Zobrazí sa mu formulár na zadanie emailovej adresy.

Obr. 23: Formulár pre zabudnuté heslo

Po jej zadaní je kontrolovaná jej validita pomocou funkcie eregi() (viz kapitola 3.1.4). Ďalším krokom je kontrola, či sa daná adresa v databáze nachádza. Ak áno, je užívateľovi poslaný email s novým náhodne vygenerovaným heslom.



Obr. 24: Email o zmene hesla

6.5 ZMENA HESLA

Užívateľ si môže zmeniť heslo, len keď je prihlásený. Preto sa k tejto ponuke dostane z hlavnej stránky až po úspešnom prihlásení. To je zaisťované pomocou session.

Obr. 25: Zmena hesla – ak nie je užívateľ prihlásený

Do formuláru o zmene hesla je nutné zadať heslo pôvodné a dvakrát potvrdiť heslo nové. Skript najprv overí, či je staré heslo platné a potom zapíše do databázy heslo nové.

7 SSL CERTIFIKÁTY

Každý tvorca webových aplikácií, ktoré pracujú s citlivými údajmi, by mal zabezpečiť, aby sa takéto informácie nedostali do rúk nepovolaným osobám. O to viac, že údaje medzi prehliadačom a webovým serverom sa internetom, pri použití HTTP protokolu, prenášajú v otvorenej podobe.

Pre účely šifrovania spojenia, vznikol protokol SSL (Secure Socket Layer). Ten je schopný zabezpečiť, aby boli všetky prenášané dáta, prenášané bezpečne - zašifrované. Prostriedkom pre bezpečnú komunikáciu sú certifikáty.

7.1 CERTIFIKÁT

Jednoducho povedané, certifikát nám hovorí, kto je kto. Slúži teda na ubezpečenie, že webová stránka s ktorou komunikujeme je skutočne tá, aká si myslíme alebo tá, za akú sa vydáva. Ako ale overíme, že certifikát je skutočne pravý a nebol podvrhnutý? Pretože bol podpísaný niekým komu veríme. Nemusí to byť práve niekto koho osobne poznáme, ale niekto kto je všeobecne uznávaný ako dôveryhodná autorita. Nazýva sa tiež aj certifikačná autorita.

Celá myšlienka SSL spočíva v bezpečnom prenášaní dát od serveru ku klientovi a späť. Toto je zaistené šifrovaním. Pri prvom spojení serveru s klientom pomocou SSL je klient prezentovaný certifikátom. V ňom sú najdôležitejšie nasledujúce 3 informácie:

- Časová platnosť certifikátu
- Certifikát je platný práve pre daný webový server
- Cerifikát je podpísaný niekým komu klient verí

7.2 CERTIFIKAČNÁ AUTORITA

Pre naše účely si vytvoríme vlastnú certifikačnú autoritu. Nazveme ju „demoCA“ a budeme ju používať na vytváranie personálnych certifikátov[19]. Hlavný dôvod prečo nepoužiť skutočnú dôveryhodnú certifikačnú autoritu je ten, že

je nutné za to platiť. Na druhú stranu budem certifikáty používať len pre svoju vlastnú potrebu a preto nepotrebujem overovanie zabezpečovať niekým iným.

Najprv je nutné vytvoriť si vlastnú certifikačnú autoritu, ktorou potom podpíšeme vlastný serverový certifikát. K tejto práci som použil perlovský program CA.pl od Tima Hudsona a konfiguračný súbor openssl.cnf. [15]

Novú certifikačnú autoritu vytvorí príkaz [16]:

CA.pl – newca

```
Making CA certificate ...
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.........................++++++
.................++++++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:SK
State or Province Name (full name) [Some-State]:Piestany
Locality Name (eg, city) []:Prasnik
Organization Name (eg, company) [Internet Widgits Pty Ltd]:DemoCA
Organizational Unit Name (eg, section) []:.
Common Name (eg, YOUR name) []:Milan Kazik
Email Address []:milan.kazik@gmail.com
```

Obr. 26: Vytváranie certifikátu certifikačnej autority

Po úspešnom vytvorení, získame tri súbory:

- cacert.pem (vlastný certifikát)
- cakey.pem (tajný kľúč v otvorenej forme)
- careq.pem (tajný kľúč v zašifrovanej forme)

7.2.1 Zrušenie certifikátu

Pre bezpečnosť je užitočnou funkciou možnosť certifikačnej autority zrušiť certifikát, ktorý vydala. Najčastejšie sa tak deje, ak je zistené, že bol odcudzený prípadne inak kompromitovaný. Každá CA publikuje zoznam zrušených certifikátov známy ako CRL (Certificate Revocation List). Server si tak ešte pred overením môže zistiť, či sa daný certifikát na zozname nenáchádza[19].

7.3 CERTIFIKÁT SERVERU

Teraz vytvoríme vlastný certifikát, ktorý podpíšeme certifikačnou autoritou, ktorú sme práve vytvorili. Pomocou príkazu programu *CA.pl* s parametrom *newreq* vytvoríme zašifrovaný kľúč. Príkazom *openssl req -text -noout < newreq.pem* ho dekódujeme a konečne príkazom *CA.pl –sign* podpíšeme.

Na Obr. 9 je vidieť, že certifikát bol skutočne podpísaný tajným kľúčom našej certifikačnej autority (cakey.pem). Po ukončení vyššie uvedených operácií dostaneme opäť tri súbory (newcert.pem, newkey.pem a newreq.pem). Tentokrát ide o certifikát a kľúče serveru. Pre lepšiu orientáciu ich aj na serverové premenujeme (server_cert.pem, a pod).

Pri uvádzaní systému do činnosti sa vyskytol problém s kľúčom. Webový server Apache pod operačným systémom Windows nepodporuje zašifrované privátne kľúče. Na vyriešenie tohoto problému bolo nutné použiť nezašifrovanú verziu kľúča. Novovytvorený kľúč už nevyžaduje šifrovaciu frázu (pass phrase)[17].

Použil som príkaz:

openssl rsa -in server_key.pem -out server_key.key

```
D:\wamp\bin\apache\Apache2.2.11\bin>perl CA.pl -sign
Using configuration from D:\wamp\bin\apache\Apache2.2.11\conf\openssl.cnf
Loading 'screen' into random state - done
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            80:7f:74:4b:e0:4f:e8:51
        Validity
            Not Before: May 21 21:20:02 2009 GMT
            Not After : May 21 21:20:02 2010 GMT
        Subject:
            countryName               = SK
            stateOrProvinceName       = Piestany
            localityName              = Prasnik
            organizationName          = MyCompany
            commonName                = localhost
            emailAddress              = admin@server.example.dom
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                C0:DD:B9:4D:75:12:5B:0C:B7:D1:8E:46:CA:CD:21:42:43:31:4D:AA
            X509v3 Authority Key Identifier:
                keyid:B6:95:57:96:E9:13:42:6F:A1:28:C7:48:D4:10:B3:5A:4E:98:B5:1
                E

Certificate is to be certified until May 21 21:20:02 2010 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Signed certificate is in newcert.pem
```

Obr. 27: Podpísaný certifikát

7.4 KONFIGURÁCIA WEBOVÉHO SERVERU

Akonáhle máme vytvorené potrebné certifikáty, môžeme pristúpiť ku konfigurácii webového serveru. Pre správnu funkciu systému je nevyhnutné upraviť tieto konfiguračné súbory:

httpd.conf (hlavný konfiguračný súbor webového serveru)
Na tomto mieste je nevyhnutné povoliť SSL modul mod_ssl a spoločne s tým zabezpečiť, aby sa tiež načítal hlavný konfiguračný súbor SSL serveru. Odkomentujeme pripadne dopíšeme nasledovné riadky:

LoadModule ssl_module modules/mod_ssl.so

Include conf/extra/httpd-ssl.conf

php.ini (konfiguračný súbor programovacieho jazyka PHP)
Tu je nutné tiež SSL rozšírenie povoliť:

Extension = php_openssl.dll

Najdôležitejší konfiguračný súbor je **httpd-ssl.conf**. Tu je na správnu funkciu systému potrebné zmeniť najviac nastavení.

Semafór pre zaručenie synchronizácie:

SSLMutex default

Umiestnenie www súborov:

DocumentRoot "D:/wamp/www/"

Cesty k serverovému certifikátu a tajnému serverovému kľúču:

SSLCertificateFile "D:/wamp/OpenSSL/ssl.cert/server_cert.pem"
SSLCertificateKeyFile "D:/wamp/OpenSSL/ssl.key/server_key.key"

A v neposlednom rade aj nastavenie správnych ciest logovacích súborov:

ErrorLog "logs/sslerror.log"
TransferLog "logs/sslaccess.log"

CustomLog "logs/ssl_request_log"

Syntaktickú kontrolu je možné vykonať pomocou httpd.exe s parametrom –t [18].

7.5 CERTIFIKÁT NA STRANE KLIENTA

Všetky nastavenia až do tohoto bodu sa týkali fungovania serverového certifikátu. Teraz sa pozrime na certifikát na strane klienta. Základom autentifikácie na strane klienta je schopnosť webového servera overiť, kto sa nachádza na druhej strane webového prehliadača. Webový server žiada o tzv. certifikát, na základe ktorého užívateľovi povolí či nepovolí prístup na stránku.

Pre správnu funkciu overovania je nutné ju najprv povoliť v konfiguračnom súbore httpd-ssl.conf ďalšie funkcionality.

Zapnúť overovanie klienta a nastaviť hĺbku kontroly:

SSL VerifyClient require

SSLVerifyDepth 1

A tiež definovať umiestnenie certifikátu certifikačnej autority:

SSLCACertificatePath "D:/wamp/bin/apache/Apache2.2.11/bin/demoCA/"

SSLCACertificateFile
"D:/wamp/bin/apache/Apache2.2.11/bin/demoCA/cacert.pem"

V našom prípade bude overovanie klientským certifikátom vyžadované vo všetkých stránkach webu. Užitočnou možnosťou je vymedziť len určitú časť webu, kde sa bude toto overovanie vyžadovať. To je možné nastaviť uzavretím príkazu SSL VerifyClient do direktívy Location.

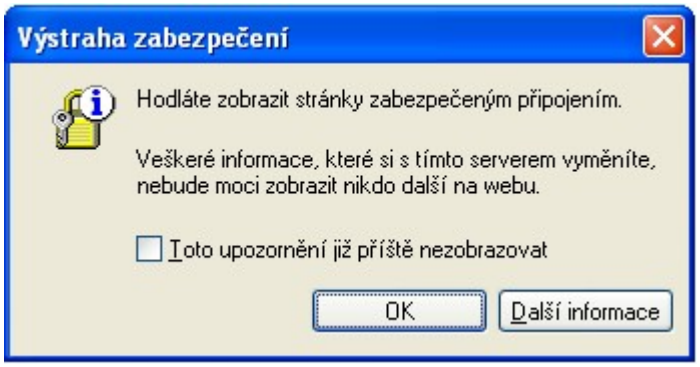
Poslednou časťou je skonvertovanie vytvoreného certifikátu do podoby, aby ho bolo možné importovať do prehliadača. Konvertovanie vyžaduje vytvorenie hesla, ktoré je potom nutné zadať pri importe. Formát certifikátu sa zmení na súbor s príponou p12. Túto činnosť vykoná nasledovný príkaz:

openssl pkcs12 -export -in server_cert.pem -inkey server_key.key -out browser-import-key.p12

Takto vytvorený certifikát už zostáva len naimportovať do prehliadača.

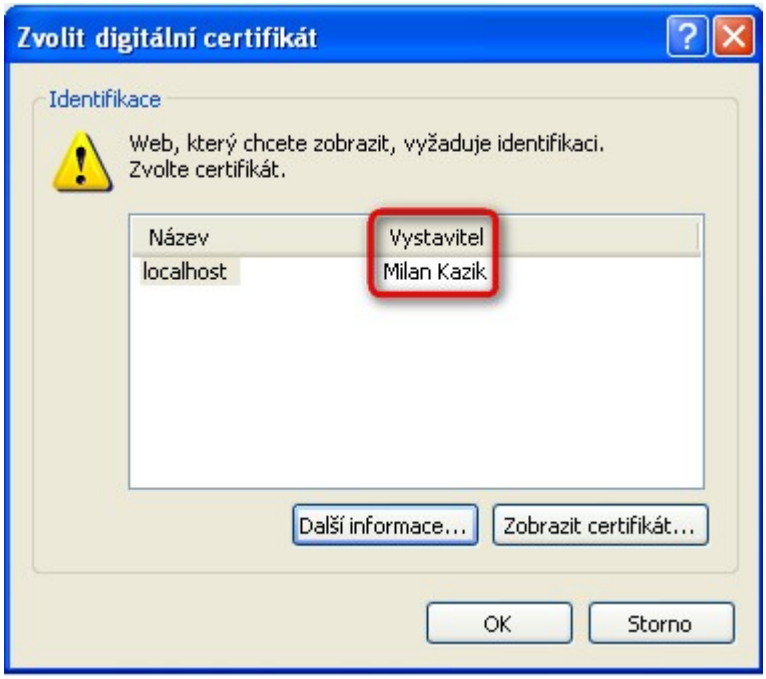
7.6 OVERENIE FUNKČNOSTI CERTIFIKÁTOV

Po otvorení prehliadača sme oboznámený s udalosťou, že vstupujeme na stránky so zabezpečeným pripojením, čo je predpoklad, že server s podporou SSL funguje správne.



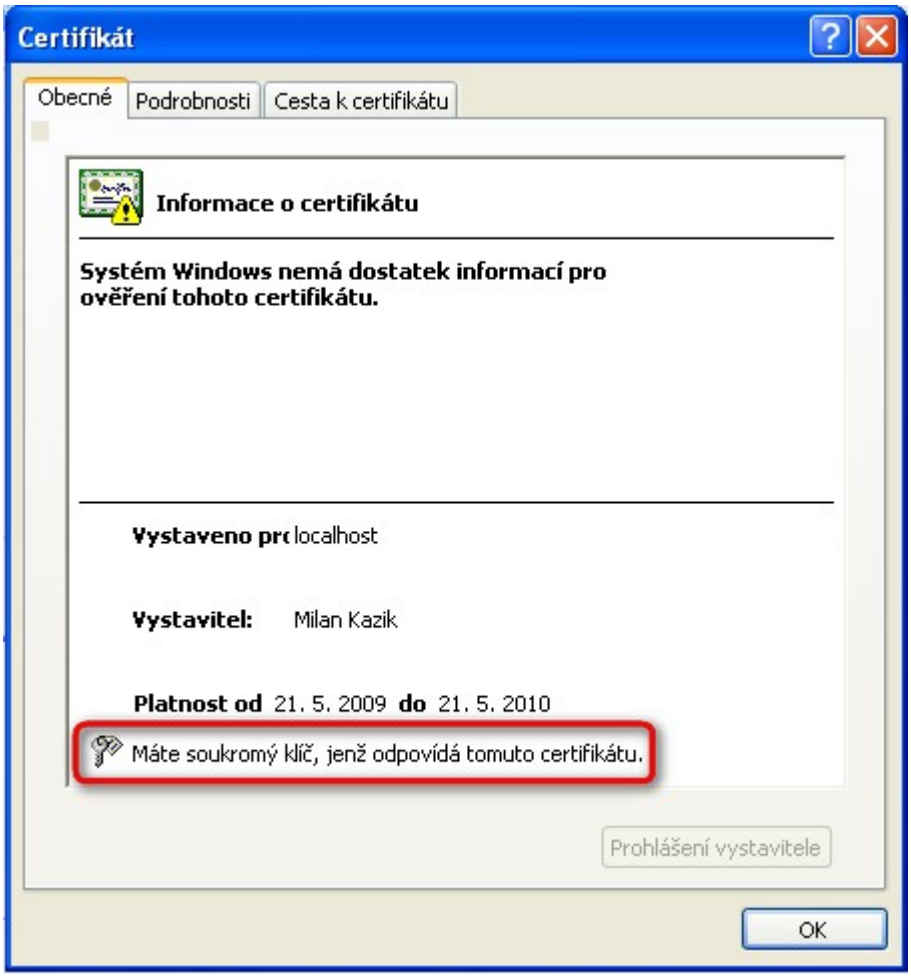
Obr. 28: Vstup na stránky so zabezpečením

Ďalším krokom je výber klientského certifikátu pokiaľ sme ho už do prehliadača naimportovali.



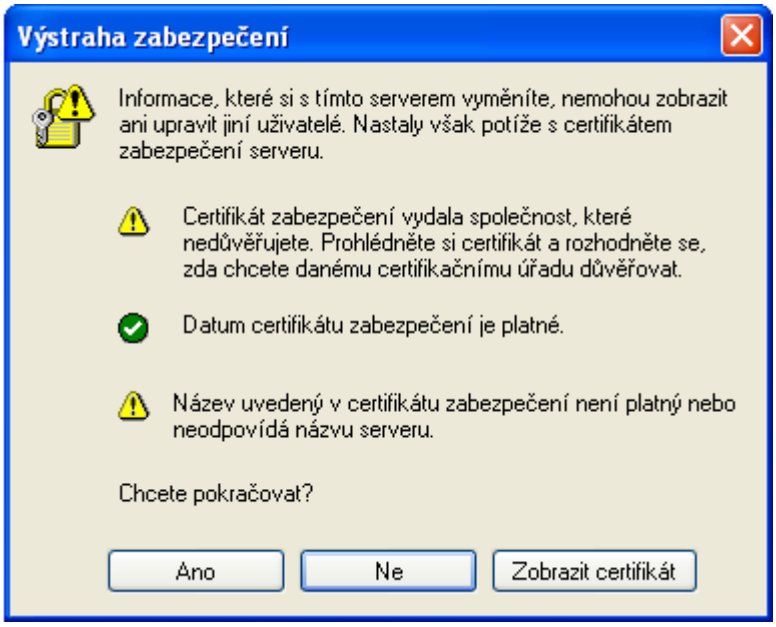
Obr. 29: Výber klientského certifikátu

Podrobnejšie informácie hovoria, že ide skutočne o náš certifikát, ktorý server eviduje ako platný. Preto umožní prístup ku svojmu obsahu.

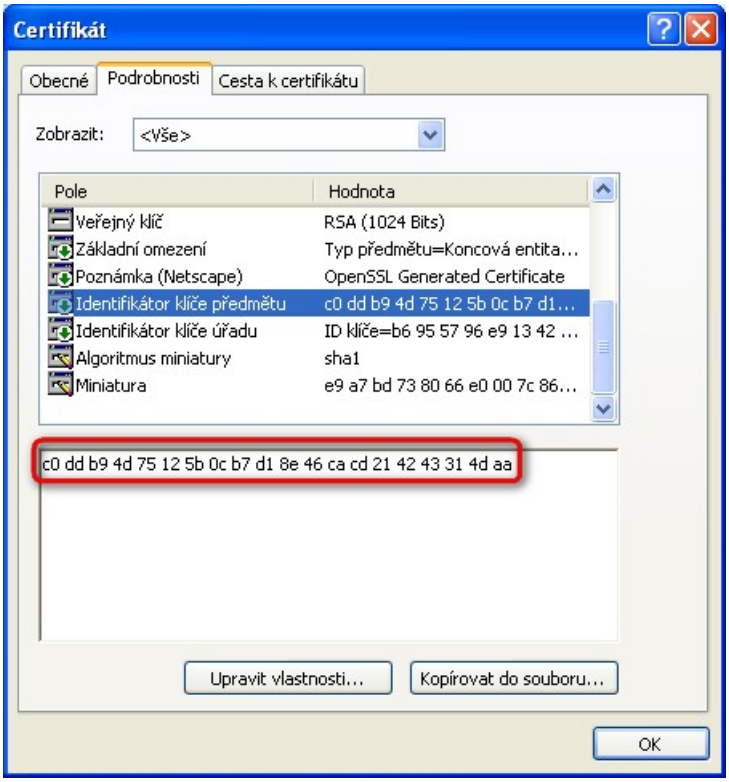


Obr. 30: Podrobnosti klientského certifikátu

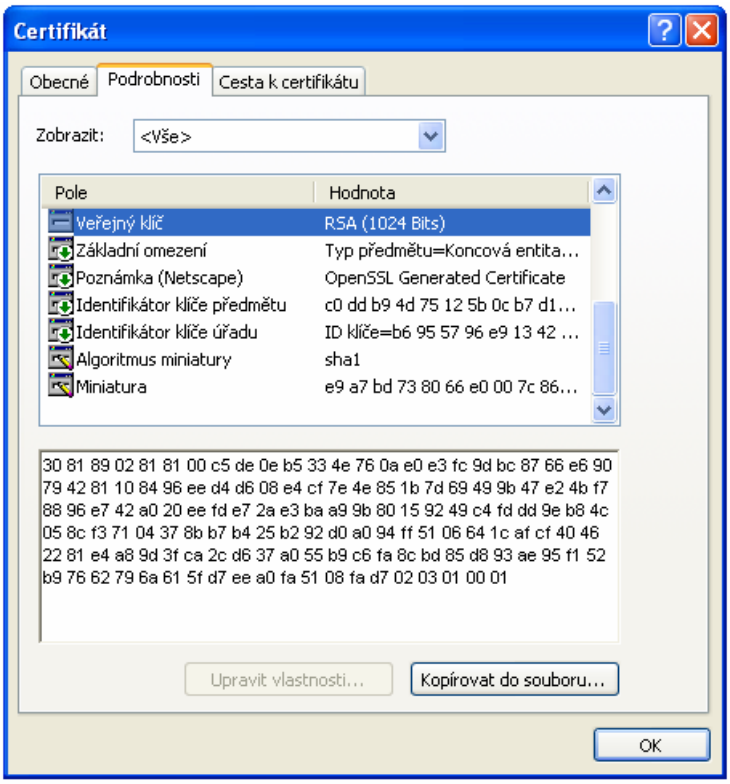
Ďalším krokom overovania je potvrdenie certifikátu serveru. Toto overenie sa zobrazuje pretože certifikát nie je overený pre prehliadač dôveryhodnou CA. Zistíme teda, či je server skutočne ten, na ktorý si myslíme, že sa prihlasujeme. V detailoch certifikátu si overíme či skutočne ide o server, na ktorý sa chceme prihlásiť. Všetky identifikátory sa musia zhodovať. Po potvrdení sa úspešne dostaneme na stránky so šifrovanou komunikáciou.



Obr. 31: Informácia o existencii serverového certifikátu



Obr. 32: Detaily certifikátu



Obr. 33: Verejný kľúč

7.7 VÝHODY A NEVÝHODY CERTIFIKÁTOV

Výhody:

- tak klient ako i server si na základe certifikát môžu byť istý s kým skutočne komunikujú
- šifrovanie – komunikácia medzi nimi je v šifrovanej forme (nie je možné použiť útok MITM)

Nevýhody:

- Zvýšené zaťaženie linky – šifrovanie so sebou prináša zvýšené nároky na prenosovú kapacitu linky medzi klientom a serverom
- Pri odcudzení klientského certifikátu sa zlodej môže vydávať za svoju obeť

8 ZÁVER

Najbezpečnejšou aplikáciou je tá, ktorá nemá žiadne spojenie s okolím. Táto veta vyjadruje najpresnejšie problematiku bezpečnosti webových aplikácií.

V diplomovej práci je rozobraná problematika bezpečnosti webovej aplikácie. Je v nej opísaných množstvo techník, ktoré sú momentálne vo svete používané. Na webe sa takmer nenachádza aplikácia, ktorá by nebola z hľadiska bezpečnosti napadnuteľná. Inteligentný a vynaliezavý útočník si vždy nájde cestu ako preniknúť cez bezpečnostnú bariéru. Preto je neustále nutné sledovať najnovšie trendy vývoja, opravy výrobcov webových serverov a databází, ktorý pravidelne vydávajú záplaty na bezpečnostné trhliny svojich výrobkov, a tiež snažiť sa pochopiť myslenie webových hackerov.

Projekt, ktorý je výstupom tejto diplomovej práce spĺňa kritéria bezpečnej aplikácie. Sú v ňom ošetrené všetky vstupy a výstupy. Heslá sú ukladané v šifrovanie podobe. Pre prihlásenie sa používa session autentifikácia. Bezpečnosť je znásobená tým, že sú použité certifikáty, a šifrovaná tak komunikácia medzi klientom a serverom. No napriek všetkým opatreniam je celkom možné, že skúsený „hacker“ by túto bariéru prelomil a vedel by ju narušiť.

Štúdium problematiky bezpečnosti bol pre mňa veľkým prínosom a budem robiť všetko pre to, aby sa moje znalosti stále zdokonaľovali.

Zoznam použitej literatúry

[1] MCCLURE, Stuart. *Web Hacking: Útoky a Obrana*. Praha : SoftPress, 2003 448 s. ISBN 80-86497-53-4

[2] HUSEBY, Sverre H. *Zranitelný kód*. Brno : Computer Press 2006. 207 s. ISBN 80-251-1180-6

[3] Štatistický server Netcraft [online]. 2008 Dostupné z: < http://survey.netcraft.com/Reports/200810/>

[4] BENÍŠEK, Pavel. *Přihlašování uživatelů do webové aplikace v PHP*. [online]. 2003 Dostupné z: http://www.abclinuxu.cz/clanky/programovani/prihlasovani-uzivatelu-do-webove-aplikace-v-php>

[5] Zope Community. [online]. Dostupné z: < http://www.zope.org/Members/jim/ZopeSecurity/ClientSideTrojan >

[6] VRÁNA, Jakub. *Ukládání souborů od uživatele.* [online]. 2005 Dostupné z: < http://php.vrana.cz/ukladani-souboru-od-uzivatele.php>

[7] TICHÝ, Jan. Nenechte si uhodnout Session ID. [online]. 2008 Dostupné z: < http://www.phpguru.cz/clanky/nenechte-uhodnout-sid>

[8] Microsoft.com. *Tipy pro vytvoření silného hesla* [online]. 2008 Dostupné z: < http://windowshelp.microsoft.com/Windows/cs-CZ/help/37565844-50dc-47e7-9260-a5a0e903db571029.mspx>

[9] VRÁNA, Jakub. *Ukládání hesel.* [online]. 2005 Dostupné z: < http://php.vrana.cz/ukladani-hesel.php>

[10] CERT – organizácia zaoberajúca sa internetovou bezpečnosťou. [online]. Dostupné z:<http://www.cert.org/>

[11] Samuraj. Jak na formuláře v PHP – bezpečnost a použitelnost. [online]. 2007 Dostupné z: <http://www.samuraj-cz.com/clanek/jak-na-formulare-v-php-bezpecnost-a-pouzitelnost/>

[12] CommView [online]. Dostupné z:< http://www.net-security.org/software.php?id=283>

[13] Manuálové stránky jazyka PHP [online] Dostupné z:http://www.php.net

[14] What is Magic Quotes GPC (magic_quotes_gpc) in PHP and the php.ini [online]. 2000 Dostupné z: < http://www.jimmysworld.org/article.html?aID=59>

[15] OpenSSL Dostupné z: <http://www.openssl.org>

[16] Creating your own CA with OpenSSL [online]. 2001 Dostupné z: <http://sandbox.rulemaker.net/ngps/m2/howto.ca.html>

[17] Thawte certifikáty [online] Dostupné z: <https://www.thawte.com/ssl-digital-certificates/technical-support/ssl/apache.html#error6>

[18] HTTPS and SSL Setup Step-by-Step guide [online]. 2008 Dostupné z: <http://www.wampserver.com/phorum/read.php?2,32986,page=1>

[19] Client Authentication with SSL [online]. 2001 Dostupné z: <http://www.freebsddiary.org/openssl-client-authentication.php >

[20] Securing A Website With Client SSL Certificates [online]. 2006 Dostupné z: <http://it.toolbox.com/blogs/securitymonkey/howto-securing-a-website-with-client-ssl-certificates-11500 >

[21] Websupport – hostingová spoločnosť [online]http://www.websupport.sk

[22] VRÁNA, Jakub. *Kontrola e-mailove adresy*. [online]. 2006 Dostupné z: <http://php.vrana.cz/kontrola-e-mailove-adresy.php

Zoznam použitých skratiek a symbolov

CA	-	Certification Authority
CGI	-	Common Gateway Interface
CRL	-	Certificate Revocation List
CSRF	-	Cross-site Request Forgery
DoS	-	Denial of Service
HTML	-	HyperText Markup Language
HTTP	-	HyperText Transfer Protocol
HTTPS	-	HyperText Transfer Protocol over SSL
IDS	–	Intrusion Detection Systems
IIS	-	Internet Information Services
ISAPI	-	Internet Server Application Programming Interface
ISP	-	Internet Service Provider
MD5	-	Message-Digest algorithm 5
MITM	-	Man In The Middle
NSA	-	National Security Agency
PHP	-	Hypertext Preprocessor
SHA	-	Secure Hash Algorithm
SID	-	Session identificator
SQL	-	Structured Query Language
SSL	-	Secure Socket Layer
TLS	-	Transport Layer Security
TCP	-	Transmission Control Protocol
URL	-	Uniform Resource Locator
XSS	-	Cross-site Scripting

PRÍLOHY

Príloha 1: CD

CD obsahuje diplomovú prácu vo formáte pdf, zdrojové a konfiguračné súbory