Vysoká škola ekonomická v Praze
Fakulta informatiky a statistiky

Bakalářská práce

2014 **Jan Menčík**

Vysoká škola ekonomická v Praze
Fakulta informatiky a statistiky
Katedra systémové analýzy

Studijní program: Aplikovaná informatika
Obor: Informatika

Aplikace e-learningu a bezpečnost dat

BAKALÁŘSKÁ PRÁCE

Vypracoval: Jan Menčík
Vedoucí práce: Ing. Jaromír Veber, Ph.D.
Oponent práce: Ing. Radim Čermák
Rok vypracování: 2014

Čestné prohlášení:

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Veškeré použité podklady, ze kterých jsem čerpal informace, jsou uvedeny v seznamu použité literatury a citovány v textu podle normy ČSN ISO 690.

V Praze dne ..... Podpis: .....

Poděkování:

Chtěl bych poděkovat vedoucímu práce Ing. Jaromíru Veberovi, Ph.D. za vstřícný přístup, odborné vedení a poskytnutí cenných rad a připomínek, které mi velmi napomohly při vypracování této práce.

Abstrakt

V této bakalářské práci je řešeno téma bezpečnostních hrozeb pro webové aplikace s praktickou částí vyhodnocení zabezpečení vybraných e-learningových aplikací. Jsou zde popsány nejčastější současné hrozby pro webové aplikace, techniky útoků a techniky zabezpečení.

Prostředí webu dalo vzniknout celé škále technik narušení bezpečnosti webových aplikací. První část práce je věnována především nejčastějším hrozbám a útokům. V další části práce jsou představeny techniky zabezpečení, a to jak obecné založené na zabezpečení protokolu, tak i techniky proti konkrétním hrozbám. Protokol, na kterém aplikace běží, je jednou z nejdůležitějších složek bezpečnosti, proto je v práci podrobněji popsáno fungování protokolu HTTPS a jeho bezpečnostních vrstev. Dále je rozebrána oblast bezpečnosti e-learningu. Čtenář bude seznámen s bezpečnostními riziky, se kterými se může setkat při provozu open-source e-learningových řešení. Na závěr teoretické části jsou popsány základní principy bezpečnostního testování pomocí metodiky definované OWASP (Open Web Application Security Project).

Praktická část práce se věnuje výsledkům testování bezpečnosti 3 vybraných open-source software systémů: Moodle, Dokeos a eFront. Testování bylo zaměřeno na hrozby popsané v teoretické části práce a využívá poznatky z příručky OWASP Testing Guide v3. U každého testovaného e-learningového systému jsou popsány jednotlivé testovací útoky, jejich výsledky a celkové bezpečnostní doporučení. V závěru praktické části je uvedeno celkové vyhodnocení testovaných systémů.

Klíčová slova

Webové aplikace, bezpečnost, e-learning, LMS, OWASP, HTTPS, penetrační testování

Abstract

This bachelor's thesis addresses the topic of security threats for web applications, with the practical part presenting a security assessment of selected e-learning applications. It describes the most common current threats for web applications, attack techniques and security techniques.

The web environment gave rise to a whole range of techniques for breaching the security of web applications, and this thesis therefore presents the most common threats. The second part of the thesis introduces security techniques, both general techniques based on securing the protocol and techniques against specific threats. The protocol on which an application runs is one of the most important security components, and therefore the thesis analyses the functioning of the HTTPS protocol and its security layers in greater detail. The following part provides an analysis of the field of e-learning security. The reader learns about the security risks which he can encounter in operating open-source e-learning solutions. The conclusion of the theoretical part describes the basic principles of security testing by means of the methods defined by the Open Web Application Security Project.

The practical part of the thesis deals with the results of security testing of three selected open-source software systems: Moodle, Dokeos and eFront. The testing was focused on threats introduced in the theoretical part of the thesis and uses the findings from the OWASP Testing Guide v3. Individual testing attacks, their results and overall security recommendations are described for every tested e-learning system. The conclusion of the practical part provides an overall assessment of the tested systems.

Key words

Web applications, security, e-learning, LMS, OWASP, HTTPS, penetration testing

Obsah práce

1 Úvod

1.1 Důvod výběru tématu

Webové aplikace zažívají od svého vzniku neustálý a rychlý rozvoj. Dnes si již nedokáži představit podnik, který by neměl žádnou podnikovou aplikaci napojenou na internet. Jednou z aplikací, která je v drtivé většině implementována jako webová, je e-learning.

Řada menších a středních podniků chce používat moderní informační systémy, ale často nemají dostatek finančních prostředků na nákup drahých licencí od renomovaných společností. Malé a střední firmy proto často poptávají open-source software (OSS) řešení, a to nejen e-learningu. Vyvstává však otázka, zda je takové řešení bezpečné, a zda není e-learning v oblasti firemní bezpečnosti podceňovaný.

Velká většina firem dnes již nebere bezpečnost v IT na lehkou váhu, ale e-learning je většinou druhořadým systémem, který si nezaslouží ve firmě takovou pozornost jako systémy na podporu klíčových business procesů.

Rozhodně nechci uměle zvyšovat důležitost e-learningu pro podnik, ale je potřeba si uvědomit, že bezpečnost e-learningu není méně důležitá, než u jiných podnikových aplikací. V e-learningu je často zapsáno know-how celé firmy, a to přehledně a uceleně. Přístup do aplikací e-learningu je často řešen přes internet, aby se zaměstnanci mohli vzdělávat i mimo své kanceláře. V neposlední řadě je e-learningová aplikace většinou integrována s podnikovou databází zaměstnanců.

To všechno dává podněty pro útoky na tyto aplikace a důvody pro jejich dostatečné zabezpečení.

Druhou stránkou věci je problematika testování bezpečnosti těchto a jiných webových aplikací, která je řešena v praktické části práce.

Bezpečnost webových aplikací mě zajímá již několik let a v posledním roce se věnuji testování bankovních aplikací, včetně e-learningu. Proto jsem si vybral téma aplikace e-learningu a bezpečnost dat.

1.2 Cíle práce

Cílem práce je představit bezpečnostní hrozby pro webové aplikace, techniky zabezpečení a blíže jednu z metod testování bezpečnosti webových aplikací. Poté se zaměřit na praktické využití popsané teorie v oblasti testování bezpečnosti open-source e-learningových aplikací. Cíle mé práce se pokusím naplnit v následujících bodech:

- popsat deset nejčastějších hrozeb pro webové aplikace současnosti a obranu proti nim;

- představit základní techniky zabezpečení a principy jejich funkce;
- poukázat na bezpečnostní hrozby pro e-learning;
- představit metodu penetračního testování a provést praktické otestování vybraných aplikací.

1.3 Metoda dosažení cíle

Obeznámit se s hrozbami pro webové aplikace a podstatou testování bezpečnosti webových aplikací. Pomocí rešerše odborných zdrojů popsat principy nejčastějších bezpečnostních hrozeb a obrany proti nim. Představit pozici e-learningu v podniku. Studiem příručky OWASP získat potřebné znalosti pro provedení testů nad vybranými systémy. Studiem zdrojů o identifikovaných chybách získat podklady pro užší zaměření testů. Na základě nabytých znalostí a získaných podkladů provést praktické testování, popsat výstupy a porovnat zvolené systémy z hlediska bezpečnosti.

1.4 Předpoklady a omezení práce

Hlavním předpokladem této práce je úspěšně vyjmout oblast zabezpečení a testování bezpečnosti e-learningových aplikací z rozsáhlé oblasti informační bezpečnosti. Zároveň však zachovat kontext s obecnými pravidly bezpečnosti v informatice.

Druhým významným předpokladem práce je cílené omezení popisovaných a testovaných hrozeb na hrozby v současnosti nejvíce časté.

Posledním, velice důležitým, předpokladem je dostupnost odborných zdrojů o zabezpečení webových aplikací v oblasti e-learningu a jejich testování. Tyto zdroje existují, ale jsou většinou v anglickém jazyce, je tedy žádoucí znalost angličtiny.

1.5 Struktura práce

Práce je strukturována do teoretické části a praktické části. Teoretická část se věnuje nejčastějším bezpečnostním hrozbám současnosti, technikám zabezpečení a představení příručky OWASP Testing Guide v3. Praktická část práce uvádí výstupy z testování vybraných OSS e-learningových systémů a srovnání jejich bezpečnosti.

1.6 Výstupy práce

Výstupy práce jsou určeny jednotlivcům, případně malým a středním firmám, které uvažují o implementaci některého OSS e-learningového řešení a mají zájem na jeho bezpečném provozu. Práce také dává možnost blíže se seznámit se současnými přístupy k testování bezpečnosti webových aplikací.

2 Bezpečnostní hrozby

V prostředí internetu existuje celá řada hrozeb pro webové aplikace. Při snaze bránit se těmto hrozbám je nejprve nutné tyto hrozby identifikovat a pochopit jejich principy. Teprve, když je hrozba známá, tak se jí můžeme efektivně bránit.

Pravděpodobně nikdy se nepodaří identifikovat všechny hrozby, z tohoto důvodu je žádoucí zaměřit se na hrozby nejvíce pravděpodobné. Tato kapitola se věnuje studii nejčastějších hrozeb pro webové aplikace, které identifikoval projekt Open Web Application Security Project (dále jen OWASP[1]) v roce 2013 a zveřejnil je v dokumentu „OWASP Top 10 – 2013“.

Jednotlivé hrozby jsou v této kapitole řazeny sestupně podle četnosti zaznamenaných útoků na webové aplikace projektem OWASP.

Zvláštní pozornost je věnována principu daného útoku a návrhu obrany. Pro větší názornost možných bezpečnostních chyb jsou některé útoky doplněny popisem praktické ukázky.

2.1 SQL injection

SQL injection je dlouhodobě nejběžnějším útokem na webové aplikace všeho druhu. Nejčastějším cílem útoku SQL injection je získat, pozměnit nebo smazat data.

Klasifikace útoku z hlediska náročnosti provedení útoku, rozšíření, schopnosti detekce a možného dopadu je uvedena v tabulce 1.

Náročnost provedení	**Rozšíření**	**Schopnost detekce chyby**	**Možný dopad na aplikaci**
jednoduché	střední	průměrná	těžký

Tabulka 1 – Klasifikace útoku [1]

2.1.1 Princip útoku

Celý útok je založen na napadení databázové vrstvy webové aplikace, kdy útočník využije slabého místa v kódu, který zajišťuje komunikaci mezi aplikací a databází. Mezi nejvíce náchylná místa patří formuláře, které předávají data od uživatele a data předávaná jako parametr v URL webové stránky. Slabé místo pak představují nedostatečně ošetřené vstupy do aplikace, které jsou následně použity v SQL dotazu. [2]

[1] OWASP je projekt zaštiťovaný OWASP Foundation, který vznikl v roce 2001 a věnuje se bezpečnosti webových aplikací. Je to nekomerční projekt, ale jeho výstupy komerční sféra používá.

Následující příklad ukazuje standardní vstup do aplikace a vykonání dotazu do databáze. Implementováno v PHP.

```
Vstup: Novák
```

Databáze provede dotaz a vrací všechny údaje z tabulky Uzivatel pro uživatele s uživatelským jménem Novák.

```
$uzivatel = $_POST['uzivatel']; //získání vstupu
z formuláře

mysql_query ("select * from Uzivatel where uzivatelske_jmeno=
'".$uzivatel."' ;");
```

Jak je vidět, tak vstup $_POST[uzivatel] není nijak ošetřen, útočník může vyplnit cokoli, například:

```
test' or 1=1 --
```

Při výše uvedeném vstupu vznikne dotaz:

```
select * from Uzivatel where uzivatelske_jmeno= 'test' or 1=1 --
' ;
```

Tento dotaz vrací všechny záznamy z tabulky Uzivatel, protože podmínka „WHERE uzivatelske_jmeno=test or 1=1“ bude splněna vždy. Pomocí „--“ dojde k převedení na komentář tak, aby zbytek původního dotazu nezpůsoboval chybu v SQL dotazu.

Na tomto příkladu je vidět, že použití SQL injection na neošetřené vstupy je poměrně jednoduché i pro méně zkušené útočníky.

2.1.2 Obrana

V mnoha zdrojích s tématikou SQL injection se vyskytuje doporučení escapovat znaky. Pro většinu programovacích jazyků existují explicitní funkce, které escapování znaků zajišťují. Příklad uvedený v přechozí části by mohl být upraven například takto:

```
vstup: test' or 1=1 --
```

$uzivatel = mysql_escape_string($_POST['uzivatel']);

```
mysql_query ("select * from Uzivatel where uzivatelske_jmeno=
'".$uzivatel."' ;");
```

Funkce mysql_escape_string upraví vstup následovně:

```
test\' or 1=1 --
```

Výsledný dotaz nic nevrací. Takovéto ošetření vypadá jednoduše a spolehlivě, ale je třeba si uvědomit, že SQL injection útoky mohou být mnohem více sofistikované a prosté escapování znaků rozhodně není dostatečnou ochranou.

Zkušenější útočníci mohou využít faktu, že je escapování mnohými programátory považováno za dostatečnou ochranu, ve svůj prospěch. Častou chybou je, že programátor použije escapovací funkci, ale zápis dotazu nedonutí útočníka vůbec použít vyhrazené znaky[2]:

```
vstup: test or 1=1 --
```

$clanek = mysql_escape_string($_GET['clanek']);

```
mysql_query ("select * from Clanek where ID= ".$clanek.";");
```

V tomto případě dojde k vypsání všech údajů z tabulky Clanek. Vypsání všech článků z databáze nepředstavuje problém, ale toto slabé místo představuje ohrožení pro celou aplikaci. Útočník není prakticky nijak omezen a může se dostat k údajům i v jiných tabulkách:

```
vstup: 12345 union select 1, 2, Heslo, uzivatelske_jmeno from
Uzivatel --
```

Za předpokladu, že má tabulka Clanek 4 sloupce, vypíše dotaz kompletní přehled uživatelských jmen a hesel z tabulky Uzivatel. Kolik má tabulka sloupců zjistí útočník poměrně rychle, nejčastěji metodou pokus omyl, kdy postupně zvyšuje počet sloupců v dotazu, až do okamžiku, kdy nedojde k chybě. [**3**]

Jak je vidět, prosté escapování není vždy dostatečné, proto je dobré doplnit ochranu webové aplikace i dalšími prvky:

- vhodně omezit práva uživatele, se kterým aplikace přistupuje do databáze;
- doplnit escapování kontrolou vstupu na základě očekávané hodnoty (pokud aplikace například standardně dostává na vstupu číslo, mělo by dojít k validaci na číselnou hodnotu, dobrým opatřením je také kontrola délky vstupního řetězce a její omezení);
- zamezit použití jazyka SQL ve vstupním řetězci (výrazy SQL jazyka můžeme vhodnou funkcí ze vstupního řetězce odfiltrovat nebo nahradit neutrálními znaky);
- zabránit zobrazování chybových hlášek databáze v aplikaci (zobrazené hlášky mohou útočníkovi značně zjednodušit útok).

2.2 Prolomení ověřování identity a chybný session management

Autentizace uživatelů a správa session jsou další velmi kritickou oblastí dnešních webových aplikací. Celý problém je komplikován faktem, že spolehlivé řešení session a autentizace je potřeba správně navrhnout již na počátku vývoje webové aplikace. [**4**]

[2] Vyhrazené znaky: znaky, které escapovací funkce ošetří (například přidáním zpětného lomítka)

Cílem útoku na session management nebo systém autentizace je získání neoprávněného přístupu do webové aplikace, případně krádež identity oběti.

Klasifikace útoku z hlediska náročnosti provedení útoku, rozšíření, schopnosti detekce a možného dopadu je uvedena v tabulce 2.

Náročnost provedení	**Rozšíření**	**Schopnost detekce chyby**	**Možný dopad na aplikaci**
střední	velmi rozšířené	průměrná	těžký

*Tabulka 2 – Klasifikace útoku [**1**]*

2.2.1 Princip útoku

Princip útoku spočívá ve využití chybně navržené autentizace nebo managementu session pro ovládnutí uživatelského účtu oběti. Mezi běžné bezpečnostní nedostatky patří:

- hesla ukládaná do databáze jsou ukládána jako prostý text;
- identifikace session je předávána v URL adrese;
- identifikace session se pro jednoho uživatele nemění;
- sezení (session) nemá nastavenou dobu platnosti (timeout);
- hesla, identifikace session a další citlivé údaje nejsou přenášeny zabezpečeným protokolem. [**5**]

Hesla uložená jako text jsou pro útočníka, například při útoku SQL injection, vítaným ulehčením práce a dnes se s takto nedbalým přístupem téměř nesetkáme. Co lze naopak relativně často pozorovat, je použití zastaralých hashovacích funkcí.

Identifikace session předávaná v URL adrese je velmi zásadní bezpečnostní hrozba a v rozsáhlejších aplikacích se jen těžko dodatečně opravuje. Session řeší problém bezstavovosti protokolu HTTP(S), proto se session a jejich předávání nevyhne prakticky žádná webová aplikace. Předávání identifikace v URL adrese je ovšem velmi nebezpečné, protože ji útočník může získat, například vylákáním odkazu na aplikaci od oběti, která je v tu dobu přihlášená. Útočníkovi pak většinou stačí jen změnit identifikaci své session za identifikaci oběti a je přihlášen jako oběť.

Pokud se identifikace session nezmění, například při řádném odhlášení uživatele, a útočník tuto identifikaci zná, tak získává trvalý přístup k účtu oběti, kdykoli se oběť přihlásí do aplikace.

Pokud nemá session nastavený timeout, tak se značně zvyšuje riziko její krádeže, a to zejména v případech, kdy uživatel přistupuje z veřejného počítače a řádně se z aplikace neodhlásí (například jen uzavře záložku prohlížeče). Útočníkovi stačí znovu otevřít aplikaci a je v sezení oběti. [**4**]

Krádež přihlašovacích údajů pomocí odposlechu komunikace mezi uživatelem a serverem není snadná, ale pro zkušeného útočníka nepředstavuje nepřekonatelný problém. Předpokladem pro tento útok je to, že aplikace nevyužívá pro předávání citlivých údajů zabezpečený protokol HTTPS, ale běžný http, který veškerá data předává jako běžný text.

2.2.2 Obrana

Jak je vidět, tak existuje mnoho způsobů, jak provést útok na session nebo autentizaci uživatele. Obrana proti útokům tohoto typu spočívá především v dobré informovanosti návrhářů a vývojářů webových aplikací o možnostech útoků. Jedná se o komplexní problém, který si vyžaduje komplexní řešení, které je navrhnuté ještě před začátkem vývoje aplikace.

Samotné řešení obrany není příliš složité a většinou stačí použít obecně ověřené a uznávané postupy. Postup řešení session managementu a autentizace by měl být jasně vymezen a striktně dodržován ve všech částech aplikace a po celou dobu vývoje.

2.3 Cross-Site Scripting (XSS)

Cross-Site Scripting, česky skriptování mezi stránkami, nejčastěji uváděný pod zkratkou XSS, je velmi rozšířený typ útoku, který využívá slabin webových aplikací. Většinou je využíván pro nenápadné získání uživatelských dat oběti, nikoli k poškození webové aplikace. [**6**]

Klasifikace útoku z hlediska náročnosti provedení útoku, rozšíření, schopnosti detekce a možného dopadu je uvedena v tabulce 3.

Náročnost provedení	**Rozšíření**	**Schopnost detekce chyby**	**Možný dopad na aplikaci**
střední	velmi rozšířené	vysoká	střední

*Tabulka 3 – Klasifikace útoku [**1**]*

2.3.1 Princip útoku

XSS je technika, při které útočník podstrčí na webovou stránku vlastní kód (skript), který se při navštívení infikované stránky provede. Skript je napsán v jazyce, který podporuje hostitelská webová aplikace nebo webový prohlížeč. Nejpoužívanějším jazykem pro XSS útok je Java Script, který podporují všechny moderní prohlížeče.

Existují dva základní typu XSS útoků:

- reflected, kdy je skript proveden ihned;
- stored, kdy je skript uložen do databáze a je prováděn opakovaně, a to při každém načtení infikovaného obsahu.

XSS útok typu reflected je běžně realizován vložením skriptu přímo do URL webové stránky a následném podstrčení infikované URL oběti. Skript je proveden v okamžiku, kdy oběť na link klikne.

Při XSS útoku typu stored je script nejdříve uložen do databáze, například ve vzkazu v diskusi, a následně opakovaně prováděn při každém zobrazení infikovaného obsahu. [**7**]

Fakt, že se pro útok použije programovací jazyk, představuje pro útočníka velmi variabilní nástroj. Útočník může uživatele například přesměrovat na podvodnou stránku, získat údaje z prohlížeče nebo zaznamenat úkony uživatele.

2.3.2 Obrana

Obrana proti XSS útokům je velmi jednoduchá, avšak při vývoji webových aplikací často opomíjená. Cílem obrany je zamezit spuštění jakýchkoli útočných skriptů, a to bez ohledu na programovací jazyk, který využívají.

Nejúčinnější obranou je, podobně jako v případě SQL injection, ošetření vstupů do aplikace. Ošetření spočívá v nahrazení nebo odstranění znaků specifických pro skriptovací jazyky a HTML, konkrétně < a >. Většina běžných programovacích jazyků opět nabízí explicitní funkce, které provedou nahrazení za znakové entity.

V případě, kdy jsou data vypisována z databáze, může k ošetření dojít až před vypsáním těchto dat na obrazovku. Útočný skript databázi nijak neovlivní. [**8**]

Příklad neošetřeného a ošetřeného výstupu:

```
$vystup =
"<script>window.location.href=\"http://utocnik.cz\"</script>";
echo $vystup;
```

Zde dojde k provedení skriptu a přesměrování uživatele na podvodnou stránku http://utocnik.cz.

```
$vystup =
"<script>window.location.href=\"http://utocnik.cz\"</script>";
echo htmlspecialchars($vystup);
```

V druhém příkladu je výstup ošetřen funkcí, která převádí < a > na znakové entity. Na obrazovku se vypíše neškodný text:

```
&lt;script&gt;window.location.href=&quot;http://utocnik.cz&quot;
&lt;/script&gt;
```

Většina moderních prohlížečů převede znakové entity zpět na původní znaky, ale k provedení skriptu nedojde, protože jde pouze o úpravu textového výstupu prohlížečem.

2.4 Neoprávněný přístup k objektu aplikace

Cílem útoku je získat přístup k datům, souborům nebo funkcím aplikace, ke kterým nemá útočník oprávnění. K získání přístupu využije slabinu v řízení přístupu. [**9**]

Klasifikace útoku z hlediska náročnosti provedení útoku, rozšíření, schopnosti detekce a možného dopadu je uvedena v tabulce 4.

Náročnost provedení	**Rozšíření**	**Schopnost detekce chyby**	**Možný dopad na aplikaci**
jednoduché	střední	vysoká	střední

Tabulka 4 – Klasifikace útoku [1]

2.4.1 Princip útoku

Základním předpokladem pro možnost použití tohoto útoku je existence přímých referencí (nejčastěji odkazů) na datové objekty (soubory, adresáře, funkce), dále pak nedostatečná kontrola práv přístupu uživatele.

Při útocích tohoto typu je útočník většinou řadovým uživatelem dané aplikace se základními právy přístupu. Takový útočník nemusí při svých pokusech budit pozornost administrátorů, dokonce může mít jejich důvěru, například pokud se jedná o interního zaměstnance. [**9**]

Provedení útoku je velice jednoduché a spočívá nejčastěji v prosté změně parametru v odkazu. Mějme například odkaz na detail účtu přihlášeného uživatele s identifikací 123 (v tomto případě útočníka), který aplikace vygeneruje na základě informace ze session.

```
<a href="./detail_uctu.php?uzivatel_id=<?php echo
$_SESSION['uzivatel_id']?>">Můj účet</a>
```

Je vytvořen HTML kód a odkaz „Můj účet“ zobrazen v prohlížeči.

```
<a href="./detail_uctu.php?uzivatel_id=123">Můj účet</a>
```

Při standardním použití odkazu je zobrazena stránka s detailem účtu přihlášeného uživatele. Aplikace získá data z databáze pro identifikaci uživatele předanou v URL.

```
http://stranka.cz/detail_uctu.php?uzivatel_id=123
```

V případě, že aplikace na stránce s detailem znovu nezjišťuje identitu uživatele, pak při změně parametru dojde k zobrazení detailu cizího účtu.

```
http://stranka.cz/detail_uctu.php?uzivatel_id=321
```

2.4.2 Obrana

Efektivní obrana zahrnuje tři základní pravidla:

- vyvarovat se použití přímých referencí na objekty aplikace;
- dbát na důslednou kontrolu práv přístupu při každé operaci uživatele, která je právy podmíněna;
- dostatečně otestovat aplikaci po stránce řízení uživatelského přístupu.

Přímým referencím se lze ve většině případů zcela vyhnout a uživateli je nezobrazovat. Ve výše uvedeném příkladu by stačilo získávat identifikaci pro dotaz do databáze přímo ze session.

Při každé operaci, která není obecně přístupná všem, je třeba kontrolovat oprávnění přihlášeného uživatele. Nelze jen předpokládat, že se uživatel k takové operaci nedostane, protože se mu nikde nezobrazí příslušný odkaz.

Testy jsou v tomto případě poměrně jednoduché a schopnost detekce chyby je vysoká. Tester může jednoduše měnit parametry v referencích na datové objekty a zjišťovat, zda aplikace neumožní neoprávněný přístup. [**10**]

2.5 Chyby v konfiguraci

Cílem je identifikovat a využít slabinu v konfiguraci aplikace pro získání kontroly nad aplikací, krádež dat nebo k monitorování činnosti uživatelů. Chyba v konfiguraci může také usnadnit překonání ochrany proti ostatním typům útoků.

Útoků, při kterých je využita chyba v konfiguraci, v posledních letech přibývá. Zatímco v roce 2010 se tyto útoky řadily v žebříčku OWASP TOP 10 na 6. místo, v roce 2013 jsou na 5. místě. Vyvstává tedy otázka, zda nejsou tyto, na první pohled primitivní, chyby podceňovány. [**1**]

Klasifikace útoku z hlediska náročnosti provedení útoku, rozšíření, schopnosti detekce a možného dopadu je uvedena v tabulce 5.

Náročnost provedení	**Rozšíření**	**Schopnost detekce chyby**	**Možný dopad na aplikaci**
jednoduché	střední	vysoká	střední

*Tabulka 5 – Klasifikace útoku [**1**]*

2.5.1 Princip útoku

Chyba v konfiguraci může mít mnoho podob. V následujících bodech je uveden přehled nejčastějších chyb:

- server podporuje jinou verzi implementačního jazyka, než v jaké je napsána aplikace;

- verze provozované aplikace, OS serveru nebo databáze není aktuální;
- v aplikaci nebo databázi existují výchozí hodnoty z instalace;
- instalační nebo konfigurační soubor nebyl z aplikace odstraněn.

V případě, kdy se provozovatel aplikace nedozví o změně podporované verze implementačního jazyka, může dojít k ignorování některých funkcí, které jsou z nové verze vyřazeny nebo nahrazeny jinými. Tento problém se běžně vyskytuje u jazyka PHP, který se rychle rozvíjí a nasazení nové verze je poměrně časté.

Provozování neaktuální aplikace, především v případech, kdy jde o rozšířenou aplikaci, je značně nebezpečné. Důvod je prostý, na starší verze aplikací existují veřejně dostupná hlášení o chybách, které může útočník zneužít.

Při instalaci webové aplikace dojde ve většině případů k automatickému vytvoření výchozího účtu pro každou uživatelskou roli. Tato výchozí nastavení aplikace mohou být útočníkovi známá. [**11**]

2.5.2 Obrana

Především je důležité udržovat integritu verzí na všech úrovních aplikační infrastruktury, zejména mezi serverem, databází a aplikací. Důležitá je spolupráce mezi vývojáři, správci aplikace a poskytovatelem serverových služeb. Vzájemná informovanost všech zúčastněných stran a dobrý management změn pomohou předcházet chybám v konfiguraci.

Ostatní rizika v podobě zapomenutého instalačního souboru nebo ponechání výchozích účtů v databázi jsou zapříčiněna lidskou nedbalostí. Pro eliminaci těchto rizik postačí doplnění bezpečnostních testů o scénáře, které odhalí případné chyby tohoto typu. [**12**]

2.6 Expozice citlivých dat

Běžným cílem útoku je získání neoprávněného přístupu do aplikace nebo odcizení citlivých dat uložených v databázi, která nejsou dostatečně nebo vůbec šifrována. K získání těchto dat mohou být použity různé metody útoku, běžně je to napadení databáze pomocí SQL injection. [**13**]

Klasifikace útoku z hlediska náročnosti provedení útoku, rozšíření, schopnosti detekce a možného dopadu je uvedena v tabulce 6.

Náročnost provedení	**Rozšíření**	**Schopnost detekce chyby**	**Možný dopad na aplikaci**
vysoká	malé	běžná	těžký

Tabulka 6 – Klasifikace útoku [1]

2.6.1 Princip útoku

Útočník získá citlivá data z databáze, například pomocí SQL injection. Citlivá data jsou například hesla uživatelů. Tato data bývají běžně šifrována a v databázi jsou uložena jako hash, neboli šifrovaný otisk původního textu.

Útočník se většinou nesnaží prolomit šifrování přímo, ale využije vlastností daného šifrovacího algoritmu. Základní vlastností hashovacích funkcí je, že k jednomu textovému řetězci vrátí vždy identický otisk. Pro běžné hashovací funkce, jako je MD5 nebo pokročilejší SHA, existují veřejné databáze otisků. Útočník může získaná hesla porovnat s takovou databází a je vysoce pravděpodobné, že v některých případech nalezne shodu, tedy heslo v textové podobě, které může použít pro přihlášení do aplikace.

Další možné ohrožení představuje využívání automatického šifrování na straně databáze. Toto zabezpečení je efektivní v případě, kdy dojde ke krádeži přístupu k databázi, nikoli při získání dat SQL injection nebo odposlechnutí útočníkem. Data jsou totiž šifrována pouze v databázi, pokud tedy databáze dostane požadavek od aplikace, tak odesílaná data rozšifruje a útočníkovi se zobrazí jako text. [**13**]

2.6.2 Obrana

Pro efektivní šifrování, které nebude jednoduše napadnutelné, existují dva základní přístupy, ze kterých jsou odvozeny ostatní postupy při šifrování obsahu jednosměrnými kryptografickými funkcemi[3]:

- doplnění šifrovaného řetězce o kryptografickou sůl;
- použití šifrovacích funkcí, které umožňují iterace.

Kryptografická sůl představuje náhodný řetězec znaků, kterým je doplněn text určený k zašifrování. Vnesením prvku náhodnosti je zajištěno, že pro dva stejné vstupy (například hesla) vzniknou dva různé otisky. Případný útočník poté již není schopen z otisku získat původní heslo slovníkovým útokem nebo porovnáním s databází otisků. Použití kryptografické soli má své pravidla. Je potřeba zajistit unikátnost přidávaného řetězce, jeho dostatečnou délku a dostatečnou náhodnost. [**14**]

Opakovaná aplikace šifrovací funkce také znemožní použití slovníkového útoku nebo porovnání s databází otisků. Běžné funkce jako MD5 se na opakované použití příliš nehodí, lepší je použít funkce, které iterace přímo podporují. Při dostatečném počtu opakování má tento přístup ještě vedlejší efekt, kterým je zpomalení útočníka při slovníkovém útoku. [**15**]

[3] Jednosměrná funkce: funkce, ke které neexistuje inverzní funkce, takže z výstupu funkce nelze jednoduše odvodit její vstup. [**47**]

Oba přístupy poskytují značné zvýšení bezpečnosti šifrovaných dat, nejsou však neprolomitelné. Zkušený útočník s dostatkem času a výpočetního výkonu může zjistit počet iterací nebo pozici kryptografické soli v původním řetězci a následně upravit program, který provádí slovníkový útok. Tomu se jen velmi obtížně brání, částečným řešením může být unikátnost a složitost šifrovacího postupu, který útočníka odradí. Problém je u aplikací s otevřeným kódem, kde se útočník může přímo podívat, jak je zabezpečení řešeno.

I zde platí obecné pravidlo, že pro přenos citlivých údajů je žádoucí použít protokol HTTPS, aby v případě odposlechu komunikace nebylo snadné data rozšifrovat.

2.7 Chybějící kontrola úrovně přístupu

Problém nedostatečné kontroly úrovně přístupu se vyskytuje především v rozsáhlých aplikacích, kde existuje více uživatelských rolí, a tedy i více úrovní přístupu k jednotlivým částem aplikace. Útočník nemusí nic ovlivňovat, stačí, když využije neodhalené chyby.

Klasifikace útoku z hlediska náročnosti provedení útoku, rozšíření, schopnosti detekce a možného dopadu je uvedena v tabulce 7.

Náročnost provedení	**Rozšíření**	**Schopnost detekce chyby**	**Možný dopad na aplikaci**
nízká	střední	běžná	střední

*Tabulka 7 – Klasifikace útoku [**1**]*

2.7.1 Princip útoku

Útok většinou provádí legitimní uživatel aplikace, který zjistí, že má přístup i tam, kam by ho mít neměl. Princip se velmi podobá útoku neoprávněný přístup k objektu aplikace. Rozdílem je, že útočník nemusí nic aktivně měnit. Aplikace může například podřízenému pracovníkovi zobrazovat odkazy na stránky určené pouze nadřízenému a podobně. Společnou vlastností je zanedbání kontroly přístupu uživatele.

2.7.2 Obrana

Zvláště u rozsáhlých aplikací s mnoha uživatelskými rolemi je zásadním prvkem obrany dobré otestování celé aplikace. Přístup k funkcím, které mají být dostupné jen některým uživatelům, je potřeba otestovat vždy se všemi uživatelskými rolemi.

Dobře navržená a přehledná hierarchie uživatelských rolí je také nedílnou součástí prevence. Pokud je předem známé, co která uživatelská role může vykonávat, je mnohem jednodušší úrovně přístupu implementovat.

2.8 Podvržení požadavku webové aplikace (CSRF)

Cílem útoku cross-site request forgery (CSRF), česky podvržení požadavku webové aplikace jinou stránkou, je podstrčit uživateli http požadavek na stránce vytvořené útočníkem, který bude vykonán aplikací, do které je oběť přihlášena.

Klasifikace útoku z hlediska náročnosti provedení útoku, rozšíření, schopnosti detekce a možného dopadu je uvedena v tabulce 8.

Náročnost provedení	**Rozšíření**	**Schopnost detekce chyby**	**Možný dopad na aplikaci**
střední	střední	vysoká	střední

*Tabulka 8 – Klasifikace útoku [**1**]*

2.8.1 Princip útoku

Úspěšné provedení útoku je podmíněno dobrou znalostí aplikace útočníkem a neopatrností oběti.

Princip útoku je poměrně jednoduchý. Útočník vytvoří podvodnou stránku, která odešle HTTP požadavek na cílovou aplikaci, kdykoli na ní někdo vstoupí. Tento požadavek musí být skutečný, cílová aplikace ho musí umět zpracovat. Požadavek bývá na útočníkově stránce zamaskován, například jako obrázek. [**16**] [**17**]

```
<img src="http://www.cilova-
aplikace.cz/posli_penize.php?castka=2000&ucet=123456789"
alt="vtipný obrázek se nepodařilo načíst" />
```

Pochopitelně se o žádný obrázek nejedná a oběti se zobrazí pouze alternativní text, ale vytvoří se požadavek na stažení obrázku.

```
GET: /posli_penize.php?castka=2000&ucet=123456789 HTTP/1.1

Host: www.cilova-aplikace.cz
```

V případě, kdy bude oběť zároveň přihlášená v cílové aplikaci a požadavek „posli_penize.php?castka=2000&ucet=123456789“ bude legitimní, dojde k jeho vykonání cílovou aplikací.

Pokud není uživatel přihlášen nebo nemá dostatečná oprávnění, tak se požadavek neprovede. Aby útočníci zvýšili svou šanci na úspěch, tak často přistupují k metodám sociálního inženýrství, vybírají si méně zkušené uživatele internetu nebo podnikají plošné útoky na velké množství uživatelů.

2.8.2 Obrana

Plošná ochrana všech operací, které může uživatel provádět je poměrně náročná a většinou zbytečná. Obrana by měla směřovat k citlivým a důležitým operacím.

Jednou z metod obrany je využití unikátních identifikátorů, tzv. tokenů. Token je náhodný řetězec, který je odesílán spolu s HTTP požadavkem, například v URL adrese nebo jako skryté formulářové pole. Token je většinou vytvořen při přihlášení uživatele a platí pouze po dobu jeho přihlášení, při dalším přihlášení je vygenerován token nový. To útočníkovi značně znesnadňuje podvrhnout platný požadavek. [**16**]

Příklad bezpečnostního tokenu v URL adrese:

```
http://www.stranka.cz/admin/index.php?akce=pridat_uzivatele&token=5a2951e0c2f3363e91884deeeadb2d52
```

Pro správu bezpečnostních tokenů existují pro řadu běžných programovacích jazyků specializované knihovny, které značně usnadní implementaci ochranného mechanismu proti CSRF útokům. [**18**]

Druhou možností je dvojitá autentizace uživatele k autorizaci provedení důležitých operací, tento přístup je typický pro aplikace internetového bankovnictví, kdy uživatel dostane před odesláním transakce autorizační e-mail nebo SMS zprávu. Je však jasné, že tento striktní přístup lze aplikovat pouze na omezený počet operací tak, aby byla zachována použitelnost aplikace.

2.9 Použití známých zranitelností komponent

Tento typ útoku je velice variabilní a může mít různé cíle. Slabiny v podobě známých zranitelností jsou poměrně rozšířené a jejich odhalení nebývá snadné. [**19**]

Klasifikace útoku z hlediska náročnosti provedení útoku, rozšíření, schopnosti detekce a možného dopadu je uvedena v tabulce 9.

Náročnost provedení	**Rozšíření**	**Schopnost detekce chyby**	**Možný dopad na aplikaci**
střední	velké	nízká	střední

*Tabulka 9 – Klasifikace útoku [**1**]*

2.8.1 Princip útoku

Útočník se nesnaží přijít na slabé místo aplikace sám, ale využije zdrojů běžně dostupných na internetu nebo v komunitě hackerů. Ohrožení se týká především rozšířených a populárních aplikací. Jedna odhalená slabina může postihnout mnoho jednotlivých aplikací prakticky po celém světě.

Útočníci využívají špatné informovanosti koncových provozovatelů aplikací o nových verzích nebo bezpečnostních záplatách. Někteří provozovatelé také nejsou vždy ochotni přejít na novou verzi aplikace nebo to není možné provést z technických důvodů. Tato prodleva mezi hlášením chyby, její opravou a distribucí opravené verze je výhodou pro útočníky. [**19**]

Problém mohou také způsobit komponenty třetích stran, které jsou do aplikace přidány provozovatelem jednorázově a nejsou tak vázány na pravidelnou aktualizaci aplikace. Může jít o různé jednorázově vyvinuté moduly, o jejichž další vývoj se nikdo nestará. [**20**]

Známá slabina může mít mnoho podob, většinou zahrnuje některou z možností výše jmenovaných útoků.

2.8.2 Obrana

Obrana proti útokům tohoto typu může být velice složitá, zvláště pak u open-source aplikací. Známé chyby mohou být rozesety na mnoha místech internetu a jednotliví vývojáři o nich mohou ztratit přehled. Většina OSS aplikací nemá centralizovaný systém hlášení chyb a jejich řešení. Hlášení o chybách v bezpečnosti lze pak dohledat na různých fórech a nezávislých webech, které nemají s aplikací nic společného.

I přes tato omezení je většina známých chyb v další verzi aplikace již odstraněna. Udržování provozované aplikace v nejaktuálnější verzi patří k hlavním, mnohdy jediným, prvkům obrany.

Na straně vývoje je důležitá především dobrá informovanost o známých chybách, jejich evidování a včasné řešení. Důležitá je i komunikace s provozovateli aplikací, jejich včasné informování o možných hrozbách a zvolených opatřeních. [**19**]

2.10 Neošetřené přesměrování

Cílem útoku je přesměrování oběti na podvodnou stránku útočníka pomocí důvěryhodně vypadajícího odkazu aplikace. Podvodná stránka může z oběti vylákat přístupové údaje nebo jiné citlivé údaje, případně nainstalovat škodlivý software.

Klasifikace útoku z hlediska náročnosti provedení útoku, rozšíření, schopnosti detekce a možného dopadu je uvedena v tabulce 10.

Náročnost provedení	**Rozšíření**	**Schopnost detekce chyby**	**Možný dopad na aplikaci**
střední	nízké	vysoká	střední

*Tabulka 10 – Klasifikace útoku [**1**]*

2.8.1 Princip útoku

Základním předpokladem pro útok je, že napadená aplikace používá při přechodu na jiné weby funkci přesměrování. Odkazy, které směřují ven z aplikace, nejsou přímé, přechod na jinou stránku zajišťuje samostatná funkce. Odkaz může vypadat následovně:

```
<a
href="http://www.moje_stranka.cz/presmeruj.php?url=www.seznam.cz
">Vyhledávač seznam</a>
```

Funkce na stránce „presmeruj.php“ získá parametr z URL a uživatele přesměruje na cílovou adresu, většinou ještě dojde k zaznamenání této uživatelské akce, jinak by implementace přesměrování neměla význam.

Útočníkovi v tomto případě stačí nahradit parametr v URL tak, aby došlo k přesměrování na podvodnou stránku.

```
www.moje_stranka.cz/presmeruj.php?url=www.utocnik.cz
```

Pozměněný odkaz poté musí podstrčit oběti útoku, například v e-mailu. Odkaz, vede primárně na důvěryhodnou stránku aplikace, čímž dojde ke zmatení oběti. Funkce na stránce „presmeruj.php“ přesměruje oběť na podvodnou stránku.

Útok může být pochopitelně daleko rafinovanější a parametr v URL se může lišit jen minimálně nebo se podobat názvu zcela normální stránky. Další možností je umístit odkaz přímo na stránku, například v diskusi. Pak není nutné oběti odkaz posílat, stačí počkat, až na něj klikne. Výsledný odkaz v aplikaci může vypadat podobně jako v příkladu níže.

```
<a href="http://www.moje_stranka.cz/presmeruj.php?url=
www.moje_stranka.cz/presmeruj.php?url=www.utocnik.cz">Zajímavý
odkaz</a>
```

V případě kliknutí dojde nejdříve k přechodu na „presmeruj.php“ s parametrem url=www.utocnik.cz a poté na stránku útočníka.

2.8.2 Obrana

Velmi účinnou obrannou je vůbec nepoužívat funkce k přesměrování a odkazovat přímo:

```
<a href="http://www.seznam.cz ">Vyhledávač seznam</a>
```

V případech, kdy je potřeba využívat pro přesměrování speciální funkci, například pokud webová aplikace zjišťuje počet kliknutí na daný odkaz, je nutné, aby bylo odkazování ošetřeno. Ošetření spočívá v omezení URL adres, na které lze přesměrovat. K tomu může dobře posloužit porovnání požadované adresy s databází adres, na které stránka odkazuje a eviduje u nich počet přesměrování. Druhou možností je nepoužívat jako parametr přímo cílovou URL adresu, ale nějaký jiný identifikátor, podle kterého se cílová URL získá z databáze.

2.11 Shrnutí kapitoly

Jednotlivé útoky lze odděleně popsat, ale je důležité vidět je ve vzájemném kontextu. Zkušení útočníci jednotlivé typy útoků kombinují a zdokonalují útočné strategie. Znalost kódu aplikace a její masové rozšíření přispívají k vyšší

potenciální zranitelnosti open-source aplikací, kterými jsou i testované aplikace e-learningu v této práci.

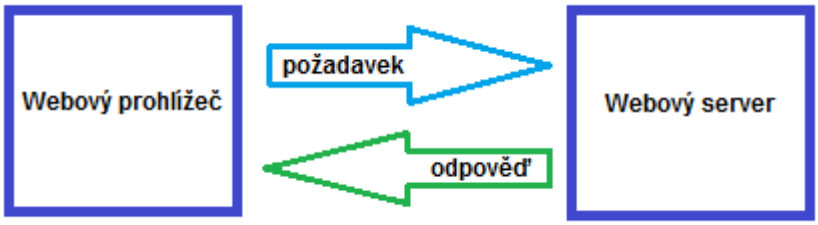
3 Ochrana webových aplikací

Kvalita ochrany webových aplikací je silně závislá na komunikačním protokolu, který zajišťuje její chod. To je důvod, proč se tato kapitola věnuje komunikačním protokolům a jejich bezpečnosti. Jedná se o nezabezpečený protokol HTTP a protokol pro šifrovanou komunikaci HTTPS. Pozornost je věnována zejména protokolu HTTPS a jeho přínosu pro bezpečnost webových aplikací.

Techniky sociálního inženýrství využité pro útoky na uživatele webových aplikací jsou často úspěšné a z technického hlediska prakticky neřešitelné. Z tohoto důvodu se druhá část této kapitoly věnuje sociálnímu pozadí bezpečnosti webových aplikací, osvětě a komunikaci s uživateli.

3.1 Protokol HTTP

Hypertext Transfer Protocol, česky protokol pro přenos hypertextu, je internetový protokol určený pro přenos libovolných datových objektů mezi webovým serverem a prohlížečem. Existují 3 hlavní verze protokolu HTTP, 0.9, 1.0 a 1.1. Nejnovější verze má dnes plnou podporu moderních prohlížečů i většiny serverů, základní funkce je však pro všechny verze stejná. Protokol HTTP je vystaven na modelu požadavek/odpověď.



Obrázek 1 – HTTP komunikace [AUTOR]

Model požadavek/odpověď lze popsat čtyřmi základními kroky, které představují princip komunikace pomocí HTTP:

- navázání spojení;
- zaslání požadavku klientem (prohlížečem);
- zaslání odpovědi serverem;
- uzavření spojení.

Jak je vidět, protokol HTTP je bezstavový, to znamená, že server neudržuje stálé spojení s klientem. Spojení je po načtení všech prvků stránky, tedy vyřízení všech požadavků klienta, uzavřeno. To přináší zásadní problém v jednoznačné identifikaci

klienta pro webové aplikace, které potřebují stavovou informaci pro své fungování. [**21**] [**22**]

Řešení tohoto problému existuje několik, nejpoužívanější je využití cookies v kombinaci se session proměnnými.

Cookie je malá textová informace, která je nabídnuta serverem k uložení v prohlížeči. Tato informace slouží k identifikaci klienta a posílá se při další komunikaci se serverem. Předávání stavových informací pomocí cookies je nebezpečné, protože se informace o stavu posílají v každém požadavku a v případě protokolu HTTP jako prostý text. [**23**]

Podstatné zvýšení bezpečnosti přináší kombinace session proměnných, uložených na straně serveru a cookies uložených v prohlížeči. V session jsou uloženy stavové informace a cookie obsahuje identifikátor těchto informací (session ID). Není tedy nutné posílat stavové informace při každém požadavku, ale pouze při jejich vytvoření nebo změně. Ke spárování uživatele a stavových informací uložených v session na serveru slouží právě identifikátor v cookie. Tento identifikátor je odesílán s každým HTTP požadavkem klienta. Bezpečnost spočívá v tom, že je posílána jen malá část informací a session ID. Session ID je navíc obtížně odhadnutelné, jedná se nejčastěji o náhodné číslo zašifrované jednosměrnou funkcí MD5 nebo SHA. [**21**]

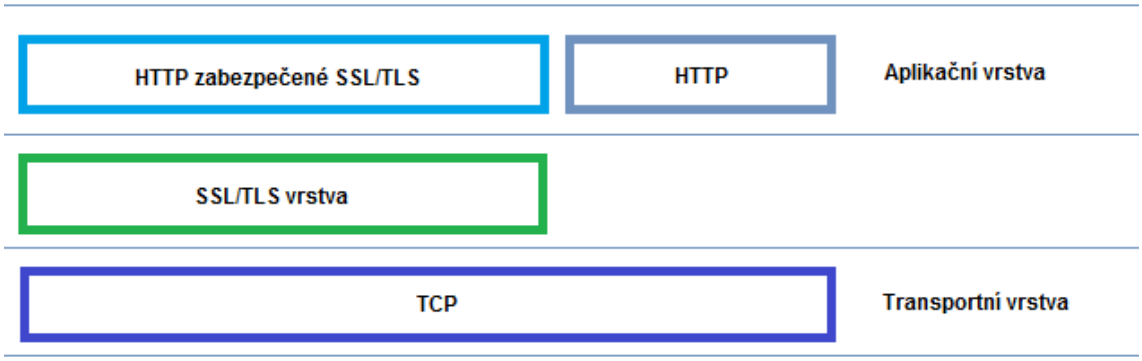
3.1.1 Nebezpečí protokolu HTTP

Protokol HTTP nikdy nebyl navržen jako zabezpečený a v době jeho vzniku (1991) nikdo nepočítal s jeho dnešním masovým využitím. Základní bezpečnostní slabinou HTTP je, že veškerá data mezi prohlížečem a serverem přenáší v textové podobě. Pokud se útočníkovi podaří tuto komunikaci zachytit, nejsou data nijak chráněna a hrozí jejich zneužití.

V dnešní době je na internetu mnoho aplikací, které pracují s citlivými osobními nebo firemními daty. Pro jejich účinnou obranu je nejen důležité mít bezpečně naprogramované aplikace, ale také tyto aplikace provozovat na bezpečném protokolu. Pro účely bezpečné komunikace mezi serverem a klientem vznikl standard HTTPS, o kterém pojednává následující kapitola.

3.2 Bezpečný protokol

Hypertext Transfer Protocol Secure (HTTPS), je bezpečnostní nadstavbou protokolu HTTP, která umožňuje šifrovat komunikaci mezi serverem a prohlížečem a také ověřit identitu protistrany. HTTPS využívá SSL nebo TSL, což jsou bezpečnostní vrstvy vložené mezi aplikační vrstvu a transportní vrstvu. U webových aplikací je aplikační vrstvou nezabezpečený protokol HTTP a transportní vrstvu představuje protokol TCP, umístění bezpečnostní vrstvy je vidět na obrázku 2.



Obrázek 2 – SSL/TLS vrstva [AUTOR]

Protokol HTTPS poskytuje dva základní prvky zabezpečení. Prvním je šifrovaná komunikace, která je zajištěna použitím asymetrické kryptografie. Druhým pak ověření identity a důvěryhodnosti protistrany v internetové komunikaci, toto ověření zajišťují digitální certifikáty a certifikační autority, kterým je věnována kapitola 3.2.3.

3.2.1 SSL

Security Socket Layer (SSL), česky vrstva zabezpečených soketů, je bezpečnostní vrstva vložená mezi TCP a HTTP, která zajišťuje šifrovanou komunikaci a umožňuje autentizaci komunikujících stran.

První verze SSL (SSL 1.0) byla zveřejněna společností Netscape Comunication roku 1993 a o několik měsíců později byla zveřejněna verze SSL 2.0. V dalších letech probíhal konkurenční boj mezi společnostmi Microsoft a Netscape Comunication, výsledkem byl na jedné straně bezpečný protokol PCT od Microsoftu a na straně druhé SSL 3.0. SSL poslední verze se stal oblíbeným bezpečným protokolem a postupem času se stal synonymem pro bezpečnost na internetu. [**24**]

3.2.1.1 Princip

Navázání SSL spojení (tzv. SSL handshake) je založeno na principu asymetrické kryptografie, kdy každá strana (klient a server) má dvojici klíčů - veřejný a soukromý. Veřejný klíč je dostupný protistraně a slouží k zašifrování předávané zprávy. Soukromý klíč je tajný a slouží k rozšifrování zprávy zašifrované příslušným veřejným klíčem. Je zajištěno, že šifrovaná zpráva může být rozšifrována pouze soukromým klíčem, takže při odposlechu komunikace je zachycená zpráva pro útočníka nerozluštitelná, respektive její rozluštění by trvalo velmi dlouho (závisí na síle použitého šifrování). Při navazování bezpečného spojení spolu komunikují server a klient, tato komunikace má ustálená pravidla:

- klient pošle serveru požadavek na SSL spojení a připojí doplňující informace o verzi SSL a dostupných kryptografických algoritmech;

- server pošle klientovi odpověď, která obsahuje certifikát serveru s veřejným klíčem serveru a doplňující informace;
- na základě certifikátu si klient ověří autentičnost serveru;
- klient vygeneruje základ pro šifrovací klíč, zašifruje ho pomocí veřejného klíče serveru a odešle;
- server pomocí svého soukromého klíče rozšifruje základ šifrovacího klíče, který mu zaslal klient;
- server i klient vygenerují ze základu šifrovací klíč;
- dojde ke vzájemnému potvrzení o zahájení bezpečné komunikace pomocí vytvořeného klíče;
- komunikace mezi serverem a klientem probíhá šifrovaně.

V první fázi navazování spojení jsou dohodnuty kryptografické algoritmy pro různé fáze komunikace, v SSL 3.0 je pro každou fázi komunikace na výběr z několika možností:

- pro výměnu klíčů: RSA, Diffie-Hellman, DSA nebo Fortezza;
- pro symetrické šifrování: RC2, RC4, IDEA, DES, 3DES nebo AES;
- pro jednosměrné šifrování: MD5 nebo SHA.

Poznámka: Z výše popsaného postupu navázání SSL spojení je patrné, že k navázání spojení stačí jen sada klíčů serveru. SLL podporuje oboustranné ověření, ale u webových aplikací není běžné. [**25**]

3.2.2 TLS

Transport Layer Security (TLS) je zabezpečený protokol, stejně jako SSL. TLS je přímým nástupcem SSL 3.0.

TLS verze 1.0 vznikl v roce 1996 přejmenováním protokolu SSL společností Internet Engineering Task Force, která se zabývá standardizací v prostředí internetu. Rozdíly mezi SSL 3.0 a TLS 1.0 jsou minimální. Poslední vydanou a standardizovanou verzí je TLS 1.2 z roku 2008. [**24**]

SSL i TLS poskytují stejnou službu, tedy zabezpečené spojení s autentizací protistrany. Oba protokoly používají řadu stejných metod (asymetrická kryptografie) i navazování spojení je podobné. TLS především odstraňuje některé drobné nedostatky svého předchůdce a je tedy logicky bezpečnější. Přehled nejzásadnějších odlišností je uveden v následujících bodech.

- Zasílání varování klientem: Pokud klient nemá certifikát, může v TLS odeslat zprávu „No certificate“, takže je server informován a nečeká na veřejný klíč klienta. V SSL tato možnost není.
- Ověřovací kód zpráv: TLS používá pro výpočet ověřovacího kódu zprávy standard HMAC[4], který umožní použití více hashovacích funkcí, zatímco SSL používá pouze MD5 a SHA.
- Vytvoření základu pro šifrovací klíč: TLS používá pro generování základu klíče standard HMAC a PRF (pseudonáhodná funkce, která rozdělí vstupní data na poloviny a každou z polovin šifruje jiným algoritmem). SSL používá omezený počet algoritmů (RSA, Diffie - Hellman nebo Fortezza). [**26**] [**27**]

3.2.3 Certifikáty

Digitální certifikát je digitálně podepsaný veřejný šifrovací klíč, který je odesílán serverem v odpovědi na požadavek klienta o navázání zabezpečeného spojení. Digitální certifikát je ve formátu X.509, který je specifikován standardem RFC 5280.

Digitální certifikát neobsahuje pouze veřejný šifrovací klíč, ale i informace o majiteli, vystaviteli certifikátu, sériové číslo, datum expirace a řadu dalších údajů. [**28**]

Jak již bylo zmíněno, digitální certifikát slouží především k ověření důvěryhodnosti protistrany. Důvěryhodnost protistrany závisí na pravdivosti a ověřitelnosti informací obsažených v certifikátu.

Při ověřování se využívá princip přenosu důvěry, kdy se za pravdivost údajů v certifikátu zaručí jeho vydavatel – certifikační autorita. Za vydavatele většinou ručí další (nadřízená) certifikační autorita, čímž vznikají globální řetězce přenosu důvěry. Pomyslným vrcholem je kořenová certifikační autorita. Samotné ověření provádí klient (prohlížeč), který rozhodne, zda je certifikát možné považovat za důvěryhodný (byl vydán důvěryhodnou certifikační autoritou). [**29**]

3.2.3.1 Certifikační autorita

Certifikační autorita je společnost, která vydává digitální certifikáty a přitom se zaručuje za pravdivost údajů uvedených v certifikátu.

Většina certifikačních autorit poskytuje několik druhů certifikátů, které se liší mírou důvěryhodnosti, výší pojistného plnění, účelu použití a především cenou. Cenový rozsah certifikátů je stejně široký jako jejich nabídka, lze sehnat i certifikát zcela zdarma. U placených certifikátů má zákazník většinou jistotu akceptace certifikátu nejrozšířenějšími prohlížeči.

[4] HMAC je typ autentizačního kódu zprávy vypočtený pomocí hashovacího algoritmu (specifikace HMAC je v dokumentu RFC 2104)

V České republice zákon č. 227/2000 Sb. (Zákon o elektronickém podpisu) definuje pojem „Kvalifikovaný certifikát“, který může vydat pouze státem akreditovaná certifikační autorita. U kvalifikovaného certifikátu má zákazník jistotu, že bude přijat ve všech členských zemích EU. [**29**]

4 E-learning

E-learning je ve své podstatě účelné využití informačních technologií v procesu vzdělávání. První aplikace pro výuku s pomocí počítačů začaly vznikat na konci minulého století a byly určeny hlavně pro akademickou sféru. Tyto aplikace nelze označit za e-learning, protože výuku pouze podporovaly, zatímco e-learning výuku plně zajišťuje a řídí. E-learning je systém pro vzdělávání.

Díky neustále rostoucímu významu znalostí se e-learning brzo prosadil i v podnikové sféře. Druhým faktorem prudkého rozvoje je globální rozšíření internetových a multimediálních technologií. Dnes je e-learning plně integrovaný do programu vnitrofiremního vzdělávání v řadě velkých i menších firem. [**30**] [**31**]

4.1 Přednosti e-learningu

Předností e-learningu lze nalézt mnoho. Ty základní představují zároveň dnešní požadavky firem na moderní systémy vzdělávání zaměstnanců:

- dostupnost,
- efektivita,
- úspora času a nákladů.

Dostupnost je základní výhodou e-learningu, zaměstnanec se může vzdělávat nejen v práci, ale i doma nebo na cestách. Dnešní e-learningové systémy jsou dostupné 24 hodin denně. Dostupnost s sebou přináší také výhodu v podobě volnosti plánování vzdělávání, které již není vázané na vzdělávací pracovníky a nenarušuje tak pracovní proces.

Efektivita vyplývá z možnosti přizpůsobení e-learningu konkrétnímu zaměstnanci, který si může sám volit tempo a parametry vzdělávání. Zaměstnanec se vzdělává jen v těch oblastech, které skutečně využije. To v klasickém skupinovém vzdělávání realizovat nelze.

Úspory času i nákladů je dosaženo především díky efektivnějšímu řízení vzdělávání a snížení výdajů na zajištění lektorů, školících prostorů, dopravu nebo ubytování zaměstnanců ve školících centrech. [**30**]

4.2 Postavení e-learningu v podniku

Moderní e-learningové systémy nemají v podniku jen vzdělávací funkci, v podobě nahrazení prezenční výuky a elektronického ověřování znalostí. Moderní LMS

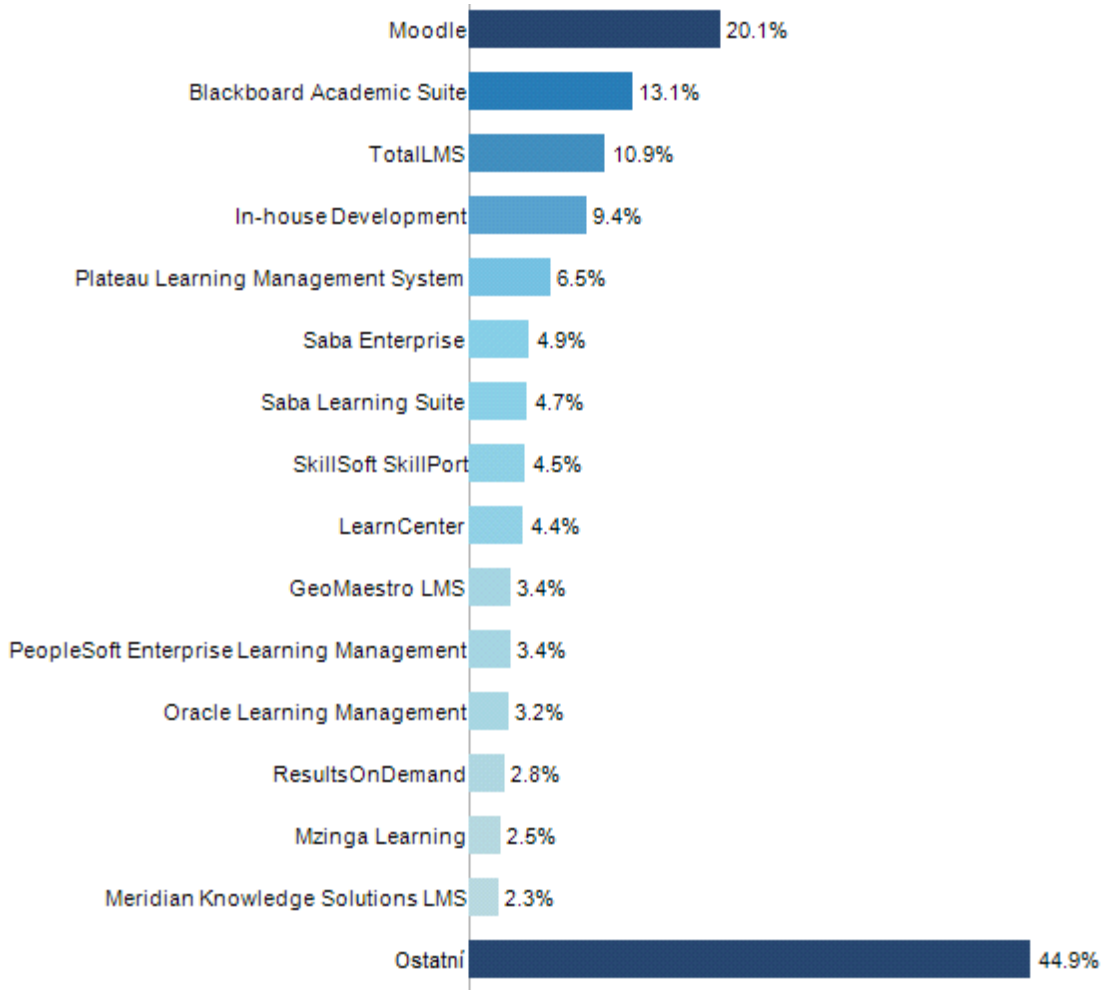
(learning management systém – systém pro řízení výuky) pokrývají v podniku řadu dalších oblastí.

LMS kontinuálně shromažďují znalosti podniku a udržují je v konzistentní a dostupné podobě, stávají se tak často znalostní bází podniku. K těmto znalostem se může uživatel kdykoli vracet a mít tak potřebné informace stále dostupné. LSM také shromažďují data o uživatelích, úrovni jejich znalostí a jejich celkovém začlenění do vzdělávacího procesu podniku. Tvorba reportů z těchto empirických dat umožňuje zefektivnění řízení procesu vzdělávání ze strany managementu.

Celosvětové propojení sítí internet dělá z e-learningu ideální nástroj pro integrované vzdělávání ve velkých nadnárodních společnostech. Jednotlivé národní pobočky mohou velmi rychle komunikovat znalosti s ostatními a udržovat stejnou úroveň vzdělanosti zaměstnanců napříč nadnárodním podnikem. V dnešní velké konkurenci a businessu založeném na využití informací je velmi žádoucí mít ty správné informace a znalosti právě včas. [**31**]

4.3 Open-source e-learning

Open-source software má v podnikové sféře většinou zastoupení hlavně u malých a středních podniků, které nemají dostatek finančních prostředků pro nákup drahých licencí. Tento celosvětový trend je vidět prakticky na všech podnikových IS. E-learning má ve světě open-source softwaru výjimečné postavení. Jedním ze světových leaderů v oblasti LMS je e-learnig Moodle, který má značné zastoupení u podniků všech velikostí. Podle výzkumu, provedeného společností The Elearning Guild v roce 2009, se Moodle řadil na první příčku světového žebříčku (viz Obrázek 3).



*Obrázek 3 – Světový žebříček LMS 2009 [**32**]*

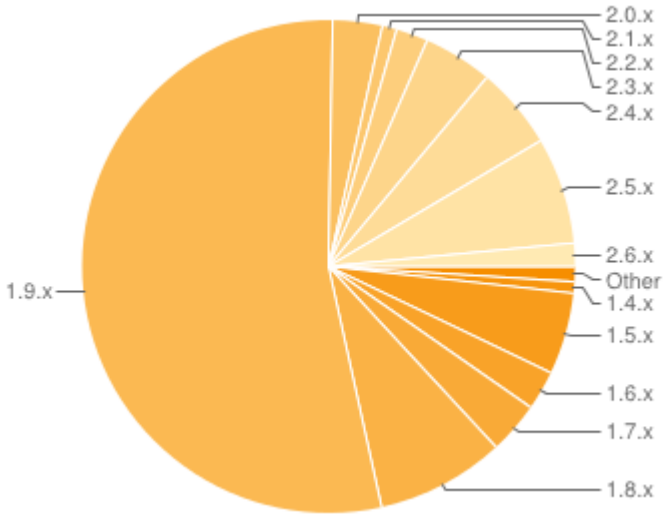
Dalším důvodem velkého rozšíření OSS LMS je fakt, že jde o velmi univerzální nástroj, který si najde své místo v každém podniku a zároveň nepřináší velké náklady na implementaci. Ostatní podnikové IS musí být často přizpůsobeny typu podniku; jiné nároky na IS má výrobní podnik a jiné pojišťovna nebo banka, zatímco e-learning ve své univerzální formě je vhodný pro oba. [**32**]

Zůstávají dvě otázky. Tou první je, zda OSS LMS dokáže obstát v konkurenci klasicky licencovaných LMS produktů z hlediska nabízených funkcionalit. Zde se dá konstatovat, že ano. Všechny testované LMS nabízejí srovnatelnou paletu funkcí jako placené produkty. Druhou otázkou je bezpečnost OSS LMS, kterou se prakticky zabývá tato práce v dalších kapitolách. Nejprve je nutné se seznámit s testovanými subjekty.

4.3.1 Moodle

Moodle patří mezi nejrozšířenější LSM systémy a je vyvíjen od roku 2001 komunitou programátorů v čele s vývojářem Martinem Dougiamasem. Verze Moodle 1.0 byla zveřejněna v roce 2002 a následoval rychlý rozvoj. Dnes je Moodle přeložen do více než 100 světových jazyků a je finančně podporován partnery

z celého světa. Poslední vydanou verzí je Moodle 2.6.1 z roku 2014. Nejrozšířenější verzí je však verze 1.9, jak je vidět na obrázku 4. [**33**] [**34**]



*Obrázek 4 – Podíl registrací Moodle dle verzí [**33**]*

LMS Moodle využívají i velké nadnárodní společnosti, v České republice je to například pojišťovna Kooperativa (člen skupiny VIG).

4.3.1.1 Funkcionalita a možnosti rozšíření

Základní funkcionalita poskytuje moderní prostředky pro kompletní pokrytí vzdělávací ho procesu v organizaci. Moodle umožňuje vytvářet, organizovat a sdílet multimediální učební materiály. Obsahuje nástroje pro ověřování znalostí uživatelů, správu studijních plánů a řadu komunikačních nástrojů.

Další prostor pro rozšíření představuje velké množství aplikačních modulů a rozšíření, jejichž instalace je rychlá a jednoduchá. [**34**]

4.3.1.2 Licence

Moodle LMS podléhá Obecné veřejné licenci GNU verze 3 (Moodle verzí 1.x podléhá licenci GNU verze 2), vydané Free Software Foundation. Program je tedy volně šiřitelný a modifikovatelný podle ustanovení licence. Na používání Moodle LMS nejsou poskytovány žádné záruky. [**35**]

4.3.2 eFront

Open-source LMS eFront je, stejně jako Moodle, velice populární. Ve své základní verzi je považován za méně robustní alternativu LMS Moodle. Systém eFront je vyvíjen od roku 2001 týmem řeckých programátorů a zpočátku byl projekt financován řeckou vládou. Dnes je vývoj financován partnery programu, podobně jako v případě LMS Moodle. LMS eFront je přeložen do 40 jazyků, včetně češtiny. Poslední vydanou verzí je eFront 3.6.14.4 z roku 2014.

LMS eFront využívá řada velkých nadnárodních společností se zastoupením v České republice, například Fujitsu nebo Panasonic. [**36**]

4.3.2.1 Funkcionalita a možnosti rozšíření

LMS eFront je nabízen ve 3 variantách: Open-source, Educational nebo Enterprise. První varianta je zcela zdarma, ostatní dvě varianty jsou placené podle počtu uživatelů. Jednotlivé varianty se liší svými funkcemi, ale také úrovní zabezpečení. Open-source varianta, kterou se zabývá tato práce, neobsahuje oproti placeným variantám profesionální uživatelskou podporu, některé komunikační nástroje, možnost propojení na sociální sítě a některé funkce reportingu a administrace. I přes tato omezení poskytuje Open-source varianta konzistentní LMS systém, který může být doplněn pestrou paletou rozšiřujících modulů, které jsou ve většině případů zdarma.

Velkou předností LMS eFront je, že podporuje standardy SCORM 1.2, případně SCORM 2004 u placených verzí systému. SCORM je soubor principů a standardů pro tvorbu obsahu v LMS, který umožňuje tento obsah používat i v jiných LMS, které pravidlům SCORM vyhovují. Studijní materiály mohou být využívány v různých LMS s podporou SCORM, což představuje nespornou výhodu při přechodu z jednoho LMS na jiné. [**36**] [**37**]

4.3.2.2 Licence

Varianta Open-source podléhá licenci Common Public Attribution License 1.0 (CPAL) vydané Open-source Initiative v roce 2007. Program je volně šiřitelný a modifikovatelný podle ustanovení licence. Na používání eFront LMS ve variantě Open-source nejsou poskytovány žádné záruky.

4.3.3 Claroline

Claroline LMS patří k menším hráčům na trhu LMS, v České republice není příliš známý, především proto, že není plně dokončen překlad do češtiny. V mnoha zemích je však poměrně významným konkurentem ostatních open-source LMS. Claroline je oceňován především pro svou jednoduchost a snadnou správu.

Vývoj Claroline započal v roce 2000 na Catholic University of Louvain v Belgii, první verze byla publikována v roce 2001. Vývoj je financován partnery projektu a z dobrovolných darů. Claroline byl postupně přeložen do 35 jazyků a na dalších překladech se pracuje. LMS Claroline je rozšířen do 117 zemí, včetně České republiky, a provozován ve více jak 2 500 registrovaných organizacích. Poslední vydanou verzí je Claroline 1.11.9 z roku 2013. [**38**] [**39**]

4.3.1.1 Funkcionalita a možnosti rozšíření

Claroline poskytuje možnosti pro tvorbu studijních materiálů, kurzů, testů a úkolů. Obsahuje nástroje pro organizaci skupinové práce, reporting a řízení procesu

vzdělávání. Claroline je i komunikační nástroj, který nabízí nástroje pro organizaci diskusního fóra. Claroline podporuje standardy SCORM.

Claroline nenabízí příliš mnoho rozšíření základní verze. Je to dáno politikou komunity, která chce Claroline vyvíjet tak, aby byla zachována jednoduchost a přehlednost pro uživatele. Nové prvky nejsou nabízeny jako moduly, ale jsou začleněny do nové verze. [**40**]

4.3.1.2 Licence

Claroline LMS podléhá Obecné veřejné licenci GNU verze 3, vydané Free Software Foundation. Program je tedy volně šiřitelný a modifikovatelný podle ustanovení licence. Na používání Claroline LMS nejsou poskytovány žádné záruky. [**35**]

4.4 Bezpečnostní rizika e-learningu

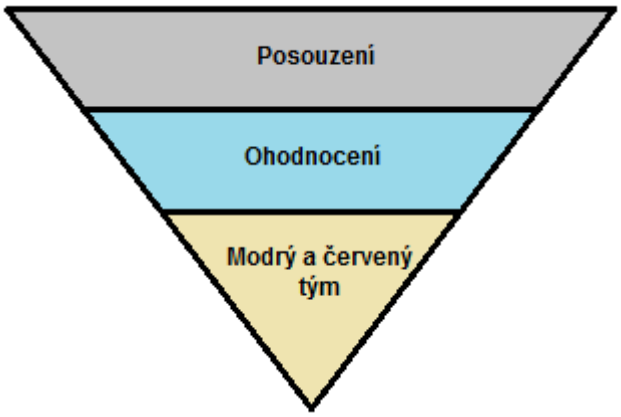
LMS organizace často obsahuje interní znalosti a informace významné pro činnost podniku, firemní know-how. Tyto informace jsou obsaženy v různých IS podniku, ale LMS patří k systémům, kde se tyto informace více koncentrují. Informace mají svou cenu a jsou hybnou silou obchodu po celém světě, informace lze zcizit a zpeněžit.

LMS organizace také shromažďuje osobní data uživatelů, jejichž únik je bezpečnostním rizikem. Ochraně osobních dat v podnikových aplikacích by měla každá organizace věnovat mimořádnou pozornost a LMS není v tomto případě žádnou výjimkou. U LMS je riziko potenciálně vyšší než u interních podnikových aplikací, důvodem je požadavek na dostupnost e-learningu i mimo interní podnikovou síť.

Dalším rizikem je únik přihlašovacích údajů. Získání kontroly nad uživatelským účtem v aplikaci e-learningu může představovat bezpečnostní riziko i pro další podnikové aplikace. Důvod je poměrně prostý, řada uživatelů má stejné heslo do všech podnikových aplikací. Míra ohrožení bude v tomto případě záviset na oprávněních daného uživatele v ostatních aplikacích, ale ani toto riziko nelze podceňovat. [**41**]

5 Testování bezpečnosti informací

Společně s růstem rozšíření informačních a komunikačních technologií rostou i nároky na jejich spolehlivé a bezpečné fungování. Oblast řízení podnikové bezpečnosti se stala velmi významnou a vyžaduje vhodné techniky a nástroje. Jednou z uznávaných metodik je Information Assurance Methodology (IAM) sestavená americkou Národní bezpečnostní asociací (NSA). IAM rozděluje posuzování bezpečnosti informací do třech úrovní (viz obrázek 5).



*Obrázek 5 – Úrovně hodnocení bezpečnosti informací [**42**]*

Úroveň I – Posouzení (Assessment) zahrnuje posouzení bezpečnosti informací z vnějšího pohledu na procesy a postupy v řízení bezpečnosti. Cílem je posoudit, jak dobře organizace identifikuje možná rizika, navrhuje řešení a prosazuje je do praxe.

Úroveň II – Ohodnocení (Evaluation) je hlubším zkoumáním bezpečnosti a je spojeno s ověřováním skutečně aplikovaných bezpečnostních opatření v ICT podniku. K ověřování jsou nejčastěji použity automatizované testovací nástroje.

Úroveň III – Modrý a červený tým (Blue&Red team) je nejhlubší úrovní posuzování bezpečnosti a zahrnuje reálné pokusy o útok na informační systémy organizace - penetrační testy. Ověřování provádějí zpravidla specialisté – bezpečnostní testeři.

Pro provádění bezpečnostních testů třetí úrovně existuje řada metodik, úzce i široce zaměřených. Jednou z uznávaných a dostupných pro testování webových aplikací je metodika popsaná v OWASP Testing Guide, která je využita pro účely této práce. [**42**]

5.1 OWASP Testing Guide v. 3

OWASP Testing Guide v. 3 je příručka vydaná komunitou OWASP v roce 2008, která popisuje metodiku pro penetrační testování webových aplikací. Příručka poskytuje návod pro provedení celkem 66 kontrol, rozdělených do 9 kategorií.

- Testování konfigurace
- Testování business logiky
- Testování autentizace
- Testování autorizace
- Testování session managementu
- Testování validace dat

- Testování hrozeb typu DoS
- Testování webových služeb
- Testování hrozeb spojených s Ajax

[**43**]

5.2 Penetrační testování

Penetrační testování slouží k ověření bezpečnosti aplikace simulací reálného útoku. Cílem je odhalit bezpečnostní slabiny, zdokumentovat je a předat k opravě. OWASP rozděluje penetrační test do dvou částí, pasivní a aktivní.

Pasivní část testování zahrnuje seznámení testera s aplikací, kdy si tester s aplikací hraje a snaží se náhodně přijít na možná slabá místa. Po skončení této fáze by měl tester rozumět všem hlavním vstupním bodům aplikace a měl by umět aplikaci ovládat.

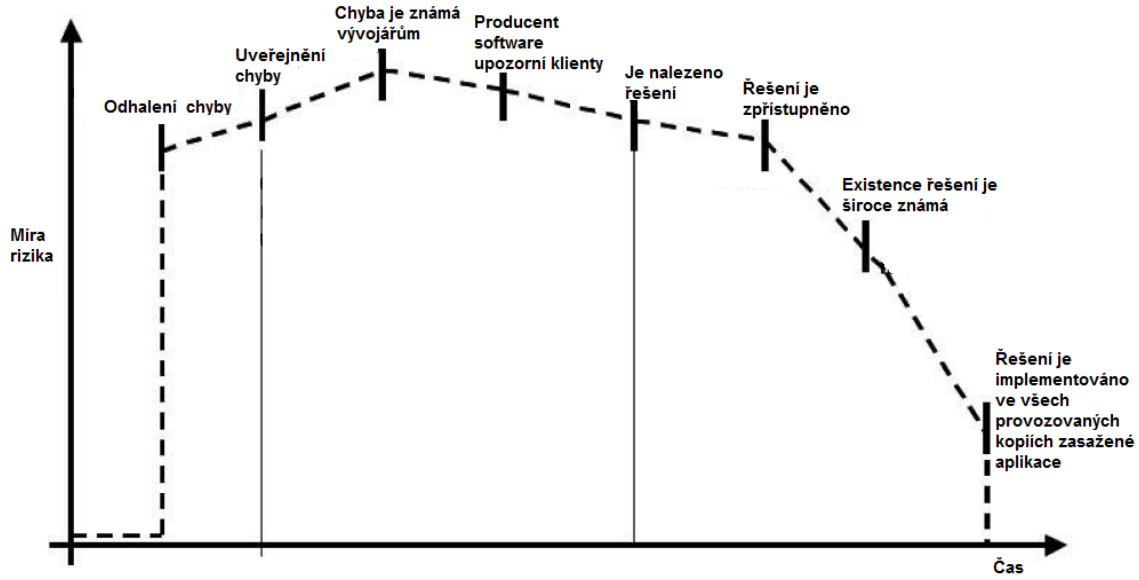
V aktivní části tester využije metodiku popsanou v příručce a otestuje aplikaci. K provedení aktivní části využívá tester nástroje pro penetrační testování, jakým je například WebScarab, taktéž vyvinutý komunitou OWASP.

Pro užší zaměření testů lze využít aktuální žebříček OWASP TOP 10, databázi známých slabin aktuálních i předchozích verzí nebo reporty o chybách testované aplikace. [**43**] [**44**]

5.3 Využití příručky OWASP Testing Guide

Příručka OWASP není určena jen testerům a bezpečnostním pracovníkům, je určena i vývojářům a návrhářům webových aplikací, kterým slouží pro lepší pochopení bezpečnostních hrozeb. Cílem je předcházet bezpečnostním hrozbám již při vývoji aplikace, vzniklé chyby efektivně odhalit a řešit.

Jak je vidět na obrázku 6, životní cyklus chyby nekončí jejím opravením, ale až nasazením opravy do všech provozovaných kopií aplikace.



*Obrázek 6 – Životní cyklus bezpečnostní chyby [**43**]*

Příručka doporučuje několik zásad, které mají vést ke zkrácení životního cyklu chyb a předcházení jejich vzniku:

- integrovat otázku bezpečnosti do každé vývojové fáze projektu;
- testovat bezpečnost v průběhu vývoje;
- pochopit bezpečnostní požadavky dané aplikace;
- mít k dispozici adekvátní bezpečnostní testy a testery schopné myslet jako útočník;
- vytvářet a udržovat dokumentaci aplikace;
- disponovat vhodnými testovacími nástroji;
- vyvíjet aplikace podle metrik;
- dokumentovat výsledky bezpečnostních testů.

Je zřejmé, že příručka OWASP najde uplatnění jak při penetračním testování, tak při návrhu a vývoji aplikace. Může být dobrým pomocníkem pro bezpečnostní pracovníky, vývojáře, analytiky a testery. Příručka nedává odpovědi jen na otázku jak a co testovat, ale i proč, kdy a kde testovat webové aplikace. [**43**]

5.4 Postup při testování vybraných LMS

Cílem této kapitoly je definovat konzistentní postup využitý pro praktické testování bezpečnosti vybraných open-source LMS. Postup je rozdělen do několika samostatných fází.

1) Seznámení testera s aplikací, aplikace principů OWASP pro pasivní část penetračního testování.
2) Identifikace potenciálně slabých míst na základě předchozí fáze a známých chyb z předchozích verzí webové aplikace.
3) Provedení aktivní části penetračního testování s využitím ověřených postupů popsaných v příručce OWASP se zaměřením na majoritní hrozby popsané v kapitole 2.
4) Vytvoření přehledu nalezených chyb s doplňujícím komentářem a hodnocením jejich závažnosti.
5) Vytvoření bezpečnostního doporučení pro provozovatele daného LMS.

6 Výsledky testování bezpečnosti vybraných systémů

Za účelem testování bezpečnosti vybraných aplikací e-learningu vzniklo jednoduché testovací prostředí, hostované na běžném webovém serveru. Provozovatel byl řádně upozorněn na záměr provedení penetračních testů a s tímto záměrem souhlasil. Doména byla po celou dobu testování veřejně nepřístupná a po skončení testů byl veškerý obsah odstraněn. Základní přehled parametrů je v tabulce 11.

Parametr	**Hodnota**
Server	Apache, OS Linux 2.6.32-279.5.2.el6.x86_64
Verze PHP	5.4.21
Verze MySQL Client API	5.5.27
HTTPS	Ano, OpenSSL 1.0.0-fips
Doména	www.mencik.cz
Hosting	www.endora.cz

Tabulka 11 – Základní údaje testovacího prostředí [AUTOR]

6.1 LMS Moodle

Testování LMS Moodle bylo provedeno na verzi Moodle 2.6.1, vydané 14. ledna 2014, a to v její výchozí konfiguraci. Jednotlivé fáze testování jsou popsány v následujících podkapitolách.

6.1.1 Identifikace vstupů a potenciálně slabých míst

Přehled základních vstupů a potenciálně slabých míst je uveden v tabulce 12. K identifikaci těchto míst byla využita technika pretest (seznámení testera

s aplikací), dokumentace k aplikaci, kód aplikace, databáze dříve hlášených chyb a nástroj WebScarab.

Vstup, operace nebo funkce	**Potenciální hrozba**
Přihlašování do aplikace	SQL injection, prolomení ověřování identity, chybný session management
Registrace uživatele	XSS
Vyhledávání	SQL injection
Parametry předávané v URL	SQL injection, CSFR
Zasílání zpráv v aplikaci	XSS, CSFR
Vstupní pole ostatních formulářů	XSS, CSFR

Tabulka 12 – Identifikace vstupů a potenciálně slabých míst [AUTOR]

Poznámka: Cílem této fáze není odhalit existující chyby, ale vybrat prvky aplikace, které mají takovou funkčnost, že představují bezpečnostní riziko. Na tyto prvky se tester zaměří při provádění penetračních testů.

6.1.2 Výsledek penetračního testování LMS Moodle

V tabulce 13 je uveden přehled provedených simulací (útoků) na webovou aplikaci LMS Moodle. V kapitole 6.1.2.1 jsou jednotlivé útoky a jejich výsledky podrobněji popsány.

Číslo	**Typ útoku/testu**	**Oblast aplikace**	**Výsledek**	**Závažnost**
1	Chybný session management, fixace session	Přihlášení do aplikace, odhlášení z aplikace	Negativní	
2	Prolomení autentizace, SQL injection	Přihlášení do aplikace	Negativní	
3	CSRF, metody GET a POST	Administrace – editace účtu, kurzu a nastavení administrace	Negativní	
4	Krádež identifikace session odposlechem komunikace, HTTPS vypnuté	Celá aplikace	Pozitivní	Nízká

Číslo	Typ útoku/testu	Oblast aplikace	Výsledek	Závažnost
5	XSS	Zasílání zpráv v aplikaci, příspěvky v diskusi ke kurzu, editace štítků	Negativní	
6	XSS	Založení/editace kurzu	Pozitivní	Nízká
7	XSS	Vytvoření testových úloh	Pozitivní	Střední
8	Neoprávněný přístup k objektu aplikace	Objekty nepřístupné pro běžného uživatele	Negativní	
9	Chyba v konfiguraci	Instalace aplikace	Negativní	
10	Expozice citlivých dat	Šifrování hesla	Negativní	
11	SQL injection	Vyhledávací formuláře	Negativní	
12	SQL injection	Parametry předávané v URL	Negativní	
13	XSS	Registrace uživatele	Negativní	

Tabulka 13 – Výsledky penetračního testování LMS Moodle 2.6.1 [AUTOR]

6.1.2.1 Komentáře

1. Session management aplikace Moodle nepředstavuje bezpečnostní hrozbu. Při přihlášení dojde k vytvoření zcela nové session. Při odhlášení z aplikace je session smazána a opět nahrazena novou.

2. Přihlašovací formulář je odolný proti SQL injection.

3. Aplikace používá bezpečnostní token pro zabrání CSRF útoků. Token se generuje při každém přihlášení do aplikace. Bezpečnostní token je předáván jako parametr v URL (zvýrazněn tučně).

```
…/bctest/moodle/course/switchrole.php?id=2&sesskey=zTPGIlxnZB
```

V případě formulářů je předáván jako skryté pole.

```
<input name="sesskey" type="hidden" value="zTPGIlxnZB" />
```

4. Identifikace session se zapisuje do cookie s názvem MoodleSession. Identifikace je přenášena protokolem HTTP nešifrovaně.

```
Set-Cookie: MoodleSession=qt953t461nm2ok6po2u032o3u4;
path=/bctest/moodle/
```

Nejedná se o chybu aplikace, ale o vlastnost protokolu, při použití HTTPS nelze odposlechnutou identifikaci použít.

5. Aplikace ošetřuje zprávu proti použití XSS před uložením do databáze.

6. Pole „Popis kurzu“ není ošetřeno proti XSS, lze vložit skript. Závažnost chyby je nízká, protože se chyba vyskytuje pouze v administraci, kde se útok tímto způsobem nepředpokládá.

7. Pole „Text úlohy“, „Obecná reakce“ a „Text odpovědi“ na stránce „moodle/question/question.php“ nejsou ošetřena proti útoku XSS. Přístup k této funkci má ve výchozím nastavení většina rolí (učitel, manažer, tvůrce kurzu a administrátor).

8. Aplikace pracuje s přímými referencemi na objekty (v URL). Test byl proveden na významných objektech aplikace (administrace uživatelů, importované soubory, editace kurzů apod.) a nebyla odhalena žádná chyba. Nahrané soubory jsou ukládány do speciálního adresáře, který není přímo přístupný z webu, například zadáním URL adresy. Toto opatření zajišťuje efektivní ochranu nahraných souborů.

9. Aplikaci nelze instalovat na server, který nemá potřebnou konfiguraci. Kontrola parametrů serveru a databáze proběhne v jednom z kroků instalace.

10. Pro šifrování hesla je použita kombinace několika kryptografických funkcí s iteracemi a kryptografickou solí. Použitá opatření jsou dostatečná a brání možnosti porovnání s databází otisků.

11. Vyhledávací formuláře jsou chráněné proti SQL injection.

12. Parametry získané z URL procházejí validací a jsou chráněné proti SQL injection.

13. Formulář pro registraci uživatele je chráněn proti XSS útokům.

6.1.2.2 Ostatní bezpečnostní nedostatky

LMS Moodle obsahuje funkci pro zapamatování uživatele. Tato, na první pohled užitečná funkce, představuje určité bezpečnostní riziko. Pokud je tato funkce používána na veřejném počítači, tak může útočník získat přístup k účtu oběti.

Druhým nedostatkem je zbytečně podrobné informování uživatelů o chybách aplikace. Chyby často vracejí názvy tabulek a funkcí, což může útočníkovi usnadnit orientaci v aplikaci. V následujícím příkladu je změněn parametr v URL na neexistující hodnotu.

```
…/moodle/mod/forum/discuss.php?d=test
```

Zobrazená chybová hláška prozrazuje název tabulky.

```
V databázové tabulce forum_discussions nemohu najít žádný
záznam.
```

Posledním identifikovaným nedostatkem je chybějící upozornění na nutnost odstranění instalačního souboru po skončení instalace.

6.1.3 Zhodnocení bezpečnosti a doporučení

LMS Moodle 2.6.1 je celkově dobře zabezpečená aplikace. Je dobrým příkladem toho, jak se vývojáři dokázali poučit z předchozích chyb a efektivně je řešit. Každý týden vychází update pro poslední vydanou verzi aplikace, který pokrývá nově zjištěné chyby i bezpečnostní nedostatky. Instalace updatu je velice jednoduchá a provádí se přímo v administraci aplikace.

Problémem pro bezpečnost je poměrně velké rozšíření starých verzí aplikace, zejména pak verze 1.9, kterou používá nadpoloviční většina registrovaných provozovatelů. [**33**] Podpora této verze již skončila a Moodle již neopravuje ani bezpečnostní chyby. [**45**]

Pro bezpečný provoz LMS Moodle lze doporučit provozování poslední dostupné verze a pravidelnou aktualizaci. Pro celkové zvýšení bezpečnosti a důvěryhodnosti je dobré používat HTTPS, a to alespoň pro přihlašování uživatelů a administraci webu.

6.2 LMS eFront

Testování LMS eFront bylo provedeno na verzi eFront 3.6.14.4 (build 18016), vydané 5. února 2014, a to v její výchozí konfiguraci. Jednotlivé fáze testování jsou popsány v následujících podkapitolách.

6.2.1 Identifikace vstupů a potenciálně slabých míst

Přehled základních vstupů a potenciálně slabých míst je uveden v tabulce 14. K identifikaci těchto míst byla využita technika pretest, dokumentace k aplikaci, kód aplikace, databáze dříve hlášených chyb a nástroj WebScarab.

Vstup, operace nebo funkce	**Potenciální hrozba**
Přihlašování do aplikace	SQL injection, prolomení ověřování identity, chybný session management
Fórum	XSS
Vyhledávání	SQL injection
Parametry předávané v URL	SQL injection, CSFR
Zasílání zpráv v aplikaci	XSS, CSFR

Vstup, operace nebo funkce	Potenciální hrozba
Správa účtu	XSS, CSFR, SQL injection
Mapování účtů	SQL injection
Exporty z aplikace	SQL injection

Tabulka 14 – Identifikace vstupů a potenciálně slabých míst [AUTOR]

6.2.2 Výsledek penetračního testování LMS eFront

V tabulce 15 je uveden přehled provedených simulací (útoků) na webovou aplikaci LMS eFront. V kapitole 6.2.2.1 jsou jednotlivé útoky a jejich výsledky podrobněji popsány.

Číslo	Typ útoku/testu	Oblast aplikace	Výsledek	Závažnost
1	Chybný session management, fixace session	Přihlášení do aplikace, odhlášení z aplikace	Negativní	
2	Prolomení autentizace, SQL injection	Přihlášení do aplikace	Negativní	
3	CSRF, metody GET a POST	Administrace – editace účtu, kurzu a nastavení administrace	Negativní	
4	Krádež identifikace session odposlechem komunikace, HTTPS vypnuté	Celá aplikace	Pozitivní	Nízká
5	XSS	Uživatelský účet – správa účtu uživatelem	Pozitivní	Vysoká
6	XSS	Zasílání zpráv v aplikaci	Pozitivní	Střední
7	XSS	Tvorba kurzů - formulář	Pozitivní	Nízká
8	XSS	Tvorba kategorií - formulář	Pozitivní	Nízká
9	XSS	Fórum – přidání příspěvku	Negativní	

Číslo	Typ útoku/testu	Oblast aplikace	Výsledek	Závažnost
10	Neoprávněný přístup k objektu aplikace	Objekty nepřístupné pro běžného uživatele	Negativní	
11	Chyba v konfiguraci	Instalace aplikace	Negativní	
12	Expozice citlivých dat	Šifrování hesel	Negativní	
13	SQL injection	Vyhledávací formuláře	Negativní	
14	SQL injection	Parametry předávané v URL	Negativní	
15	SQL injection	Mapování účtů	Negativní	
16	SQL injection	Exporty z aplikace	Negativní	

Tabulka 15 – Výsledky penetračního testování LMS eFront 3.6.14.4 [AUTOR]

6.2.2.1 Komentáře

1. Session management aplikace eFront nepředstavuje bezpečnostní hrozbu. Při přihlášení dojde k vytvoření zcela nové session. Při odhlášení z aplikace je session smazána a opět nahrazena novou.

2. Přihlašovací formulář je odolný proti SQL injection.

3. Aplikace používá pro ověření akcí uživatele bezpečnostní token, který je ve většině případů předávaný ve skrytém formulářovém poli.

```
<input name="qfS_csrf" type="hidden"
value="b226212b7f2a6b76d7fc63706043f1a3" />
```

4. Identifikace session se zapisuje do cookie PHPSESSID. Identifikace je přenášena protokolem HTTP nešifrovaně.

```
Set-Cookie: PHPSESSID=1uss36949jm8obgkhj5f67o6j1; path=/
```

Nejedná se o chybu aplikace, ale o vlastnost protokolu, při použití HTTPS nelze odposlechnutou identifikaci použít.

5. Formulářové pole „Příjmení“ na stránce „…/efront/www/student.php?ctg=personal“ není ošetřeno proti útoku XSS. Jedná se o velmi vážnou chybu, protože formulář pro editaci uživatelského účtu má přístupný každý uživatel. Problém vzniká již při vložení samotného tagu <script>, kdy dojde ke skrytí daného uživatele v přehledu uživatelů a administrátor nemůže provádět základní operace s uživateli.

6. Formulářové pole „Tělo“ pro zadání textu zprávy na stránce „…/efront/www/#role#.php?ctg=messages“ není ošetřeno proti útoku XSS. Chyba

se vyskytuje pouze u rolí s vyššími právy (profesor, administrátor), nikoli u běžného uživatele s rolí student.

7. Formulářové pole „Název kurzu“ na stránce „…/efront/www/#role#.php?ctg=courses&add_course=1“ není dostatečně ošetřeno proti útoku XSS. Aplikace nekontroluje použití znaků < a >, které umožňují vložit skript. Chyba se vyskytuje pouze u rolí s vyššími právy (profesor, administrátor).

8. Formulářové pole „Název kategorie“ na stránce „…/efront/www/#role#.php?ctg=directions&add_direction=1“ není dostatečně ošetřeno proti útoku XSS. Chyba je stejná, jako v případě testu číslo 7.

9. Vkládání příspěvků do fóra je velmi dobře ošetřeno proti útoku XSS, ačkoli je uživatelům umožněno editovat text příspěvku i pomocí HTML tagů.

10. Aplikace eFront příliš nevyužívá přímé reference. Test byl proveden na významných objektech aplikace, kde se přímé reference vyskytují, a nebyla odhalena žádná chyba. Importované soubory jsou ukládány do složky „…/efront/uploads/“, která je přímo dostupná z webu. Složka je chráněna souborem „.htaccess“ a přístup k uloženým souborům řídí aplikace. Toto řešení lze považovat za standardní.

11. Aplikaci nelze instalovat na server, který nemá potřebnou konfiguraci. Kontrola parametrů serveru a databáze proběhne v jednom z kroků instalace. Správnou konfiguraci serveru si může administrátor kdykoli ověřit v administraci, aplikace navíc sama informuje administrátora, pokud nějakou změnu v konfiguraci serveru zaznamená.

12. Hesla jsou šifrována funkcí MD5 s použitím kryptografické soli.

13. Vyhledávací formuláře jsou chráněné proti SQL injection.

14. Parametry v URL jsou chráněné proti SQL injection.

15. Mapování účtů je chráněné proti SQL injection.

16. Exporty z aplikace jsou chráněné proti SQL injection.

6.2.2.2 Ostatní bezpečnostní nedostatky

LMS eFront nekontroluje sílu hesla administrátora, ani jeho délku. Uživatelské heslo musí být dlouhé alespoň 6 znaků, ale taktéž není kontrolována jeho síla. Aplikace tedy nijak nenutí uživatele, aby používali dostatečně silná hesla, která budou odolnější proti slovníkovému útoku.

6.2.3 Zhodnocení bezpečnosti a doporučení

Aplikace LMS eFront 3.6.14.4 je dobře zabezpečená proti většině útoků, jejichž simulace byly předmětem penetračního testování. Z výsledků testování vyplývá, že aplikace eFront je nejméně zabezpečená proti XSS útokům.

Pro potlačení možnosti vzniku bezpečnostní chyby, která ohrožuje chod aplikace (test číslo 5), doporučuji dočasně zakázat volnou registraci nových uživatelů a uživatele přidávat ručně, po prověření administrátorem. Tím lze předejít nekontrolované registraci neznámých uživatelů, kteří mohou chtít na aplikaci zaútočit. Volnou registraci uživatelů lze povolit, až po opravě této bezpečnostní slabiny.

Stejně jako v případě LMS Moodle lze doporučit provozování aplikace eFront na zabezpečeném protokolu HTTPS, pravidelné aktualizování a sledování hlášení o bezpečnostních chybách.

6.3 LMS Claroline

Testování LMS Claroline bylo provedeno na verzi Claroline 1.11.9 vydané 26. listopadu 2013, a to v její výchozí konfiguraci. Jednotlivé fáze testování jsou popsány v následujících podkapitolách.

6.3.1 Identifikace vstupů a potenciálně slabých míst

Přehled základních vstupů a potenciálně slabých míst je uveden v tabulce 16. K identifikaci těchto míst byla využita technika pretest, dokumentace k aplikaci, kód aplikace, databáze dříve hlášených chyb a nástroj WebScarab.

Vstup, operace nebo funkce	**Potenciální hrozba**
Přihlašování do aplikace	SQL injection, prolomení ověřování identity, chybný session management
Registrace uživatele	XSS
Vyhledávání	SQL injection
Parametry předávané v URL	SQL injection, CSFR
Zasílání zpráv v aplikaci	XSS, CSFR
Správa účtu	XSS, CSFR, SQL injection
Konfigurace platformy	CSFR
Mapování účtů	Neoprávněný přístup k objektu aplikace
Fórum	XSS

Tabulka 16 – Identifikace vstupů a potenciálně slabých míst [AUTOR]

6.3.2 Výsledek penetračního testování LMS Claroline

V tabulce 17 je uveden přehled provedených simulací (útoků) na webovou aplikaci LMS Claroline. V kapitole 6.3.2.1 jsou jednotlivé útoky a jejich výsledky podrobněji popsány.

Číslo	Typ útoku/testu	Oblast aplikace	Výsledek	Závažnost
1	Chybný session management, fixace session	Přihlášení do aplikace, odhlášení z aplikace	Pozitivní	Střední
2	Prolomení autentizace, SQL injection	Přihlášení do aplikace	Negativní	
3	CSRF, metody GET a POST	Administrace aplikace, editace účtu	Pozitivní	Střední
4	Krádež identifikace session odposlechem komunikace, HTTPS vypnuté	Celá aplikace	Pozitivní	Nízká
5	XSS	Registrace uživatele	Negativní	
6	XSS	Zasílání zpráv, fórum	Negativní	
7	XSS	Vstupní pole ostatních formulářů	Negativní	
8	Neoprávněný přístup k objektu aplikace	Správa účtu uživatele – přidělování rolí	Pozitivní	Vysoká
9	Neoprávněný přístup k objektu aplikace	Dokumenty přístupné pouze po přihlášení	Pozitivní	Střední
10	Chyba v konfiguraci	Instalace aplikace	Negativní	
11	Expozice citlivých dat	Šifrování hesel	Pozitivní	Střední
12	SQL injection	Vyhledávací formuláře	Negativní	
13	SQL injection	Parametry předávané v URL	Negativní	

Číslo	Typ útoku/testu	Oblast aplikace	Výsledek	Závažnost
14	Neoprávněný přístup k objektu aplikace	Mapování účtů	Negativní	
15	Neoprávněný přístup k objektu aplikace	Obnovení zapomenutého hesla	Pozitivní	Střední

Tabulka 17 – Výsledky penetračního testování LMS Claroline 1.11.9 [AUTOR]

6.3.2.1 Komentáře

1. Session management aplikace LMS Claroline není dobře nastaven. Před přihlášením uživatele je vytvořena identifikace session, která se po přihlášení nezmění. Ke změně identifikace nedojde ani po odhlášení uživatele. Toto nastavení zvyšuje pravděpodobnost úspěšného zneužití session v případě krádeže její identifikace.

2. Přihlašovací formulář je odolný proti SQL injection.

3. Aplikace LMS Claroline není chráněná proti CSFR útokům, a to ani v administrační části aplikace. První byl proveden útok zasláním požadavku GET z jiné stránky, na kterou vstoupil administrátor přihlášený v aplikaci Claroline. Na podvodné stránce byl vložen následující HTML kód. Jako zdroj pro obrázek byl uveden odkaz do administrace, který odstraňuje uživatele číslo 6.

```
<img
src="http://www.mencik.cz/bctest/claroline/claroline/admin/admin
_users.php?cmd=exDelete&user_id=6&offset=0" alt="Obrázek se
nepodařilo načíst" />
```

Po načtení stránky byl odeslán požadavek na získání obsahu GET.

```
GET
http://www.mencik.cz:80/bctest/claroline/claroline/admin/admin_u
sers.php?cmd=exDelete&user_id=6&offset=0 HTTP/1.1
Host: www.mencik.cz
Referer: http://bow-tie-club.cz/test.php
```

Požadavku bylo vyhověno.

```
HTTP/1.1 200 OK
```

Kontrolou v databázi bylo potvrzeno smazání uživatele číslo 6. Bezpečnostní chyba je způsobena tím, že Claroline nepoužívá v odkazech bezpečnostní token nebo jiný druh zabezpečení proti CSFR útokům.

Podobná simulace byla provedena i pro metodu POST, kterou aplikace používá pro odesílání dat z formulářů. Na podvodné stránce byl vytvořen formulář, který odpovídal formuláři pro založení nového uživatele administrátorem. Do HTML

kódu stránky byl vložen jednoduchý skript, který zajistil automatické odeslání formuláře.

```
<script>
  document.formular.submit();
</script>
```

V tomto případě nebyl útok úspěšný, protože formulář je chráněn proti CSFR útokům, ochranu zajišťuje bezpečnostní token ve skrytém formulářovém poli.

```
<input type="hidden" id="csrf_token" name="csrf_token"
value="1e476c7ad7f8d9c6740da550f698605e" />
```

Tuto ochranu obsahují všechny formuláře přístupné po přihlášení aplikace, ale byl opomenut registrační formulář, který bezpečnostní token neobsahuje. Registrační formulář je ve výchozí konfiguraci aplikace volně přístupný a může tak být zneužit například pro automatizované zakládání uživatelů útočníkem.

4. Identifikace session se zapisuje do cookie s náhodně generovaným jménem. Identifikace je přenášena protokolem HTTP nešifrovaně.

```
Set-Cookie:
4c1fbb0e93832c9633700eef755c1626=mdrcdkjavnd0dsif4b13dm7l
a5; path=/
```

Nejedná se o chybu aplikace, ale o vlastnost protokolu, při použití HTTPS nelze odposlechnutou identifikaci použít.

5. Všechna pole formuláře pro registraci uživatele jsou chráněna proti XSS útokům.

6. Editor, který je k dispozici pro zasílání interních zpráv a příspěvků do fóra, je chráněn proti XSS útokům. Vložený skript není odstraněn, ale je vložen do sekce CDATA, která neumožní jeho provedení.

```
<script type="text/javascript">// <![CDATA[
window.location=http://utocnik.cz
// ]]></script>
```

7. Vstupní pole všech formulářů jsou zabezpečená proti XSS útokům.

8. Ve formuláři pro úpravu osobního profilu je bezpečnostní chyba, která umožňuje neoprávněné navýšení práv uživatelem se základní rolí student. Chyba spočívá v možnosti rozšíření formuláře o další vstupní prvek, který je po odeslání zpracován jako validní. Výchozí stav v aplikaci popisuje následující ukázka HTML kódu.

```
<form id="userSettings" method="post"
action="/bctest/claroline/claroline/auth/profile.php"
enctype="multipart/form-data">
…
```

```
<input type="radio" name="platformRole" id="student"
value="student" checked="checked" /><label
for="student">Absolvovat kurz (student)</label>
…
</form>
```

Ve většině prohlížečů (například Opera) existuje uživatelská funkce pro úpravu kódu stránky. Tato funkce nemění kód přímo na serveru, ale změna se vizuálně projeví v prohlížeči. Lze tak snadno přidat další prvek do formuláře a provést úpravu prvků ostatních.

```
<form id="userSettings" method="post"
action="/bctest/claroline/claroline/auth/profile.php"
enctype="multipart/form-data">
…
<input type="radio" name="platformRole" id="student"
value="student" /><label for="student">Absolvovat kurz
(student)</label>
<input type="radio" name="platformRole" id="courseManager"
value="courseManager" checked="checked" /><label
for="courseManager">Vytvořit kurzy (učitel)</label>
…
</form>
```

Po odeslání formuláře nedojde k žádné validaci na počet prvků formuláře, ani ke kontrole oprávnění pro tuto akci. Změna role ze studenta na manažera kurzů se provede úspěšně a útočník získá značnou kontrolu nad aplikací.

9. Ochrana dokumentů, které jsou přístupné pouze přihlášeným uživatelům, případně pouze konkrétním účastníkům kurzů, se na první pohled jevila velice dobře. Údaje o nahraných dokumentech a oprávněních přístupu jsou uloženy do databáze. Aplikace standardně řídí přístup k dokumentům pomocí speciální funkce a neposkytne dokument, ke kterému nemá daný uživatel přístup. Bezpečnostní slabinou je složka, kam se dokumenty na serveru ukládají. Složka „…/claroline/kurzy/#kód kurzu#/document/“ není nijak zabezpečená, takže soubory, které obsahuje, jsou dostupné při zadání přímé URL adresy. Pomocí vhodného skenovacího nástroje není obtížné zjistit její obsah a soubory poté jednoduše stáhnout. Celý mechanismus řízení přístupu k souborům lze tedy celkem snadno obejít.

10. Aplikaci nelze instalovat na server, který nemá potřebnou konfiguraci. Kontrola parametrů serveru a databáze proběhne v jednom z kroků instalace.

11. Šifrování hesel v aplikaci Claroline je poměrně primitivní. Hesla jsou šifrována pomocí kryptografické funkce MD5, není použita kryptografická sůl ani iterace.

```
$password_encrypted = md5($user['password']);
```

Pro dešifrování části těchto hesel lze použít duhové tabulky nebo porovnání s databází otisků. Bezpečnosti nepřispívá ani chybějící kontrola síly a délky zvoleného hesla.

12. Vyhledávací formuláře jsou chráněné proti SQL injection.

13. Parametry v URL jsou chráněné proti SQL injection.

14. Možnost mapování účtu uživatele administrátorem je chráněná proti neoprávněnému použití.

15. Obnovení zapomenutého hesla je v LMS Claroline realizovano velice nebezpečně. Zásadní chybou je, že lze obnovit heslo administrátora pomocí standardního formuláře nebo odkazu s parametrem.

```
…/bctest/claroline/claroline/auth/lostPassword.php?Femail
=admin@mencik.cz&searchPassword=1
```

E-mail administrátora je viditelný na úvodní stránce aplikace, která je přístupná všem návštěvníkům.

Druhou chybou je, že dojde k okamžité obnově hesla. Heslo je tedy změněno (nastaveno na náhodné), aniž by proběhla jakákoli kontrola, zda jde o oprávněný požadavek (například kliknutím na odkaz v e-mailu). Požadavek na změnu hesla není nutné nijak potvrzovat a útočník jej může neomezeně opakovat. Administrátor tak může dočasně ztratit možnost přihlášení do aplikace.

6.3.2.2 Ostatní bezpečnostní nedostatky

LMS Claroline nekontroluje délku a sílu hesla, a to ani u administrátora. Slabá hesla mohou být prolomena slovníkovým útokem.

Druhým, v tomto případě pouze potenciálním bezpečnostním nedostatkem je možnost nešifrovat hesla ukládaná do databáze, tedy ukládat je jako prostý text. Tuto možnost lze zvolit při instalaci aplikace. Hesla jsou citlivé bezpečnostní údaje a musí být šifrována vždy. Tato možnost by neměla v bezpečné aplikaci vůbec existovat.

6.3.3 Zhodnocení bezpečnosti a doporučení

LMS Claroline je na první pohled dobře zpracovaná open-source aplikace s řadou moderních prvků, přívětivým uživatelským rozhraním a snadným ovládáním. Jiné hodnocení aplikace přináší bezpečnostní pohled. LMS Claroline obsahuje několik velmi závažných bezpečnostních chyb, a to v základních oblastech a funkcích aplikace.

Závažnost chyb je bezpečnostní překážkou provozu LMS Claroline na internetu. Aplikace by měla být provozována pouze na privátní podnikové síti. Uživatelé by

měli být registrování pouze ručně administrátorem. Ani tato striktní opatření nezabrání popsaným bezpečnostním rizikům, ale značně je omezí.

Pro zvýšení bezpečnosti aplikace je důležité pravidelně instalovat bezpečnostní updaty, které jsou dostupné na webových stránkách *www.claroline.net*. Vývojáři Claroline pochopili, že je bezpečnost jejich aplikace velmi důležitá, důkazem jsou poslední verze aplikace, které se z velké části věnují právě opravě bezpečnostních chyb. [**46**]

6.4 Srovnání bezpečnosti testovaných LMS

Z předchozích kapitol je zjevné, že testované LMS mají různou úroveň bezpečnosti. V následující tabulce je uvedeno jejich vzájemné srovnání.

LMS	Počet chyb nízké závažnosti	Počet chyb střední závažnosti	Počet chyb vysoké závažnosti	Celkový počet chyb
Moodle	2	1	0	3
eFront	3	1	1	5
Claroline	1	5	1	7

Tabulka 18 – Srovnání bezpečnosti testovaných LMS [AUTOR]

Hodnocení odhalených bezpečnostních slabin vychází ze závažnosti možných důsledků při zneužití bezpečnostní slabiny útočníkem. Dopad stejných nebo podobných slabin může být v různých systémech odlišné, proto je závažnost vždy vztažena ke konkrétnímu LMS.

7 Závěr

7.1 Dosažené cíle a shrnutí výsledků

V práci byly popsány aktuální bezpečnostní hrozby pro webové aplikace a představeny možnosti obrany proti těmto hrozbám. Na základě poznatků z odborných zdrojů byl popsán protokol HTTP a HTTPS s bezpečnostní vrstvou SSL/TLS. Byl nastíněn jejich vliv na bezpečnost webových aplikací a princip fungování.

Byl představen pojem e-learning, jeho přednosti a postavení v moderním podniku. Byla popsána specifická pozice open-source řešení v oblasti e-learningu. Dále byly charakterizovány tři významné open-source e-learningové systémy. Na základě získaných poznatků byla identifikována a popsána bezpečnostní rizika, která plynou z povahy webových open-source aplikací a e-learningu.

Byl představen pojem penetrační testování aplikací a jeho pozice ve srovnání s ostatními prvky oblasti testování bezpečnosti informací. Dále byla představena jedna z metodik penetračního testování zaměřená na webové aplikace. Na základě studia této metodiky a souvisejících materiálů byl stanoven postup pro realizaci praktické části práce.

V praktické části byly popsány výsledky penetračního testování vybraných LMS. Všechny provedené testy byly doplněny komentářem. Nalezené bezpečnostní chyby byly demonstrovány na konkrétních příkladech z dané aplikace. U každé aplikace bylo uvedeno celkové zhodnocení bezpečnosti, doporučení pro bezpečnější provoz.

7.2 Přínos autora

Největším přínosem jsou výsledky penetračních testů a konkrétní nalezené chyby. Výstupy praktické části mohou být využity ke zvýšení bezpečnosti testovaných aplikací. Popsaný postup testování a představená metodika mohou sloužit pro testování libovolných webových aplikací. Zhodnocení bezpečnosti a představení možných hrozeb jednotlivých řešení může pomoci českým podnikům při volbě LMS.

Dalším přínosem je vytvoření uceleného přehledu nejčastějších bezpečnostních hrozeb, vysvětlení principu jejich fungování a možné obrany proti nim. Přehled najde široké využití mezi testery webových aplikací, jejich vývojáři i návrháři.

7.3 Využitelnost dosažených výsledků a náměty pro další řešení

Zjištěné a popsané výsledky mohou provozovatelé open-source LMS využít k vytvoření bližší představy o bezpečnostních hrozbách a důležitosti bezpečnostních opatření. Vývojáři konkrétních aplikací získají podklady pro opravu bezpečnostních chyb. Přehled a popis nejčastějších hrozeb je využitelný pro všechny vývojáře a návrháře webových aplikací.

Námětem pro další možné řešení by mohlo být provedení obdobných penetračních testů v konkrétních společnostech. Taková práce by nemusela být omezena na open-source software a mohla by přinést srovnání bezpečnosti mezi komerčními LMS a těmi s otevřeným zdrojovým kódem.

TERMINOLOGICKÝ SLOVNÍK

Termín	Zkratka	Význam	Zdroj
Autentizace		Jednoznačné ověření identity subjektu, která žádá o přístup do systému.	[**42**]
Autorizace		Proces získávání oprávnění přístupu k objektu aplikace na základě úspěšné autentizace.	[**42**]
Bezpečnostní token		Myšleno náhodný řetězec znaků použitý k dodatečnému ověření identity uživatele.	[AUTOR]
Hash		Výstup kryptografické (hashovací) funkce. Hashovací funkce převádí libovolný vstup na řetězec pevně dané délky.	[AUTOR]
Informační a komunikační technologie	ICT	Komplex technických, programových a komunikačních prostředků, které umožňují zpracování aplikací a manipulaci s daty.	[**29**]
Informační systém	IS	Systém, myšleno založený na počítačích, sloužící k přenášení, zpracování a vyjádření informací.	[**29**]
Learning management systém	LMS	Systém pro řízení výuky v rámci e-learningu	[AUTOR]
Open-source	OSS	Software s volně dostupným, šiřitelným a modifikovatelným kódem. Opakem je proprietární software.	[AUTOR]
Operační systém	OS	Základní programové vybavení počítače, které řídí všechny ostatní programy zpracovávané počítačem.	[**29**]
Session		Informace o uživateli webové aplikace, která je uložená na serveru a slouží k záznamu dat o stavu klienta.	[AUTOR]

SEZNAM POUŽITÉ LITERATURY

[1] OWASP. *Category: OWASP Top Ten Project - OWASP* [online]. 12. červen. 2013 [cit. 2014-únor-23]. Dostupné z: http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf

[2] KOSEK, J. *Bezpečnost webových aplikací* [online]. 5. prosinec. 2013 [cit. 2014-únor-24]. Dostupné z: http://www.kosek.cz/vyuka/4iz228/prednasky/bezpecnost/frames.html

[3] SECURITY-PORTAL.CZ. Security-portal.cz. In: *SQL Injection | Security-portal.cz* [online]. 23. leden. 2005 [cit. 2014-únor-24]. Dostupné z: http://www.security-portal.cz/clanky/sql-injection

[4] ZÁVODSKÝ, P. Root.cz. In: *Nebezpečné prolomení ověřování identity a správy sezení - Root.cz* [online]. 7. září. 2011 [cit. 2014-únor-24]. Dostupné z: http://www.root.cz/clanky/nebezpecne-prolomeni-overovani-identity-a-spravy-sezeni/

[5] OWASP. *Top 10 2013-A2-Broken Authentication and Session Management - OWASP* [online]. 12. červen. 2013 [cit. 2014-únor-24]. Dostupné z: https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management

[6] ZÁVODSKÝ, P. Root.cz. In: *XSS stále na scéně - Root.cz* [online]. 6. duben. 201 [cit. 2014-únor-24]. Dostupné z: http://www.root.cz/clanky/xss-stale-na-scene/

[7] OWASP. *Cross-site Scripting (XSS) - OWASP* [online]. 3. únor. 2014 [cit. 2014-únor-24]. Dostupné z: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[8] OWASP. *XSS (Cross Site Scripting) Prevention Cheat Sheet - OWASP* [online]. 5. únor. 2014 [cit. 2014-únor-24]. Dostupné z: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

[9] OWASP. *Top 10 2013-A4-Insecure Direct Object References - OWASP* [online]. 14. červen. 2013 [cit. 2014-únor-25]. Dostupné z: https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References

[10] CISODESK. *Insecure Direct Object References – Introduction and Mitigation Steps* [online]. 12. červenec. 2012 [cit. 2014-únor-25]. Dostupné z: http://www.cisodesk.com/web-application-security/threats-mitigation/insecure-direct-object-references/

[11] OWASP. *Top 10 2013-A5-Security Misconfiguration - OWASP* [online]. 23. červen. 2013 [cit. 2014-únor-25]. Dostupné z: https://www.owasp.org/index.php/Top_10_2013-A5-Security_Misconfiguration

[12] OWASP. *Configuration - OWASP* [online]. 3. květen. 2009 [cit. 2014-únor-25]. Dostupné z: https://www.owasp.org/index.php/Configuration

[13] OWASP. *Top 10 2013-A6-Sensitive Data Exposure - OWASP* [online]. 23. červen. 2013 [cit. 2014-únor-25]. Dostupné z: https://www.owasp.org/index.php/Top_10_2013-A6-Sensitive_Data_Exposure

[14] DEFUSE SECURITY. CrackStation. In: CRACKSTATION. *Secure Salted Password Hashing - How to do it Properly* [online]. 10. únor. 2014 [cit. 2014-únor-25]. Dostupné z: https://crackstation.net/hashing-security.htm

[15] VRÁNA, J. PHP triky. In: *PHP triky - Ukládání hesel bezpečně* [online]. 16. prosinec. 2013 [cit. 2014-únor-25]. Dostupné z: http://php.vrana.cz/ukladani-hesel-bezpecne.php

[16] OWASP. *Cross-Site Request Forgery (CSRF) - OWASP* [online]. 9. listopad. 2013 [cit. 2014-únor-26]. Dostupné z: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

[17] AUGER, R. cgisecurity. In: *The Cross-Site Request Forgery (CSRF/XSRF) FAQ* [online]. 28. duben. 2010 [cit. 2014-únor-26]. Dostupné z: http://www.cgisecurity.com/csrf-faq.html

[18] OWASP. *OWASP CSRFGuard - OWASP* [online]. 2. únor. 2014 [cit. 2014-únor-26]. Dostupné z: https://www.owasp.org/index.php/CSRFGuard

[19] OWASP. *Top 10 2013-A9-Using Components with Known Vulnerabilities - OWASP* [online]. 23. červen. 2013 [cit. 2014-únor-26]. Dostupné z: https://www.owasp.org/index.php/Top_10_2013-A9-Using_Components_with_Known_Vulnerabilities

[20] SANS. *IT Information Security Awareness Training* [online]. 24. říjen. 2013 [cit. 2014-únor-26]. Dostupné z: http://www.securingthehuman.org/developer/videos/using-known-vulnerable-components

[21] KOSEK, J. *Protokol HTTP* [online]. 8. prosinec. 2011 [cit. 2014-únor-27]. Dostupné z: http://www.kosek.cz/vyuka/4iz228/prednasky/http/frames.html

[22] KOSEK, J. *PHP Tvorba interaktivních webových aplikací*. Praha: Grada Publishing, spol. s.r.o. 1999. ISBN 80-7169-373-1.

[23] INTERNET ENGINEERING TASK FORCE. In: *RFC 6265 - HTTP State Management Mechanism* [online]. 2011 [cit. 2014-únor-27]. ISSN 2070-1721

[24] HREBENAR, J. Zive.cz. In: *SSL a TLS: Úvod do SSL a TLS* [online]. 27. listopad. 2010 [cit. 2014-únor-27]. Dostupné z: http://programovani.blog.zive.cz/2010/11/ssl-a-tls-uvod-do-ssl-a-tls/

[25] INTERNET ENGINEERING TASK FORCE. In: *RFC 6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0* [online]. 2011 [cit. 2014-únor-27]. ISSN: 2070-1721

[26] INTERNET ENGINEERING TASK FORCE. In: *RFC 2818 - HTTP Over TLS* [online]. 2000 [cit. 2014-únor-27]. Dostupné z: https://tools.ietf.org/html/rfc2818

[27] WIKE, A. Thawte. In: *Difference between SSL and TLS* [online]. 30. květen. 2013 [cit. 2014-únor-27]. Dostupné z: https://community.thawte.com/blog-posts/difference-between-ssl-and-tls

[28] INTERNET ENGINEERING TASK FORCE. In: *RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List …* [online]. 2008 [cit. 2014-únor-27]. Dostupné z: http://tools.ietf.org/html/rfc5280

[29] GÁLA, L. J. POUR a Z. ŠEDIVÁ. *Podniková informatika 2. přepracované a aktualizované vydání*. Praha: Grada Publishing, a.s. 2009. ISBN 978-80-247-2615-1.

[30] HIRŠ, M. SystemOnLine. In: *E-learning - nová podoba firemního vzdělávání* [online]. 7. srpen. 2001 [cit. 2014-únor-28]. Dostupné z: http://www.systemonline.cz/clanky/e-learning-nova-podoba-firemniho-vzdelavani.htm

[31] ŠIKO, P. SystemOnLine. In: *E-learning jako další varianta vzdělávání* [online]. 2003 [cit. 2014-únor-28]. Dostupné z: http://www.systemonline.cz/clanky/e-learning-jako-dalsi-varianta-vzdelavani.htm

[32] DAVIS, B. C. CARMEAN a E. WAGNER. Learning Solution Magazine. In: *Moodle™ Moves To the Front of the LMS Adoption Pack* [online]. 15. říjen. 2009 [cit. 2014-únor-28]. Dostupné z: http://www.learningsolutionsmag.com/articles/111/moodle-moves-to-the-front-of-the-lms-adoption-pack

[33] MOODLE. *Moodle Statistics* [online]. 2014 [cit. 2014-únor-28]. Dostupné z: https://moodle.org/stats/

[34] MOODLE. *About Moodle - MoodleDocs* [online]. 2014 [cit. 2014-bčezen-01]. Dostupné z: http://docs.moodle.org/26/en/About_Moodle

[35] FREE SOFTWARE FOUNDATION. In: *GNU GENERAL PUBLIC LICENSE Version 3* [online]. 29. červen. 2007 [cit. 2014-únor-28]. Dostupné z: http://www.gnu.org/licenses/gpl-3.0.txt

[36] EFRONT. *About eFront - eFront wiki* [online]. 22. březen. 2012 [cit. 2014-březen-01]. Dostupné z: http://wiki.efrontlearning.net/About_eFront

[37] SCORM. *SCORM - SCORM Explained* [online]. 2013 [cit. 2014-březen-01]. Dostupné z: http://scorm.com/scorm-explained/

[38] CLAROLINE. *Claroline Story - Claroline Learning Management System (LMS)* [online]. 2012 [cit. 2014-březen-01]. Dostupné z: http://www.claroline.net/the-story-of-claroline/?lang=en

[39] ŠÍN, M. Linux EXPRES. In: *Srovnání e-learningových systému Claroline a Moodle* [online]. 6. květen. 2013 [cit. 2014-březen-01]. Dostupné z: http://www.linuxexpres.cz/software/lms-claroline-e-learningovy-system-ocima-uzivatele-moodlu

[40] CLAROLINE. *Features - Claroline Learning Management System (LMS)* [online]. 2012 [cit. 2014-březen-01]. Dostupné z: http://www.claroline.net/features/?lang=en

[41] WEIPPL, E. eLearn Magazine. In: *Security in e-learning* [online]. 2005 [cit. 2014-březen-01]. Dostupné z: https://elearnmag.acm.org/featured.cfm?aid=1070943

[42] DOUCEK, P. et al. *Řízení bezpečnosti informací 2. rozšířené vydání o BCM*. Praha: Professional Publishing, 2011. ISBN 978-80-7431-050-8.

[43] OWASP FOUNDATION. In: *OWASP Testing Guide V3.0* [online]. 16. prosinec. 2008 [cit. 2014-březen-01]. Dostupné z: https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf

[44] *Category:OWASP WebScarab Project - OWASP* [online]. 23. leden. 2014 [cit. 2014-březen-01]. Dostupné z: https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project

[45] MOODLE. *Download Standard Packages* [online]. 26. únor. 2014 [cit. 2014-březen-04]. Dostupné z: http://download.moodle.org/

[46] CLAROLINE. *Downloads - Claroline Learning Management System (LMS)* [online]. 2013 [cit. 2014-březen-06]. Dostupné z: http://www.claroline.net/downloads/?lang=en

[47] BARÁŠEK, J. In: *Hashovací funkce a metody jednosměrného šifrování* [online]. 25. říjen. 2013 [cit. 2014-únor-25]. Dostupné z: http://www.php.baraja.cz/index.php?kategorie=navody&page=hashovani

SEZNAM OBRÁZKŮ A TABULEK

SEZNAM OBRÁZKŮ

SEZNAM TABULEK