Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Nette Framework z hlediska bezpečnosti proti útokům

Michal Sladký

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Michal Sladký

Informatika

Název práce

Nette Framework z hlediska bezpečnosti proti útokům

Název anglicky

Use of Nette Framework for defense against web-based attacks

Cíle práce

Cílem práce je návrh obrany proti webovým útokům s využitím Nette Framework. V práci budou představeny a analyzovány nejčastěji se vyskytující útoky na webové stránky a následně vysvětlena či navržena možná obrana proti nim.

Metodika

Řešení problematiky bakalářské práce bude založeno na studiu a analýze odborných informačních zdrojů. Na základě získaných informací budou vypracovány jednotlivé části práce a vyhodnoceny možnosti Nette Framework a jeho komponent z hlediska ochrany proti útokům. Na základě syntézy teoretických poznatků a výsledků praktické části práce budou formulovány závěry bakalářské práce.

Doporučený rozsah práce

30 – 40 stran

Klíčová slova

webové útoky, web, programování, php, nette, framework

Doporučené zdroje informací

ALCORN, W., FRICHOT, C. a ORRU, M. The Browser Hacker's Handbook. John Wiley & Sons, Inc., 2014. 978-1-118-66209-0.

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

FOGIE, S., GROSMANN, J., HANSEN, R., RAGER, A. a Petkov, P. D. XSS Attacks: Cross Site Scripting Exploits and Defense. Syngress Publishing, Inc., 2007. 978-1597491549.

WU, H. a ZHAO, L. Web Security: A WhiteHat Perspective. Taylor & Francis Group LLC, 2015. 9781466592612.

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

Ing. Petr Benda, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 28. 10. 2015

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2015

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 09. 03. 2016

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Nette Framework z hlediska bezpečnosti proti útokům" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 8.3.2016 _____

Poděkování

Rád bych poděkoval Ing. Petru Bendovi za jeho čas a odbornou pomoc při vypracovaní této práce.

Nette Framework z hlediska bezpečnosti proti útokům

Use of Nette Framework for defense against web-based attacks

Souhrn

Bakalářská práce pojednává o technikách, jak předejít bezpečnostním rizikům. Teoretická část se zaměřuje na obecné fungování webových aplikací, poté popisuje výběr nejzávažnějších útoků a ke konci představí Nette Framework, který je použit pro vytvoření zkoumaného prototypu.

Praktická část se týká samotné analýze kódu a analýze zvoleným penetračním testovacím nástrojem a návrhu ochrany sledovaných útoků. Na závěr je zhodnocena schopnost Nette Framework zajistit bezpečnost.

Summary

The bachelor thesis speaks about techniques of how to prevent your web application from security holes. Research focuses on general functionality of web applications, then describes the most serious attacks and at last, presents Nette Framework which was used to create the analyzed prototype.

Practical part is about code analysis itself and analysis of a chosen penetrating tool. Based on the results, a way how to protect your application against these attacks will be suggested. At the end, the ability of Nette Framework to prevent you from attacks will be evaluated.

Klíčová slova

webové útoky, web, programování, php, nette, framework

Keywords

web attacks, web, programming, php, nette, framework

1 Úvod

V dnešní době si většina lidí nedokáže představit svět bez počítačů a Internetu. Web se stal za několik posledních let velice populárním a denně vzniká nespočet webových stránek a aplikací. Většina aplikací je vytvářena se záměrem upoutat čtenáře, či uživatele. Značná míra úsilí je tedy věnována vzhledu, použitelnosti, funkčnosti, dostupnosti a vnímání aplikace jako celku. Co je často opomíjeno, je bezpečnost.

Dnes již není neobvyklé nakupovat skrze e-shopy, či vyřizovat bankovní transakce v internetovém bankovnictví. U těchto typů aplikací je logické očekávat maximální bezpečnost. Mylná je však představa, že by například blog nevyžadoval žádné zabezpečení. Prostředí prohlížeče je rozsáhlé a fantazie útočníků veliká. Útočník může odposlouchávat nešifrovanou komunikaci, podvrhnout na blog reklamy, či falešné výzvy, případně se dostat k citlivým datům a získat tak přístupové údaje na jinou stránku. Proto by každá webová aplikace měla být zaměřena v první řadě na bezpečnost.

Možností, jak vytvořit aplikaci, je celá řada. Existuje několik programovacích jazyků a technologií. Všechny více, či méně zajišťující bezpečnost na určité úrovni. Ostatní musí zajistit programátor, protože jedině on ví, jaká data jsou přípustná a co má aplikace dělat.

Pro tuto práci byl vybrán programovací jazyk webů - PHP a nejrozšířenější PHP Framework v České Republice – Nette Framework.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je návrh obrany proti webovým útokům s využitím Nette Framework. V práci budou představeny a analyzovány nejčastěji se vyskytující útoky na webové stránky a následně vysvětlena či navržena možná obrana proti nim.

2.2 Metodika

Řešení problematiky bakalářské práce bude založeno na studiu a analýze odborných informačních zdrojů. Na základě získaných informací budou vypracovány jednotlivé části práce a vyhodnoceny možnosti Nette Framework a jeho komponent z hlediska ochrany proti útokům. Na základě syntézy teoretických poznatků a výsledků praktické části práce budou formulovány závěry bakalářské práce.

3 Teoretická východiska

Svět techniky prošel za několik posledních let mnoha změnami a spolu s nimi i přístup, jakým jsou tvořeny a prezentovány weby. Většina webů, které dnes navštěvujeme, nejsou klasickými webovými stránkami, které by zobrazovaly pouze svůj obsah. Z velké většiny se jedná o stovky až tisíce řádků obslužného kódu na straně serveru, který zpracovává požadavky uživatelů. Tento software se jako celek nazývá webová aplikace.

3.1 Webová aplikace

Webová aplikace je specifická tím, že interaguje s uživatelem, zpracovává vstupy, spolu s databází data ukládá a nazpět zasílá uživateli vyžádaná data. Jedná se tedy o aplikaci, která se řídí modelem klient-server. Klientem je v tomto případě uživatel, který otevře web ve svém webovém prohlížeči. Serverem je zpravidla speciální počítač umístěný na jiné geografické poloze než cílový klient a přistupuje se k němu pomocí sítě Internet a služby WWW.

3.1.1 Strana klienta

Stranu klienta představují tři základní technologie, které ve výsledném prohlížeči vytváří prostředí webu. Jsou jimi HTML, CSS a JavaScript.

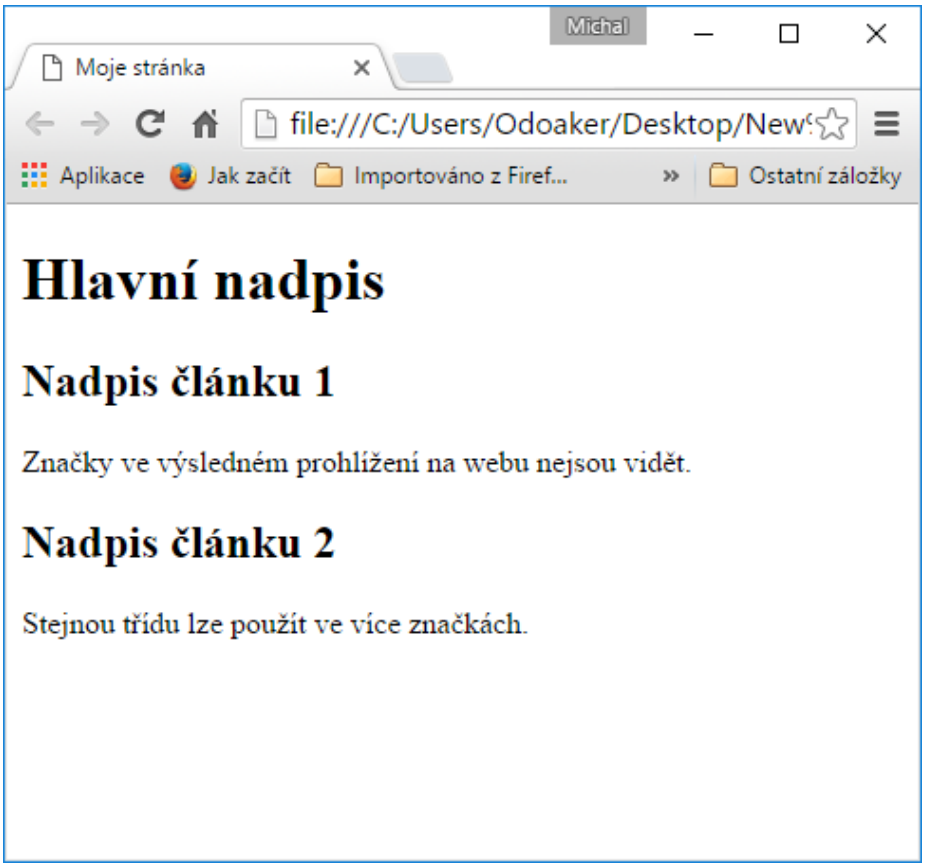
HTML

Značkovací jazyk HTML (HyperText Markup Language) je klíčovým pro vytvoření webových stránek. Od počátku webu vzniklo pět verzí HTML. Za vývoj a prohlašování oficiálně podporovaných verzí HTML je zodpovědný konsorcium W3C. Za standard se dnes považuje verze HTML5. Ze zkratky HTML je patrné, že se jedná o jazyk značkovací (M = markup). To znamená, že text v dokumentu je členěn a popsán pomocí značek (tzv. tagů).

Značky jsou zpravidla párové a spolu s textem, který uzavírají, tvoří objekt nazvaný element. Značky mohou být navíc obohaceny o atributy, které přidávají celému elementu specifické vlastnosti. Za téměř univerzální atributy se považují id, class, title a style. Tyto čtyři vlastnosti lze použít ve většině značek. Zápis a funkci značek a atributů lze vidět v ukázkovém kódu. (Larsen, 2013)

```
<!DOCTYPE html>
<html>
 <head>
  <title>Moje stránka</title>
 </head>
 <body>
  <article id="cl1">
   <h1>Hlavní nadpis</h1>
   <p>Značky ve výsledném prohlížení na webu nejsou vidět.</p>
  </article>
  <aricle id="cl2">
   <h1>Další nadpis</h1>
   <p>Stejnou třídu lze použít ve více značkách.</p>
  </article>
 </body>
</html>
```

Tento kód zobrazí webový prohlížeč Google Chrome[1] takto:



Obrázek 1: Ukázka webové stránky – zdroj: autor

[1] https://www.google.com/chrome

CSS

Počítač dospěl do fáze, kdy funkčnost se stala samozřejmostí a na vzhled a uživatelské rozhraní se klade daleko větší důraz. HTML poskytuje základní možnosti stylování, jako jsou barva, pozice obrázků, velikost a další. Není to však dostačující a zdrojový kód stránek se stává hůře čitelným. Proto, aby bylo možné udělat stránky zajímavějšími, se využívá kaskádových stylů. Kaskádové styly se zpravidla zapisují do samostatného souboru a pomocí HTML značky <link> se připojí se v hlavičce HTML kódu stránky. Následující definice obarví pozadí značky <body></body> na žlutou, nastaví barvu všech textů v této značce na modrou a definuje písmo. (Larsen, 2013)

```
body {
  background-color: yellow;
  color: blue;
  font-family: arial, verdana, sans-serif;
}
```

JavaScript

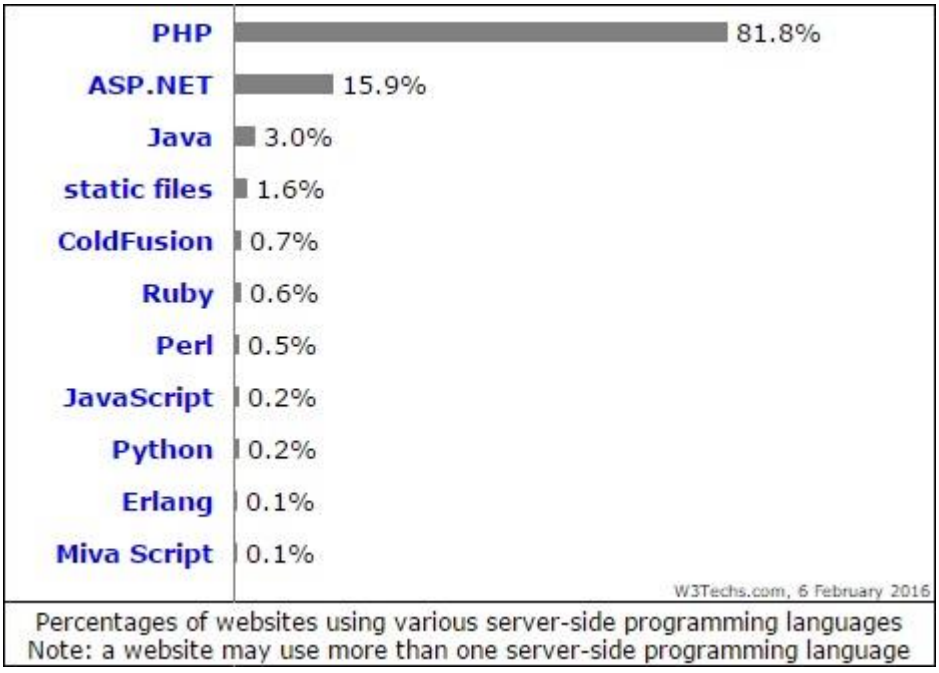
V dnešní době existuje mnoho programovacích jazyků. Jedním z nich je JavaScript, který je téměř absolutně dominujícím mezi jazyky pro tvorbu klientského prostředí. Pomocí JavaScriptu je možné provádět matematické operace, vytvářet zajímavé animace a efekty, mazat, měnit, či přidávat další HTML elementy, případně s technologií AJAX (Asynchronní JavaScript a XML) měnit určité části webu bez nutnosti zobrazení celé stránky znovu.

Při tvorbě rozsáhlých aplikací je snaha převést většinu práce na klientské zařízení, z důvodu nepřetěžování webových serverů. Znatelným ulehčením webového serveru je například zobrazování a výpočty grafů pomocí JavaScriptu. Na některé operace se však nelze spoléhat na klientské straně. Jsou jimi zejména aktivity týkajících se bezpečnosti.

```
var pocet_jablek = 5;
var pocet_hrusek = 2;
var soucet = pocet_jablek + pocet_hrusek;
document.write(soucet);
```

3.1.2 Strana serveru

Možností jak zpracovávat požadavky na straně serveru je celá řada. Hlavním cílem projektu webové aplikace je stanovit, jaká technologie bude využita, resp. jaký programovací jazyk a od toho se odvíjející vývojové prostředí a nástroje. Podle statistik portálu W3Techs[2] je patrné, že PHP je nejčastěji používaným jazykem pro tvorbu webů (viz



Obrázek 2: Popularita programovacích jazyků na straně serveru - zdroj: w3techs.com

obrázek 2).

PHP je pouze programovací jazyk umožňující vytvářet aplikace. Pro chod a samotné zpracování kódu je zapotřebí webový server. Nejčastěji spojován s PHP je webový server Apache. (Meloni, 2003)

Ve většině případů je potřeba data trvale uložit, to je možné docílit pomocí PHP v podobě ukládání souborů na server. Tento proces může například usnadnit souborová databáze SQLite. Přesto je mnohem častější a pohodlnější využít tradičních relačních databází, které nabízí daleko více možností a rozsáhlejší konfiguraci. Pro tuto bakalářskou práci byla vybrána databáze MySQL.

[2] http://w3techs.com/

Apache

Apache webový server je open-source projekt. Správu a vývoj zajišťuje společnost Apache Software Foundation a za posledních 20 let je nejpopulárnějším webovým serverem. Apache se dá nainstalovat v podobě předkompilovaného balíčku zejména pro operační systém Windows, avšak častější je samotný webový server „postavit“ ze zdroje na OS Linux/UNIX. Předkompilované instalační balíčky mají výhodu ve své jednoduchosti, oproti druhému způsobu je ovšem opožděný upgrade nových verzí a omezená konfigurace. Samotné instalační balíčky mohou být také součástí tzv. triády – což je obslužný software, který zajišťuje chod PHP, Apache a MySQL.

PHP

Jedním z programovacích jazyků strany serveru je PHP. PHP kód je tedy zpracován pouze na straně serveru a není možné, aby vnější uživatel viděl samotný zdrojový kód. Proces, jakým je požadavek o PHP soubor zpracován, je následující:

1. Webový prohlížeč požádá o PHP dokument.
2. Webový server pošle požadavek PHP parseru, který je přímo ve webovém serveru, případně existuje samostatně.
3. PHP parser vyhledává PHP kód
4. Pokud PHP kód najde, vykoná ho a nahradí ho výstupem (pokud nějaký je)
5. Výsledný dokument je zaslán webovému serveru
6. Webový server přeposílá tento soubor webovému prohlížeči
7. Webový prohlížeč zobrazí výsledek.

Jak pozná parser, že se jedná o PHP kód? V samotném souboru označeným jako PHP se může vyskytovat také HTML kód. K vyjádření, že se jedná o část, která má být zpracována PHP parserem, se používají značky <?php a ?>. Jsou i jiné alternativy, ale ty musí být zpravidla povoleny v konfiguračním souboru webového serveru. Následující kód ukazuje použití PHP uvnitř HTML kódu.

```
<html>
<head>
 <title>Moje stránka</title>
</head>
<body>
 <h1>Tabulka</h1>
 <table>
 <?php
  for (int $i = 1; $i <= 5; $i++) {
   echo „<tr><td>“ . $i . „.řádek</td></tr>“;
  }
 ?>
 </table>
</body>
</html>
```

V prohlížeči bude výsledek vypadat následovně:

Obrázek 3: Ukázková stránka s PHP - zdroj: autor

Jazyk PHP je velice oblíbený svou jednoduchostí. Zpočátku vývoje byl primárně zamýšlen jako doplněk k HTML, či „šablonovací“ jazyk. V posledních několika letech se však začal využívat i pro vývoj rozsáhlých aplikací. K tomu, aby bylo možné vyvíjet rychle

a efektivně, se začaly ve světě PHP programátorů vyskytovat předpřipravené aplikace – tzv. PHP Frameworky.

PHP Framework

PHP Framework je jakási kostra aplikace. Ulehčuje zejména monotónní práci, kterou je potřeba vykonat při většině projektů jako je připojení k databázi, ošetřování výstupů, neustálé připojování dalších PHP souborů do jiných a mnoho dalšího.

Podle portálu Sitepoint, který se zabýval průzkumem PHP frameworků na jaře 2015, bylo zjištěno, že mezi nejoblíbenější framework v Jižní a Severní Americe a Indii patří Lavarel, v Evropě je velice rozšířený Symfony2 a Česká Republika se může pyšnit vlastním a úspěšným Nette Framework. Zmíněné frameworky jsou jen nepatrným zlomkem reálného počtu. Dalšími významnými jsou CodeIngniter, Zend Framework 2, Yii 1, CakePHP, Slim, PHPixie.

Ačkoliv je jich mnoho, princip sdílejí většinou stejný a často se tak vývojáři inspirují ostatními při programování jejich vlastních frameworků. Hlavním benefitem samotného frameworku je ve většině případů prostředí a částečná podpora objektově orientovaného programování, pomocí něhož je samotný kód lehce čitelný a nabízí více možností oproti klasickému procedurálnímu

MySQL

Jak již bylo řečeno, MySQL je databází relační. Přístup relačních databází spočívá v tom, že data jsou uspořádána do tabulek. Každá tabulka má definovaný počet sloupců a každý sloupec je nějak nazván a vyjadřuje určitou datovou strukturu. Jedním ze sloupců by měl být tzv. primární klíč (jednoznačně identifikující položka řádku), kterým je zpravidla ID. Jednotlivé řádky v tabulce potom představují záznam.

PHP poté využívá tato data a zasílá na MySQL dotaz tzv. query. Pomocí těchto dotazů je možné pracovat s požadovanou databází, přidávat tabulky, měnit struktury, ale zejména získávat data a přidávat nové záznamy. Příkladem může být následující dotaz pro výběr záznamů z tabulky Osoby, které splňují podmínku id = 1.

```
SELECT * FROM Osoby WHERE id=1
```

MySQL nabízí desítky funkcí. Pro rychlejší vyhledávání se dají sloupce indexovat, nastavovat speciální klíče (cizí, unikátní). Je možné vybírat z více tabulek se složenými podmínkami, vytvářet pohledy (obraz reálné tabulky), triggery (tzv. spouště, např. při akci INSERT je takový trigger spuštěn), specifické pro MySQL jsou i časované události.

3.2 Webové útoky

Mapováním a analýzou webových útoků se zabývá komunita projektu OWASP (Open Web Application Security Project). Od roku 2004 vypadá každé tři roky list 10 nejzávažnějších webových útoků. V této práci budou zahrnuty některé z nich.

3.2.1 Injektování

Útokem pomocí injektování se považuje lstivé podvrhnutí nedůvěryhodných dat interpretu – v tomto případě PHP. Interpret těmito daty může, aniž by to zamýšlel, změnit dotazy a příkazy směřované primárně na databázi a tím umožnit přístup k citlivým datům.

Příklad útoku:

- Útočník změní hodnotu parametru GET v URL:

http://localhost/ucty/detail?id=1 OR 1 = 1

- Interpret poté slepě přepošle dotaz s parametrem databázi:

*query="SELECT * FROM ucty WHERE id=" . $_GET['id'];*

Pomocí tohoto způsobu je možné získat informace o všech účtech vedených v databázi. Útoky tohoto typu jsou velice nebezpečné a mohou mít i daleko větší dopad, než je získání citlivých dat a uživatelských údajů.

3.2.2 Chybná autentizace a správa relace

Těchto útoků se využívá k získání identit a přihlášení místo jiných uživatelů – zpravidla s většími pravomocemi. Jde zejména o nezašifrovanou komunikaci při zasílání hesel, opomenutí zahashovat hesla, případně nenastavení doby platnosti tokenů.

Příklad útoku:

- Webová aplikace podporuje URL rewriting a zároveň zobrazuje v URL adrese id relace.

http://test.cz/udaje/detail;jsessionid= KGLD0WOD0GGHP?id=5

Uživatel zašle adresu druhé osobě a webová aplikace druhou osobu považuje za našeho původního uživatele. Druhá osoba, ač by neměla být oprávněná, může získat přístup k našemu uživateli.

3.2.3 Cross-Site Scripting (XSS)

XSS je podobný typ útoku jako Injektování. Je to útok založený na podvrhnutí kódu, který pak interpret vykoná. Hlavním nástrojem, kterým je možné úspěšný útok vykonat je JavaScript, proto je útok XSS cílený na webové prohlížeče. Pomocí XSS je možné přesměrovat uživatele na jiné stránky, webové stránky přetvořit, ukrást relace, apod.

Příklad útoku:

Útočník změní parametr GET v URL, místo klasického jména vloží javascriptový kód:

http://localhost/ucty/detail?jmeno=<script>alert('XSS');</script>

PHP vypisuje zadané jméno na obrazovku, např. po přihlášení.

```
<?php
  $jmeno = $_GET['jmeno'];
  echo „Jmeno: “ . $jmeno;
?>
```

Pomocí jednoduchého alertu je možné zjistit zranitelnost proti XSS a nadále využít celou škálu funkcí, v tomto případě, javascriptu k získání věryhodnějších dat, nebo jen odlákání uživatele na svou vlastní stránku, kde může předat své citlivé informace do rukou útočníka dobrovolně, aniž by o tom věděl.

3.2.4 Nezabezpečené přímé odkazy na objekty

Za přímý odkaz se považuje jakýkoliv odkaz, který byl vystaven na vnitřní objekt implemenetace. Takovým objektem mohou být soubory, adresáře nebo databázové klíče. S těmito objekty je poté operováno bez jakékoliv kontroly, či oprávnění a útočníci mohou manipulovat s daty a získávat přístup do nepřípustných oblastí.

Příklad útoku:

Opět změníme parametr v URL adrese, nyní je naším cílem získat informace zejména o účtu s větší pravomocí:

http://localhost/ucty/detail?ucet=admin

Webová aplikace neověřuje údaje databázového dotazu:

*mysql_query(„SELECT * FROM ucty WHERE ucet=“ . $_GET['ucet']);*

Nyní neuvažujeme zranitelnost SQL injekce, útok na přímé odkazy je možné vykonat, i když je injektování ošetřeno. V tomto případě není ověřen autorizační přístup a útočník může získat informace o jakémkoliv účtu.

3.2.5 Nezabezpečená konfigurace

Nezabezpečená konfigurace se týká webového serveru, použitých frameworků, knihoven, databázového serveru, platformy, v krajních případech i samotného kódu. K zamezení těchto útoků je potřeba spolupráce jak samotných vývojářů, tak správců serveru. Oproti předešlým útokům je potřeba neustále sledovat vývoj daných softwarů, nálezů bezpečnostních trhlin a řádně je aktualizovat.

Příklad útoku:

U adresářů, které nemají být přístupné běžným uživatelům, není zakázán výpis. Útočník má tedy volné pole působnosti, může najít jakýkoliv soubor. V jistých případech může dekompilovat Java, C# zdrojový kód a získá představu o nedostatcích ve vašem kódu.

Další typickou chybou je neodstranění ukázkových kódů stažené aplikace, či modulu. Tyto ukázky neobsahují kvůli jednoduchosti ochranné části a představují tak bezpečnostní riziko.

3.3 Nette Framework

Nette Framework je framework napsaný pro PHP a řídí se modelem MVP. Minimální požadovaná verze PHP je aktuálně PHP 5.3. Celý framework využívá objektového programování (v mezích PHP) a je rozdělen na 4 hlavní části, kterými jsou:

- Nette - jádro samotného frameworku
- Latte - „šablonovací“ systém
- Tracy - ladící systém

- Tester - testovací nástroj

3.3.1 MVP

Model MVP (Model View Presenter) rozděluje aplikaci na tři části - model, řídící prvek a pohled. V ideálním případě se model stará o doménovou logiku – což obnáší převážně komunikace s databází, přenášení dat, zajištění správnost dat. Řídící prvek obstarává aplikační logiku, což je shromažďování dat, reagování na akce uživatele a následná příprava dat pro pohled. A posledním je samotný pohled, který představuje, co bude zobrazeno klientovi na obrazovce.

Dodržování tohoto modelu není striktně dáno, každý programátor ho může využít podle sebe. Každý ho chápe také trochu jinak a rozpor mezi tím, co patří ještě do modelu a o co by se měly už starat řídící prvky, není ničím neobvyklým. U malých aplikací je častý jev spojení modelu a řídícího prvku do jedné vrstvy, protože několikanásobné členění u jednoduchých aplikací je kontraproduktivní.

Model MVC (Model View Controller) vznikl před modelem MVP a liší se zejména v názvu. Princip obou modelů je totožný, v některých literaturách se autoři často liší svými názory. MVP je často spojována právě s PHP Frameworky, kde je Controller označován jako Presenter. Rozdílnost názorů na oba modely spočívá v tom, o jakou část logiky se stará Controller a o jakou Presenter. Jako další odlišnost se dá považovat přístup k pohledu. V modelu MVC je přístup z pohledu do modelu přijatelný, v MVP by se naopak každá akce měla provádět skrze řídící Presenter.

3.3.2 Adresářová struktura

Nette se pro méně znalé vyskytuje v podobě sandboxu, který obsahuje základní adresářovou strukturu. Tato struktura může být přizpůsobena vlastním potřebám. Základní podobou jsou složky:

- app
- bin
- log
- temp
- tests
- vendor

- www

Složka app obsahuje ukázku jednoduché aplikace, která obsahuje uvítací stránku a příklad, jak je možné použít přihlašovací systém zabudovaný v samotném frameworku. Do složky bin se obvykle ukládají PHP kódy, které se netýkají přímo reálné aplikace, např. spouštěče pro programátora a PHP funkce. Do logu se ukládají html soubory popisující chybové hlášky, v případě, že je aplikace v produkčním režimu. Temp slouží pro ukládání dočasných souborů, kterými jsou zejména cachované stránky. Tests, jak název napovídá, slouží pro testy. Ve složce vendor se nachází jádro frameworku, Latte, Nette a ostatní knihovny, případně se zde mohou vyskytovat i části jiných PHP frameworků. A složka www slouží pro prezentování veřejných souborů, jako jsou obrázky, javascripty, kaskádové styly. Nachází se zde také soubor index.php, který odkazuje na zavaděč celé aplikace, zpravidla do složky app.

3.3.3 Cyklus aplikace

K lepšímu pochopení Nette Framework a jeho komponent je zapotřebí vědět, jak pracuje. Následujících několik bodů nastíní hlavní úkony, které Nette Framework vykonává před zobrazením stránky. Budeme uvažovat, že klient chce zobrazit stránku umístěnou na adrese localhost/nette_aplikace/www/novinky/detail. V tomto případě je projekt nazvaný nette_aplikace a je umístěn na místním zařízení.

- Klient/uživatel vyžádá složku www
- ve složce www se nachází soubor .htaccess, který překládá vloženou URL
- je spuštěn soubor index.php ve složce www
- index.php spouští zavaděč – bootstrap, který se nachází ve složce app
- zavaděč načítá celý framework, včetně aplikace
- spouští svůj vlastní překladač HTTP požadavků
- překladač vyhodnotí vloženou adresu, Presenterem dané aplikace je Presenter nazvaný „Novinky“ a pohledem, případně akcí, je „Detail“
- Je spuštěn Presenter Novinky, který se může dotazovat na modelovou vrstvu (databázi) a shromažďuje potřebná data pro pohled
- Pohled, kterým je „šablonovací“ systém Latte nahradí proměnné za reálné hodnoty a vytvoří HTML stránku
- Výsledný HTML kód je zaslán uživateli k zobrazení

Výše zmíněný postup je typický pro jednoduchou a základní aplikaci. Nette Framework je možné z velké části měnit a upravovat. Například způsob, jakým je URL překládána, může být naprosto individuální. Je možné nastavit jeden Presenter pro celou aplikaci jako defaultní a podle různých adres URL pouze odkazovat na jiné akce.

3.3.4 Latte

Latte je šablonovací systém Nette Framework. Jedná se o nadstavbu samotného HTML. Pomocí Latte je možné vytvářet šablonu pro více stránek, tzv. layout, samotné stránky, případně i emaily. Do šablony Latte jsou zpravidla předány od Presenteru proměnné. Všechny tyto proměnné, které pomocí Latte vypíšeme, jsou automaticky správně escapované a ošetřené proti XSS. Pomocí Latte je také možné vytvářet jednoduché podmínky a cykly, což HTML neposkytuje.

U webu se zpravidla mění pouze oblast obsahu, ostatní prvky jsou neměnné. O to se stará výše zmíněný layout – hlavička, načtení kaskádových stylů, hlavní menu apod. Latte umožňuje stránku rozdělit do bloků. Jeden z těchto bloků je zpravidla „content“, což je právě obsah stránky.

Příklad: Presenter vyžádá nějaká jména z databáze a předá je v podobě pole do šablony. Grafickým výstupem se očekává tabulka těchto jmen. Předpokládá se, že layout zajistí html značky, hlavičku a ostatní neměnné části. Pole získané databází obsahuje jména Petr, Pavel, Karel, Iveta. Samotnou obsahovou šablonou bude:

```
<table n:if="$jmena">
{foreach $jmena as $jmeno}
  <tr><td id="jmeno-{$iterator->counter}">{$jmeno|capitalize}</td></tr>
{/foreach}
</ul>
```

4 Vlastní práce

4.1 Prototyp Nette aplikace

V druhé části této práce bude představen prototyp aplikace postavené na Nette Framework, analýza tohoto prototypu a návrh ochrany nejzávažnějších bezpečnostních nedostatků. Prototyp je téměř identický s ukázkovou aplikací Nette Quickstart[3]. Tato aplikace byla umístěna na webový hosting PHP5[4] pod dočasnou adresou http://ultimaonline.php5.cz/www/ k dosažení reálných výsledků při analýze zranitelnosti.

Jedná se o jednoduchý „blogerský“ web. Na úvodní straně jsou zobrazeny autorovy články, na které je možné kliknout a zobrazí se samotný článek s názory čtenářů a možností přidat svůj vlastní příspěvek. Články a příspěvky jsou uloženy do databáze MySQL. Blog také obsahuje možnost přihlášení pro samotného „blogera“, případně dalšího autora článků. Po přihlášení je možné publikovat nový článek.

4.2 Analýza webové aplikace

Možností, jak provést analýzu bezpečnosti webové aplikace je několik. V této části budou představeny dva způsoby. Prvním je analýza pomocí nástroje, který zveřejnilo uskupení OWASP a nazývá se OWASP ZAP. A druhým je tradiční způsob – analýza zdrojového kódu.

4.2.1 Analýza pomocí nástroje OWASP ZAP 2.4.3

ZAP je ideální nástroj pro programátory, developery, či testery začínající, ale může být velice užitečným pomocníkem i pro profesionály. Jedná se o software napsaný v Javě, je tedy multiplatformní.

S programem ZAP je možné skenovat weby třemi hlavními způsoby:

- automaticky aktivně
- automaticky pasivně
- manuálně

[3] https://doc.nette.org/cs/2.3/quickstart
[4] http://www.php5.cz/

K aktivnímu automatickému skenování není potřeba nic jiného, než adresu webu, na kterou chceme zaútočit. Po spuštění skenování proběhne mapování stránek – což zajišťuje takzvaný Web Crawler – webový pavouk. Po zmapování probíhá samotné skenování, u velice jednoduchých aplikací v řádu několika sekund, u středních webů je to otázka minut a u velkých až několik hodin, či dní v závislosti na použitém hardwaru útočníka a složitosti cílové webové aplikace. V případě, že je potřeba skenovat weby zároveň při surfování, ZAP nabízí možnost Plug-n-Hack. Jedná se o doplněk do webového prohlížeče zajišťující automatické přeposílání adres do ZAP, který průběžně skenuje cíle. Manuální způsob se liší od automatického nutností, případně možností nastavit, jakým způsobem má ZAP prohledávat, které adresy má ignorovat (zpravidla stránku logout a duplikáty), nebo jak má daný útok provést apod.

Pro výše uvedený prototyp je z těchto tří možností nejvhodnější první způsob, a proto bylo provedeno automatické skenování adresy http://ultimaonline.php5.cz/www/.

Skenované útoky

Některé typy útoků nelze pomocí softwarových nástrojů zautomatizovat, tudíž není ani možné odhalit všechny nedostatky webové aplikace. Ze zkušenosti je možné usoudit, že dokáže simulovat útoky:

- Injektováním
- XSS
- částečně nezabezpečenou konfiguraci.

Pokud ZAP nalezne potenciální nedostatky v bezpečnosti, vypíše jednotlivé hrozby do položky „Alerts“. Závažnost samotných útoků je přiřazena podle vlastních metodik OWASP, podle které byl i vytvořen list TOP10, z nichž 5 nejzávažnějších bylo popsáno v teoretické části. Rizika mohou být:

- Vysoká
- Střední
- Nízká
- Informační
- False Positive

Na základě výsledku skenování je nutné provést analýzu zdrojového kódu a vyšetřit danou hrozbu.

4.2.2 Analýza zdrojového kódu

Analýzu zdrojového kódu lze obtížně zautomatizovat. Existují placené penetrační nástroje, které jsou schopny prohledat částečně kód aplikace. Přesto je nutno výsledky testů těchto nástrojů prověřit ideálně odborným testerem, či programátorem.

Sledované útoky

Tester/programátor se má v tomto případě zaměřit na čtyři útoky:

- Injektování
- XSS
- Nezabezpečené přímé odkazy na objekty
- Nezabezpečená konfigurace

4.3 Počáteční stav

Na prototypové akci bylo provedeno automatické skenování pomocí softwaru OWASP ZAP. Jak je vidět na obrázku 4, celkově bylo nalezeno 117 hrozeb, z nichž dva útoky byly označeny jako potenciální XSS a další dva jako SQL Injektování.

Obrázek 4: Seznam potenciálních útoků na webovou aplikaci - zdroj: OWASP ZAP 2.4.3.

4.4 Návrh ochrany aplikace

Nyní jsou zmapované možné útoky a je potřeba tyto hrozby odstranit, případně, pokud se jedná o falešný, či planý poplach, je nutné situaci prověřit a vyvrátit.

4.4.1 SQL Injektování

Pokud je použita databáze, je nutné se zaměřit v případě bezpečnosti na samotného interpreta. V tomto případě se jedná o API[5] Nette/Database, vlastní třídu Nette Framework, která je nadstavbou standardní PDO[6] zabudované v PHP. Třída Nette/Database představuje bezpečné prostředí pro komunikaci s databází, v případě, že je správně použita.

Postup při návrhu ochrany

Byly nalezeny dvě hrozby injektování v oblasti přihlašování. Po analýze zdrojového kódu bylo zjištěno, že přihlašovací systém vůbec neinteraguje s databází a údaje nutné pro autentizaci jsou uloženy v konfiguračním souboru Nette Framework. Tyto dvě hrozby lze označit jako falešné.

Pro důkladnou ochranu je potřeba prozkoumat každou část kódu, která komunikuje s databází. Jsou jimi čtyři situace – zobrazování článků, zobrazování komentářů, vkládání článků a vkládání komentářů.

Návrh ochrany

Aby se dal kód považovat za bezpečný z hlediska injektování, je nutné používat ve všech případech:

1. Připravené příkazy
2. Vázané proměnné
3. Uložené procedury
4. Statické dotazy

V případě, že jsou data vkládána do databáze a z databáze požadována a vkládána do kódu samotných stránek, musí být řádně escapována[7].

[5] Rozhraní pro programování aplikací
[6] Univerzální rozhraní pro komunikaci s databází. Nativně jej obsahuje PHP.
[7] Procesem escapování se rozumí nahrazení znaků, které mají v daném kontextu speciální význam za jiné odpovídající sekvence

4.4.2 XSS

Jedná se o triviální způsob útoku, přesto je mnoha webovými stránkami zranitelný.

Postup při návrhu ochrany

OWASP ZAP zaznamenal dva výskyty tohoto útoku. V obou případech se jedná o reflektovaný způsob, což znamená, že uživatelské vstupy byly zobrazeny např. v podobě chybových hlášek. Pokud vstupy obsahují útočníkům kód, který není ošetřen, může se vykonat a útok být úspěšný.

Při vývoji aplikace je zapotřebí zkontrolovat zejména každý výstup a řádným způsobem ho ošetřit. V případě Nette Framework tuto činnost zajišťuje šablonovací systém Latte. Latte disponuje unikátní technologií Context-Aware Escaping, což znamená, že dokáže rozeznat, v jaké části dokumentu a kódu se nachází. Na základě toho zvolí správný způsob ošetření automaticky.

Návrh ochrany

Útokům typu cross-site scripting se dá zabránit dvěma způsoby. Prvním je striktní validace vstupů a druhou možností je správné escapování výstupů. K zajištění větší bezpečnosti se používají oba způsoby.

4.4.3 Nezabezpečené přímé odkazy na objekty

Nejčastějším případem nezabezpečených přímých odkazů je dynamický obsah stránky. Obsah stránky je zpravidla měněn parametrem GET proměnné a přímo odkazuje na dané soubory.

Postup při návrhu ochrany

Útoky jsou snadno simulované testovacími nástroji. V testovaném prototypu OWASP ZAP nenalezl ani jeden případ tohoto druhu útoku. Je nutné brát také v potaz případy, kdy má uživatel částečnou pravomoc (např. po přihlášení) a získává tím možnost setkat s parametry, které nejsou správně validovány. Proto je nutné testovat i případy přihlášených uživatelů a výši jejich pravomocí.

Návrh ochrany

Jedinou ochranou před útoky na přímé odkazy je vytvoření tzv. map. Tato mapa by měla obsahovat dvojice identifikace objektu a samotný objekt. Přístup k daným objektům je tak zajištěn nepřímo přes identifikátory.

4.4.4 Nezabezpečená konfigurace

Nezabezpečených webových serverů a aplikací je zpravidla více, než těch zabezpečených. Pravděpodobně z důvodu více možností, jak je vystavit riziku.

Postup při návrhu

Útoky na nezabezpečenou konfiguraci lze pouze z části zachytit testovacími nástroji. Je zapotřebí si ujasnit, jaký webový server používáme, jakou verzi, případně na jakém operačním systému běží, poté prostředí, programovací jazyk, či framework, pokud je nějaký použit a knihovny.

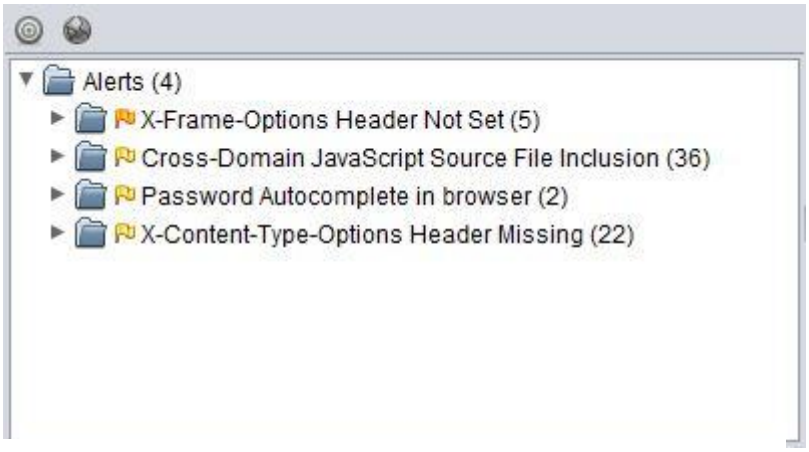
Návrh ochrany

Nelze jednoznačně zajistit ochranu těchto útoků, protože mohou být cíleny na bezpečnostní chybu samotného serveru, který ovlivní pouze vývojáři daného softwaru. Přesto je důležité:

1. zajistit nejaktuálnější verze webových serverů, frameworků, pluginů, knihoven, apod.
2. odstranit výchozí hodnoty – hesla, uživatelská jména, klíčová slova
3. neodhalovat chybové hlášky obsahující informace o aplikaci
4. vypnout directory listing
5. vypnout debug mode, pokud je aplikace v ostré verzi přístupná veřejnosti
6. nespouštět další neověřené služby na serveru

5 Výsledky

Po provedení všech zmiňovaných opatření na ochranu byl spuštěn druhý test. Jak je možné vidět na obrázku 5, nejzávažnější hrozby byly odstraněny upravením zdrojového kódu prototypu, případně byly prověřené hrozby označeny jako False Positive.



Obrázek 5 - Seznam útoků po provedení bezpečnostních opatření

5.1 Zhodnocení ochrany Nette Framework na jednotlivé útoky

SQL Injektování

Komunikace s databází byla zajištěna skrze API Nette/Database. V jednom případě bylo použito tradičního SQL dotazu s volným parametrem, kterou nabízí Nette/Database. Tento nedostatek byl zjištěn po důkladné analýze kódu, OWASP ZAP tuto slabinu neodhalil. V ostatních případech byly použity ostatní metody třídy Nette/Database, pomocí kterých se skládal samotný SQL dotaz.

Nelze se tedy spolehnout 100% na framework, je nutná znalost problematiky. Pokud je programátor neznalý, může vytvořit velice závažnou bezpečnostní chybu.

XSS

Jak bylo popsáno, proti XSS se dá bránit dvěma způsoby. Nette, resp. Latte zajišťuje maximální bezpečnost proti typu tohoto útoku v podobě správného escapování. Tkví to také z faktu, že šablony vytváří zejména grafici, stylisté a kódeři, nikoliv programátoři. Proto se u nich nepředpokládá znalost webových útoků.

V případě ochrany proti XSS je možné se na framework jednoznačně spolehnout.

Nezabezpečené přímé odkazy na objekty

Odkazy jsou v Nette Framework řešeny skrze vlastní router. Každý požadavek je tedy směřován k objektu, podle toho, jak je samotná tzv. „routa“ (směrnice) definována.

V základní podobě lze považovat framework za spolehlivý. V případě nedostatečné znalosti při úpravě směrnic je snadné vytvořit bezpečnostní díru.

Nezabezpečená konfigurace

Nezabezpečená konfigurace se týka operačního systému, webové serveru, programovacího jazyka, frameworku, knihoven a další doplňků. Pokud je použit základní sandbox, tak ten zajišťuje ve výchozím nastavení debug mod pouze na místním zařízení a ve všech složkách, kromě veřejné /www je zakázáno directory listing.

Útoky na nezabezpečenou konfiguraci se týkají celého software a framework je pouze zlomkem možných cílů. V rámci možností je ochrana těchto útoků řešena maximálně. Teoreticky možné zlepšení by se mohlo týkat automatického stahování aktuálnější verze frameworku, prakticky je však tato funkce jen velice obtížně realizovatelná, už jen z důvodu, že novější verze nejsou často kompatibilní s verzí předešlou.

6 Závěr

V bakalářské práci byla popsána problematika bezpečnosti webových aplikací. Existuje nespočet možností, jak prolomit obranu webových aplikací, proto byly vybrány pouze ty nejzávažnější útoky, které mohou mít katastrofální dopad v podobě odcizení citlivých dat, finančních prostředků nebo ztráty stávajících, či potenciálních uživatelů.

První kapitola teoretické části charakterizuje způsoby a prostředky, kterými lze vytvořit webová aplikace. Jsou zde popsány technologie klientské strany a strany serveru se zaměřením na prostředí, ve kterém je vytvořen prototyp pro tuto práci.

V další části je představen samotný Nette Framework, který je stavebním kamenem analyzovaného prototypu. Jsou zde informace o částech frameworku, principu fungování a návrhový vzor, kterým se řídí programování aplikací. Část je zejména věnována šablonovacímu systému Latte, který je součástí frameworku.

Poslední kapitola teoretické části se věnuje vybraným webovým útokům. Je použit seznam nejzávažnějších útoků, který vytváří organizace OWASP každé tři roky. Pro tuto práci byl použit poslední zveřejněný, z roku 2013. Každý útok obsahuje kromě principu také příklad použití.

Zmiňovaný prototyp je v praktické části analyzován. Analýza probíhala dvěma způsoby – penetračním testovacím softwarem a osobní analýzou kódu. Následně je navržena ochrana na každý typ ze zaměřených útoků. Samotný návrh má obecné uplatnění a lze tedy použít v jakékoliv jiném vývojovém prostředí.

Program OWASP ZAP, pomocí kterého byl proveden sken potenciálních útoků, se osvědčil jako uspokojivý nástroj pro testování menších, či středních webů s menším důrazem na bezpečnost. Ovládání je intuitivní a jednoduché a široké spektrum funkcí uspokojí většinu uživatelů.

Nette Framework se osvědčil jako velmi dobrý nástroj pro vývoj webových aplikací. Často je zmiňováno, že vyznávaným principem Nette Framework je „less code, more security“ (méně kódu znamená více bezpečnosti) a v naprosté většině případů je možné dát tomuto tvrzení za pravdu. Obvyklým způsobem vzniku bezpečnostních děr jsou stovky, či tisíce řádků kódu, ve kterých je téměř nemožné se vyznat. Samotný framework je naprogramován čitelně a jsou k tomu tak vedeni i programátoři.

Ačkoliv jsou bezpečnostní rizika zdůrazňována vývojáři frameworku a komunitou jsou doporučeny co nejjednodušší způsoby implementace kódu při různých situací, není neobvyklé setkat se s takzvaným špagetovým kódem[8]. Už jen vznik těchto nesrozumitelných kódů je důkazem toho, že kvalita a s ní i bezpečnost aplikace závisí zejména na znalosti programátora. Nette Framework je z velké většiny schopen zajistit ochranu automaticky, případně obsahuje takové API, které ochranu zabezpečí. Přesto zůstává zodpovědnost pouze na programátorovi, jestli výsledná aplikace bude, či nebude bezpečná.

[8] Nesrozumitelný, velice rozsáhlý a neudržovaný kód

7 Seznam použitých zdrojů

MELONI, Julie. *PHP essentials*. 2. vydání. Boston, MA: Premier Press, 2003. ISBN 9781931841344.

Kunder, Maurice de. 2015. The size of the World Wide Web (The Internet) [online] 2012. [cit. 19. Listopad 2012.] http://www.worldwidewebsize.com/.

PEJŠA, Jan. Co je Cross-site scripting jak mu předcházet. *Zdroják* [online]. Devel.cz Lab s.r.o., 2009, , 1 s. [cit. 2016-02-17]. ISSN 1803-5620. Dostupné z: https://www.zdrojak.cz/clanky/co-je-xss-jak-mu-predchazet/

FERSCHMANN, Petr. Bezpečnost na webu – přehled útoků na webové aplikace. *Zdroják* [online]. Devel.cz Lab s.r.o., 2008, , 1 s. [cit. 2016-02-21]. ISSN 1803-5620. Dostupné z: https://www.zdrojak.cz/clanky/prehled-utoku-na-webove-aplikace/

DESCHATRES, Solange. 5 Web Attacks Targeting Your Data. In: *Symantec Connect* [online]. Symantec Corporation, 2014 [cit. 2016-03-06]. Dostupné z: http://www.symantec.com/connect/blogs/5-web-attacks-targeting-your-data

Web server attacks. In: *Communitic International* [online]. Raoued: Communitic International, 2013 [cit. 2016-02-16]. Dostupné z: http://ccm.net/contents/16-web-server-attacks

DEDE, David. Most Common Attacks Affecting Today's Websites. In: *SUCURI Blog* [online]. Kalifornie: Sucuri, Inc., 2016 [cit. 2016-01-06]. Dostupné z: https://blog.sucuri.net/2014/11/most-common-attacks-affecting-todays-websites.html

CORNELL, Dan. Getting Started with ZAP and the OWASP Top 10: Common Questions. In:*Denim group* [online]. San Antonio: Denim Group, 2016 [cit. 2016-02-12]. Dostupné z: http://www.denimgroup.com/blog/denim_group/2015/07/getting-started-questions.html

Usage of server-side programming languages for websites. *W3Techs - World Wide Web Technology Surveys* [online]. W3Techs, 2016 [cit. 2016-02-06]. Dostupné z: http://w3techs.com/technologies/overview/programming_language/all

GRUDL, David. *Velestručné testování presenterů v Nette* [online]. 2013. [cit. 2016-02-08] Dostupné z: http://phpfashion.com/velestrucne-testovani-presenteru-v-nette

What is the maximum length of a url. *Stack Overflow* [online]. [cit. 2016-02-08]. Dostupné z: http://stackoverflow.com/questions/417142/what-is-the-maximum-length-of-a-url

OWASP Zed Attack Proxy Project. *OWASP* [online]. New Jersey: Wikimedia Foundation, Inc., 2016. Dostupné také z: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

Web Applications: What are They? What of Them? In: *Acunetix* [online]. Londýn: Acunetix, 2016 [cit. 2016-02-05]. Dostupné z: http://www.acunetix.com/websitesecurity/web-applications/

LARSEN, Rob. *Beginning HTML & CSS*. 1. vydání. Indianapolis, Ind.: Wiley, 2013. Programmer to programmer. ISBN 9781118340189.

OWASP TOP 10 – 2013. *OWASP* [online]. , 22 s. [cit. 2016-02-12]. Dostupné z: https://www.owasp.org/images/f/f3/OWASP_Top_10_-_2013_Final_-_Czech_V1.1.pdf

Nette Quickstart. *Nette Framework* [online]. Praha: Nette Foundation, 2016 [cit. 2016-02-08]. Dostupné z: https://doc.nette.org/cs/2.3/quickstart/getting-started

8 Přílohy

8.1 Seznam použitých příloh v textu

8.2 Přílohy

K této práci jsou přiloženy zdrojové kódy prototypu před provedením ochrany a po samotném refaktoringu, jejichž jsem výhradním autorem.