Bezpečnost e-shopových frameworků

Bc. Filip Hasík

Diplomová práce
2014

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Filip Hasík**
Osobní číslo:	**A12418**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Počítačové a komunikační systémy**
Forma studia:	**prezenční**
Téma práce:	**Bezpečnost e-shopových frameworků**
Téma anglicky:	**The Security of E-Shop Frameworks**

Zásady pro vypracování:

1. **Popište nejčastěji používané frameworky pro tvorbu e-shopových portálů.**
2. **Analyzujte nejčastěji používané frameworky pro tvorbu e-shopových portálů.**
3. **Proveďte testy vybraných frameworků na bezpečnostní zranitelnosti.**
4. **Navrhněte vhodná opatření pro zkvalitnění bezpečnostních řešení.**
5. **Vypracujte seznam doporučení pro implementaci jednotlivých frameworků pro produkční prostředí.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: tištěná/elektronická

Seznam odborné literatury:

1. DOSEDĚL, Tomáš. Počítačová bezpečnost a ochrana dat. Vyd. 1. Brno: Computer Press, 2004, 190 s. ISBN 80-251-0106-1.
2. JAŠEK, Roman. Informační a datová bezpečnost. Vyd. 1. Zlín: Univerzita Tomáše Bati ve Zlíně, 2006, 140 s. ISBN 80-731-8456-7.
3. KLÍMA, Vlastimil. Archivy publikací o kryptologii a počítačové bezpečnosti [online]. 1993-2013. Dostupné z: http://cryptography.hyperlink.cz/.
4. BERGMANN, Sebastian, Stefan PRIEBSCH a Karol PRZYSTALSKI. Real-world solutions for developing high-quality PHP frameworks and applications. Indianapolis, Ind.: Wiley, 2011, 378 s. ISBN 978-0470872499.
5. GRUDL, David. Začínáme s Nette Framework. Zdroják [online]. 2009. Dostupné z: http://www.zdrojak.cz/serialy/zaciname-s-nette-framework/.
6. BÖHMER, Marian. Zend Framework: programujeme webové aplikace v PHP. Vyd. 1. Brno: Computer Press, 2010, 416 s. ISBN 978-80-251-2965-4.

Vedoucí diplomové práce:	Ing. David Malaník, Ph.D. Ústav informatiky a umělé inteligence
Datum zadání diplomové práce:	7. února 2014
Termín odevzdání diplomové práce:	27. května 2014

Ve Zlíně dne 7. února 2014

prof. Ing. Vladimír Vašek, CSc.
děkan

L.S.

prof. Ing. Karel Vlček, CSc.
ředitel ústavu

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně .....

podpis diplomanta

ABSTRAKT

Cílem této diplomové práce je popsat problematiku e-shopových frameworků a jejich odolnost vůči bezpečnostním zranitelnostem.

Teoretická část popisuje jednotlivá řešení, která jsou v systémech využita. Jedná se o základní pojmy, pro počítačově zdatné jedince jsou zásadní. Také jsou sepsány klady a zápory jednotlivých PHP frameworků a samotných e-shopových řešení.

V praktické části bylo potřeba vytvořit metodiky vedoucí k nalezení správné cesty a vytvoření testů, které by mohly reálně vyzkoušet odolnost na bezpečnostní zranitelnosti. Součástí testů je závěrečná zpráva, která udává, jakých hrozeb je třeba se obávat a jak se proti nim bránit a naopak které útoky jsou spíše nepotřebné.

Klíčová slova: PHP, MySQL, webhosting, bezpečnost, e-shopové frameworky, testování

ABSTRACT

The aim of the Diploma Thesis is to describe problems of e-shop´s frameworks and their resistance to safety vulnerabilities. The theoretical part describes individual solutions which are used in systems. There are basic terms which are important for computer specialists. There are also written the pluses and minuses of PHP frameworks and e-shop´s solutions.

In the practical part was necessary to create methodology which leads to finding the right way and creating tests which could try resistance of safety vulnerabilities. There is included the final report which states threats we have to be worry about and not.

Keywords: PHP, MySQL, webhosting, security, e-shop´s frameworks, testing

Touto cestou bych rád poděkoval Ing. Davidu Malaníkovi, PhD., vedoucímu diplomové práce, za jeho odborné vedení a cenné rady při tvorbě této práce.

Také chci poděkovat své rodině a přítelkyni za jejich trpělivost a podporu při studiu na Univerzitě Tomáše Bati.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Informatika představuje nejrychleji se rozvíjející obor celého průmyslu. A mezi internetovými aplikacemi patří e-shop v současné době mezi nejčastější. E-shopy slouží zákazníkovi k vyhledávání zboží a jeho objednání. Existuje několik způsobů, jak si e-shop vytvořit. Buď si jej celý naprogramovat, nebo jej koupit od někoho jiného, nebo využít open source bezplatné řešení.

Právě třetím typem se zabývám ve své práci. A popisuji jeho plusy a mínusy z hlediska bezpečnosti. Dokud se nic nestane, tak na bezpečnost obecně nikdo nemyslí. Ale nejúčinnějším a nejlepším řešením je právě veškerým nepříjemným situacím raději předcházet a připravit se na ně. V diplomové práci se na e-shop dívám právě z tohoto hlediska, a to jak jej nejlépe zabezpečit, aby útočník nemohl provést nic škodlivého.

K provádění testů odolnosti vůči bezpečnostním chybám existuje nepřeberné množství aplikací a utilit. Zaměřil jsem se na tři taková řešení, která podle mne nejlépe dokáží otestovat, jak na tom jednotlivé frameworky jsou. Využil jsem pěti nejznámějších frameworků a snažil se upozornit na veškeré nesrovnalosti, které by měly být vyřešeny.

Pojmy jako PrestaShop, VirtueMart nebo Magento jsou v tomto odvětví velmi často skloňovány díky svým kvalitám v oblasti prodeje zboží. Rád bych vyzkoušel, zda budou tak úspěšné také při prováděném testování. Na testování naopak vyzkouším známé pojmy z oblasti útoků, jako SQL Injection nebo Cross site Scripting.

Cílem mé práce je vyhodnotit míru zabezpečení jednotlivých e-shopových frameworků. Na všechny e-shopy budu používat stejné testy a následně budu hodnotit úspěšnost ochrany jednotlivých frameworků. Testování bude probíhat formou simulovaných útoků, které mohou být použity při běžném provozu. Výsledkem bude porovnání jednotlivých testů, návrh zabezpečení nalezených problémů a výsledné doporučení e-shopového systému, který především dokáže na dané chyby správně reagovat.

I. TEORETICKÁ ČÁST

1 INTERNETOVÁ BEZPEČNOST

O nebezpečných internetových útocích slyšíme každý den. I proto se stává bezpečnost internetu a bezpečnost práce na internetu stále více sledovanou oblastí. Rostoucí počty útoků souvisí s rostoucím počtem elektrických zařízení připojovaných k internetu, kterými se obklopujeme. Neustále tak stoupají požadavky na spolehlivost a bezpečnost připojení, ale právě proto je splnění obou požadavků čím dál více náročné.

V ideálním stavu by měly všechny internetové služby fungovat tak, že veškerá data, která přes internet jakýmkoliv způsobem posíláme, by měla v neporušené podobě dojít přesně tam, kam chceme. Pokud vidíme, že nějaké stránky mají výpadek, nezajímá nás, zda je to způsobeno chybou v implementaci programovacího jazyka nebo následkem napadení webu skrze počítačové viry. Nikoho také nezajímá, zda neúmyslné zviditelnění našich citlivých údajů na síti je způsobeno výpadkem některé služby nebo útokem hackera. Přáním všech je mít svá data pod kontrolou.

Proto je důležité si uvědomit staré známé pravidlo „Řetěz je vždy tak silný, jako jeho nejslabší článek“. U počítačové bezpečnosti toto heslo platí dvojnásob. Není důležité, jak kvalitní máme heslo k počítači, když jej napíšeme na papírek a nalepíme na monitor. Vždy je nutné považovat bezpečnost jako celkový neustále trvající proces, ne jako jednorázovou nedůležitou akci. A vždy je dobré se problém dozvědět dříve, než nás plně dostihnou jeho důsledky. Testování možných zranitelností je tou správnou cestou pro zesílení nejslabších článků a tím udržování celistvého bezpečnostního procesu.

2 ZÁKLADNÍ TECHNOLOGIE

V této kapitole budou vysvětleny základní pojmy, se kterými se budeme potkávat v průběhu celé práce. Také budou popsány a vysvětleny jednotlivé technologie, které jsou použity jak pro tvorbu e-shopových frameworků. tak pro jednotlivé bezpečnostní testy.

2.1 PHP

Většina existujících řešení v této oblasti je vytvořena pomocí PHP. Proto zde vysvětlím, co si pod zkratkou PHP vůbec představit.

PHP je zkratka pro PHP: Hypertext Preprocessor, česky „PHP: Hypertextový preprocesor“, původně Personal Home Page. PHP je skriptovací programovací jazyk. Je určený především pro zpracování dynamických požadavků na serveru. Díky tomu, že je veden jako open source, tudíž zdarma, je velmi rozšířen.

Společně s ostatními programovacími jazyky ASP, ASP.NET, Java a Perl bývá označován jako server-side jazyk. Jak takové jazyky pracují? Uživatel ze svého počítače pošle požadavek na internetovou stránku. Pokud je v ní obsažen PHP kód, je tento kód serverem zpracován a následně začleněn do HTML stránky. Poté je, již jako statický HTML kód, zaslán zpět uživateli.[1]

Je především využíván pro tvorbu dynamických webových aplikací například ve formátu HTML, XHTML či WML, ale lze jej využít i pro tvorbu desktopových a konzolových aplikací. Pro desktopové použití existuje kompilovaná forma jazyka. Lze jej rozšířit velkým množstvím knihoven.[1]

První verze PHP byla vytvořena v roce 1995. Do povědomí se dostal až po vytvoření verze PHP 3 v roce 1998 a začal se hojně rozšiřovat. Nejnovější verze PHP 5.5 je z dubna 2014. Jazyk PHP je možné používat napříč různými operačními systémy, ale je vytvořen především pro použití v linuxu. V operačních systémech od Microsoftu je jeho použití složitější a může se nepatrně lišit. [1]

Jazyk PHP prošel během svého života prudkým vývojem a postupně se z něj stal jazyk, který podporuje procedurální i objektové programování. Dalším důvodem jeho oblíbenosti je jeho velká intuitivnost a jednoduchost použití. I přesto, že je optimalizován pro web, lze i velmi složité věci napsat poměrně krátkým úsekem kódu.

Díky velmi širokému využití má obrovskou komunitu uživatelů, kteří pravidelně přispívají svými „vychytávkami“ a nejrůznějšími návody. Díky tomu se i z naprostého laika může stát poměrně rychle znalý uživatel. Kromě nejrůznějších uživatelských dokumentací a tutoriálů můžeme použít i oficiální referenční příručku. Jedinou nevýhodou může být její lokalizace pouze v anglickém jazyce.

2.2 SQL

Anglickou zkratku SQL rozšifrujeme jako Structured Query Language, do češtiny je přeložena jako strukturovaný dotazovací jazyk. Jde o jednoduchý jazyk, který lze používat velmi intuitivně. Veškeré příkazy byly vytváření s ohledem na přirozený jazyk. Není tak potřeba si pamatovat speciální výrazy, většinou stačí si daný příkaz pouze přeložit do angličtiny.

Pro práci s databázemi je téměř výhradně používán právě SQL. Bývá označován jako standardizovaný dotazovací jazyk, který se používá pro práci s daty v databázích. Data v databázích jsou ukládána do tabulek. Správu jednotlivých databází pak spravuje relační systém, např. MySQL.

2.3 MYSQL

Tvůrcem MySQL je švédská firma MySQL AB, v současnosti pod názvem Sun Microsystems. Hlavními autory jsou Michael Widenius a David Axmark. MySQL je open source, tudíž jej lze sehnat zdarma. Ale existuje i placená komerční verze. Díky tomu patří mezi průkopníky dvojího licencování.[2]

MySQL je databázový systém, jehož úkolem je práce s daty, jejich ukládání, vyhledávání a správa. Databázový systém si můžeme představit jako množinu prvků, která definuje databázi a veškerou manipulaci s ní. MySQL je multiplatformní databáze. Komunikace s ní probíhá – jak už název napovídá – pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními. [2]

Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, je využíván velmi hojně. Jeho použití je ideální jak při lokálních instalacích, tak u webhostingů. Především u freehostingových serverů je téměř jisté, že pracují s MySQL. Podobná je situace i u lokálních řešení. Při instalování většiny lokálních webserverů nara-

zíme na kombinaci MySQL spolu s PHP a Apache. To platí jak u linuxových verzí (např. LAMP - Linux, Apache, MySQL, PHP), tak u řešení pro Windows (např. WAMP).

Vývoj MySQL byl vždy zaměřen především na rychlost, což bylo vyváženo určitými zjednodušeními. Například až v posledních verzích jsou doplňovány vylepšení, která byla úmyslně vynechávána - podpora pohledů, triggerů, transakcí a ukládání procedur. Tyto nedostatky však neměly velký vliv na vývoj webových databázových aplikací.

Pro jednoduchou správu databází je vytvořen grafický program phpMyAdmin. Také tato utilita je neustále vylepšována a s každou novou verzí jsou opraveny veškeré chyby a přidány nové funkce. Program umožní spravovat databázi, vytvářet kompletní strukturu. Je volně ke stažení a bez problémů implementovatelný do ostatních serverových aplikací. Pro začátečníky je vhodné grafické „klikací“ prostředí, kdy lze většinu akcí vykonat jednoduchým kliknutím tlačítka. Veškeré takto provedené příkazy jsou vypisovány na obrazovce, lze si je tak jednoduše opakovat a lépe si je zapamatovat.

2.4 JavaScript

JavaScript je multiplatformní, interpretovaný, objektově orientovaný programovací skriptovací jazyk. Jádro jazyka je vloženo do webových prohlížečů a bývá libovolně rozšiřováno. Důležitou součástí je přidání objektů, které reprezentují okno prohlížeče a také jeho obsah. Obvykle jsou jím ovládány interaktivní prvky grafického uživatelského rozhraní, např. textová pole nebo tlačítka. Také nám umožňuje tvořit animace a různé obrázkové efekty. Díky těmto možnostem se webová stránka stává dynamickou. Na stránkách mohou být spuštěny nejrůznější programy komunikující s uživatelem, které dynamicky vytváří obsah HTML nebo řídí celý prohlížeč. [3]

Vzhledem k tomu, že JavaScript je klientský skript, není nutné při vysílání požadavků kontaktovat server, veškerou práci zajišťuje samotný prohlížeč. Klientský skript pracuje tak, že požadavek se s celou stránkou odesílá do prohlížeče (na klienta) a tam je vykonáván. Alternativou ke klientským skriptům jsou serverové skripty, které jsou prováděny na serveru, a do prohlížeče již přijde kompletní výsledek, který je pouze zobrazen. [3]

Syntakticky je JavaScript řazen do rodiny jazyků C, C++ a Java. JavaScript je však samostatný standardizovaný programovací jazyk. Rozdíly můžeme vidět především v typu kompilace jednotlivých jazyků. Jazyky C a C++ jsou kompilované a Java je před interpretací také kompilována do bytového kódu, což znamená, že u těchto jazyků je vždy po-

třeba překladač, který přeloží zdrojový kód do strojového kódu. Naproti tomu JavaScript je čistě interpretovaný jazyk, u něhož je zdrojový kód prováděn interpretem, v tomto případě prohlížečem. [3]

Z hlediska fungování celého JavaScriptu je patrné, že by mohlo být velmi jednoduché skrze něj proniknout a vykonat nebezpečné útoky. Proto jsou přímo v kódu určitá opatření, která alespoň částečně některá rizika odfiltrují. K základním opatřením patří, že JavaScript na straně klienta neumožňuje čtení ani zapisování souborů. V případě umožnění volného čtení i zapisování by nebyl problém jednoduchým programem znehodnotit obsah pevného disku. Rovněž nepodporuje práci se sítí, kromě jedné výjimky: může donutit prohlížeč k načtení libovolného URL.

2.5 Cookies

Cookies jsou malé textové soubory, které jsou uloženy přímo v počítači uživatele. Webový prohlížeč k nim má umožněn přístup a může si tak z těchto souborů stahovat uložená data. Tyto soubory byly vytvořeny pro usnadnění užívání různých webových aplikací, mohou však obsahovat i citlivé informace. Velmi často obsahují uživatelovy identifikační údaje a především historii surfování po internetu. Často tak jsou využívány pro cílenou reklamu. [4]

Jak může dojít ke zneužití? Pokud je na stránce vložen určitý škodlivý kód, například upravený javascriptový kód s pomocí XSS vsunutí, umožní přístup k tomuto souboru a veškeré informace tak poskytnout útočníkovi. Ten je pak může libovolně zneužít. Další možností, jak získat citlivá data je použití snifferů, neboli analyzátorů odposlouchávaných dat. Pokud server nepoužije pro přenos dat šifrování pomocí SSL, soubory jsou posílány po síti v čitelné podobě. Mohou pak být jednoduše odposlechnuty. Není možné přes tyto soubory provádět přímo útoky na PC, díky tomu nejsou cookies v současnosti považovány za mimořádnou hrozbu. [4]

Ale pro ty, kteří dbají na své soukromí, je důrazně doporučováno používání cookie souborů zakázat. Může však nastat problém, protože některé webové stránky je vyžadují, pokud by bylo jejich používání zakázáno, budou pro uživatele tyto stránky nedostupné. Jako vhodnější zabezpečení je považováno nastavení prohlížeče tak, aby při každém vypnutí obsah cookies vymazal.

3 E-SHOPOVÁ ŘEŠENÍ

Existují desítky e-shopových systémů pro jednoduché vytvoření vlastního obchodu. Řadíme je do dvou kategorií - placené a bezplatné. Bezplatné řešení jsou většinou licencovaná jako open source. Touto kategorií se budu dále zabývat.

3.1 Open source řešení

V práci jsem využíval e-shopové systémy, které jsou k dispozici v rámci open-source licence. Co vůbec open source licence znamená? Open source software je šířen pod licencí, která zaručuje technickou i legální dostupnost zdrojového kódu. Pokud uživatel dodrží určité podmínky, je mu umožněno se zdrojovým kódem pracovat, upravovat a šířit. Dříve byl open source vnímán pouze jako školní projekt, ale s postupem času se z něj podle analytiků stal hlavní inovátor v oblasti software a po dlouhé době tak vystřídal proprietární aplikace. [5]

Téměř všechen open source je zároveň i svobodný software. Většina lidí však používá pro obě kategorie první pojem. I když jsou rozdíly mezi nimi minimální, stále tam jsou. Tyto výjimky najdeme především v akceptování některých licencí.

3.1.1 Svobodný software

V této kapitole vysvětlím základní pravidla pro určení svobodného softwaru. Aby byl takto definován, musí splňovat čtyři svobody:

- Svoboda spustit program za jakýmkoliv účelem
- Svoboda přizpůsobovat si program svým potřebám a studovat jej
- Svoboda redistribuovat kopie
- Svoboda vylepšovat program a změny zveřejňovat, aby z nich měla prospěch celá komunita. [6]

Základním bodem k určení svobodného softwaru je přístupnost zdrojového kódu. Uživatelé mohou tento kód používat, kopírovat a šířit dále zdarma, případně s poplatkem za distribuci, a přitom jej mohou libovolně upravovat. Zároveň musí být umožněno používat modifikovanou verzi bez nutnosti informovat komunitu o jejím vzniku. Tyto svobody dále platí do té doby, dokud je uživatel svým jednáním neporuší. Když vývojář tedy vydá software jako svobodný, v budoucnu již nemá právo tuto licenci změnit, pokud mu k tomu uživatel

nedal důvod. Svobodný software také neznamená, že nemůže být použit jako komerční. Opak je pravdou a nebývá to ničím neobvyklým.

3.2 PHP Frameworky

Pro jednodušší a rychlejší práci byly vytvořeny takzvané PHP frameworky. Jsou to balíky naprogramovaných objektů a je přesně definováno, jakým způsobem se mají jednotlivé objekty používat. PHP frameworky si zde představíme proto, že na jejich základech jsou vytvořeny všechny e-shopové systémy. Proto jsou důležité pro celistvou představu, jak všechna představená řešení fungují. Představíme si nejdůležitější výhody a nevýhody a z nich plynoucí problémy v bezpečnosti.

Máme celkem tři typy frameworků. Prvním typem jsou takové, které si programátor vytvoří sám pro sebe. Takové jsou vytvářeny přímo na tělo určitému řešení, proto potom může být problém s úpravou pro jiné úkoly. Nehledě na to, že trvá dlouhou dobu, než samotný Framework vytvoříme tak, aby byl plně funkční. Druhým typem je zadání vytvoření v určité firmě. Na takovém řešení spolupracuje tým programátorů, vytvoření je tak poměrně rychlé a konkurenceschopné. Firmy však takové projekty nezveřejňují. Ve své práci se tak budu zabývat především třetím a nejrozšířenějším druhem frameworků. To jsou takové, které vyvíjí tým nezávislých programátorů. Frameworky jsou poté volně šířeny, proto zde funguje rychlá zpětná vazba a jednotlivá řešení jsou téměř neustále testována a je nacházeno vhodnější řešení. Frameworky je možné dále uživatelem upravovat, můžeme vytvářet nové objekty, případně upravit nebo smazat již vytvořené. Zde je však již nutné být velmi opatrný, aby takové změny nenarušily chod celého frameworku. Je tak nutné dodržovat určitá pravidla, aby byla zachována základní funkčnost. V následujících kapitolách představím nejrozšířenější volně dostupné PHP frameworky.

3.2.1 Zend Framework

Zend framework je objektově orientovaný, webový aplikační framework. Tak jako všechny ostatní níže představené je open source, tedy volně šířený a je kompletně vytvořen v PHP 5. Zend vznikl v roce 2005 a v současnosti patří mezi nejrozšířenější a nejoblíbenější frameworky. Jeho oblíbenost plyne z toho, že je vyvíjen s ohledem na jednoduchý vývoj webových aplikací. Jeho komunita se neustále rozšiřuje, proto jeho testování a následné opravy jsou velmi časté. Framework je vyvíjen a podporován firmou Zend, která stojí za vývojem jádra PHP. Mezi další výhody patří především dobrá dokumentace. Nové kompo-

nenty nejsou zahrnuty do nové distribuce dříve, dokud nejsou zdokumentovány do referenční příručky. Výhodné je také to, že Zend využívá modulární architektury a uživatel si tak vybere pouze balíky, které bude potřebovat. Mezi komponentami však existují alespoň částečné závislosti, aby bylo možné je jednoduše propojovat. Výsledné řešení díky tomu může být přesně podle jeho představ. Zend v sobě zahrnuje komponenty pro MVC aplikace, autorizaci a autentizaci, implementuje různé druhy cache, filtrů a validatorů pro uživatelská data, jazykové komponenty a mnoho dalších. Největším záporem je jeho rychlost, která za konkurencí zaostává. [7]

3.2.2 Prado

Prado byl jeden z prvních frameworků, který byl vytvořen pro novou verzi PHP 5. Framework se od ostatních odlišuje tím, že se při vytváření inspiroval především v jazyku ASP.NET a jakoby jej přetvořil do podoby PHP. Není založen na MVC, tak jako ostatní frameworky, ale jedná se o komponentově událostní framework. Abychom tomuto termínu porozuměli, porovnám největší rozdíly těchto dvou systémů. [8]

MVC frameworky se chovají tak, že pro každou aktivitu je vytvořena nová stránka, která odpovídá zadaným požadavkům. U komponentově událostního frameworku je vytvářena pouze jedna stránka a na ní se pouze mění jednotlivé komponenty, které jsou na stránce nezávislé a samostatně reagují na události vyvolané uživatelem. Toho je docíleno pomocí JavaScriptu s technologií AJAX. [8]

Prado má poměrně nepřehlednou dokumentaci, pro lepší seznámení s frameworkem je vhodnější vyhledat různé zpracované tutoriály a procházet obsáhlé fórum.

3.2.3 Symfony

Symfony je webový aplikační open source framework, který vznikl pod původním názvem Sensio Framework. Slouží k vývoji webových aplikací a využívá programovací jazyk PHP ve verzi 5. Využívá návrhového vzoru Model-View-Controller. Symfony byl vytvořen v roce 2005 a je z velké části inspirován ostatními frameworky jako Spring, Django nebo Ruby on Rails. [9]

Symfony je v současnosti ve verzi 2, díky které se stává čím dál více oblíbený. Celou komunitu kolem Symfony tvoří tisíce vývojářů, díky tomu se rychle rozvíjí, má obsáhlou dokumentaci a spousty rozšíření. Framework opět nemusí být využíván pouze jako celek,

ale je možné si vybrat pouze jednotlivé komponenty. Existuje také možnost tyto komponenty využít v kombinaci s konkurenčními frameworky. V českém prostředí tak bývá často využíváno kombinace Symfony s Nette, které se vhodně doplňují. Některé komponenty jsou využívány i v dalších open source řešeních, z nejznámějších uvedu Drupal a Joomla.

Za nejčastější problémy bývá zmiňována především pomalost ve vývojovém prostředí, složitá a nepřehledná komponenta pro ladění a velmi složitá komponenta pro tvorbu formulářů.

3.2.4 Nette

Nette je český projekt jednoho programátora. Autor David Grudl jej nejdříve vytvořil již v roce 2004 pro vlastní potřebu, až později jej uvolnil jako svobodný software. Framework je napsán a podporuje pouze PHP 5 s plným využitím objektově orientovaného programování. Přestože celosvětově se zatím příliš nerozšířil, v České republice patří k nejrozšířenějším a především má nejaktivnější komunitu.

Nette si zakládá na dokonalém zabezpečení, zaměřuje se na eliminaci bezpečnostních rizik. Podporuje všechny moderní technologie jako AJAX, DRY, MVC a znovupoužitelnost kódu. Velmi oblíbený je především jeho ladící nástroj nazvaný Laděnka, který si vývojáři díky své jednoduchosti a přehlednosti implementují i do konkurenčních frameworků. Dle nezávislých testů patří Nette k nejrychlejším frameworkům. [10]

Nette bývá často vyčítána především neúplná dokumentace, kde chybí více ukázek a příkladů. Problémem je také nezachování zpětné kompatibility nových verzí vůči starším.

3.2.5 CakePHP

CakePHP je další open source framework pro vývoj webových aplikací. První verze je založena na PHP 4, inspirovala se u Ruby on Rails, které v té době získalo velkou popularitu. Další verze z předchozích základů vycházejí, tudíž samotné programování a ovládání je lehce zastaralé a neodpovídá moderním trendům. Především v tom ohledu, že nepoužívá objekty, ale pole, používá převážně statické třídy, nemá jmenné prostory.

Framework je postavený na vzoru MVC (model-view-controller) a je vhodný především na rychlé jednoduché projekty, právě kvůli tomu, že je založen na tradičních postupech, na

kterých není potřeba nic nového vymýšlet. Právě kvůli vysoké produktivitě je stále hojně využíván.

CakePHP není modulární, proto je vždy nutné použít celkový koncept.

3.2.6 Srovnání PHP frameworků

Bylo vybráno několik frameworků, které jsou známé mezi PHP komunitou. Kvůli časové náročnosti, která je zapříčiněna hlavně nutností alespoň základního nastudování každého z frameworků, jich není otestováno více.

Z hlediska českého uživatele je velmi zajímavé použití frameworku Nette. Nette nezaostává v žádné činnosti za ostatními známějšími frameworky. Naopak je zde obrovská komunita českých uživatelů, díky které se neustále zlepšuje dokumentace a také stabilita a použitelnost celého systému. Neméně důležité je kompletní lokalizace do českého jazyka. Z těchto důvodů bych pro další práci doporučil použití frameworku Nette.

3.3 E-shopové frameworky

Open source systém je možno bezplatně legálně využívat a modifikovat. Náklady na pořízení a provoz e-shopu na této bázi jsou pouze v ceně domény a hostingu. Za samotný systém se nic neplatí. Pokud má uživatel jistou počítačovou gramotnost, je většinou schopen tento systém sám nainstalovat a nakonfigurovat. Případně si toto může nechat udělat za jednorázový poplatek od programátora nebo odborné firmy. Od profesionálů si taktéž může nechat vytvořit originální vzhled i nejrůznější rozšíření. K open source softwaru existuje zdarma na Internetu množství modul a rozšíření. Prozatím se však takřka výhradně jedná o zahraniční aplikace, které je nutno lokalizovat pro místní podmínky (jazyk, legislativa, DPH, měna).

Jelikož se po celém světě vyskytuje velké množství instalací fungujících na stejném systému, vyplatí se internetovým záškodníkům (hackerům) programovat automatizované viry a roboty, kteří spamují nebo vykrádají data (kontakty zákazníka) a dále je zneužívají. Napomáhá jim k tomu i otevřenost a dostupnost zdrojových kódů. [11]

- Výhody open-source systémů:
    - Cena (zdarma)
    - Podpora internetové komunity (moduly, template, poradny, návody)
- Nevýhody:

- Vyžaduje znalosti nebo pomoc od odborníků
- Mnohdy zastaralé a těžkopádné systémy
- Občas problémy s lokalizací do ČR (měna, jazyk)
- Rozšířenost systému má za následek velké množství podobných (stejných) e-shopů
- Bezpečnostní rizika díky otevřenému kódu a globálnímu užití

3.3.1 Magento

Magento je profesionální systém pro tvorbu e-shopu a na první pohled zaujme především svou uživatelskou přehledností a moderností. Systém je velmi rozsáhlý, funkčně jej lze srovnat například s e-shopem ZenCart.

Velkou výhodou pro české uživatele je podpora české lokalizace a celého prostředí opět především díky stále se rozrůstající komunitě.

„Magento je postaveno na základě Zend Framework, který zajišťuje bezpečnost a škálovatelnost báze kódu. Pomocí tohoto rámce byly pro Magento vytvořeny následující základní principy:

- *Flexibilita: Každé řešení může být stejně jedinečné jako obchod, který dává vzniknout konkrétním požadavkům. Magento je tedy vytvořeno v kódu, který umožňuje lehce provádět potřebné uživatelské úpravy a „vylepšování" systému.*
- *Aktualizovatelnost: Oddělením jádra kódu od obecných a lokálních úprav je umožněno Magento jednoduše přizpůsobovat uživatelským požadavkům, aniž by došlo k omezení možnosti aktualizace na novější verze Magento. A obráceně, Magento lze aktualizovat bez ztráty lokálních změn a nastavení.*
- *Rychlost a bezpečnost: Vysoké standardy tvůrců kódu kopírují moderní praxi a požadavky vedoucí k maximalizaci výkonu softwaru a poskytují bezpečný provoz aplikace."* [12]

Systém je sice založen na frameworku Zend, některé části systému však celou aplikační logiku Zendu nedodržuje a proto se vždy naleznou problémy s administrací. Mezi další zápory patří větší hardwarové nároky a pomalejší vykonávání dotazů.

3.3.2 OsCommerce

OsCommerce se zaměřuje především na začátečníky. Díky jednoduchému kódu je jeho instalace i použití velmi rychlé a intuitivní. Především kvůli nedostatečné funkčnosti vzhledem k ostatním e-shopům není vhodný pro komerční využití. Systém je stavěn stále na původním kódu, proto svou funkčností i vzhledem v současnosti již hůře odolává modernějším e-shopům a pomalu začíná ztrácet své místo na trhu. Především díky obrovské komunitě však stále patří na přední pozice.

Systém je kompatibilní se všemi verzemi PHP od verze 4 výše. Systém byl vytvořen roku 2000 a je kompletně objektově orientovaný. K nevýhodám patří také velká náročnost především na databázový server.

3.3.3 ZenCart

ZenCart je napsán v PHP, využívá HTML komponenty a databázi MySQL. E-shop vznikl v roce 2003 jako odnož již zmíněného e-shopu OsCommerce. V současnosti patří k nejvíce rozšířeným e-shop systémům, především v zahraničí. Pro české uživatele dlouho nebyla k dispozici česká lokalizace, tudíž rozšířenost v ČR dlouho za celosvětovými čísly pokulhávala. V současnosti již existuje rozsáhlá česká komunita, která stojí za vytvořením českého překladu a návodů, včetně diskusního fóra, kde je možné řešit problémy, které při používání mohou nastat. Taková podpora je velmi vítaná a při rozhodování o volbě e-shopového systému může být jedním z nejdůležitějších faktorů. Do ZenCartu je také možné nahrát velké množství grafických šablon, na výběr jsou bezplatné i placené. [13]

Mezi nevýhody patří především velká náročnost na databázový server. Při každém znovunačtení stránky se provede více než 500 SQL dotazů na databázi a tento počet se s narůstajícím množstvím záznamů neustále zvyšuje. Celková náročnost e-shopu souvisí s velmi starým jádrem systému. Proto je jeho použití vhodné především pro uživatele, kterým vyhovují současné možnosti nastavení systému.

3.3.4 OpenCart

OpenCart je spíše menší e-shop, zato však funguje velmi stabilně, má uživatelsky příjemnější prostředí a to jak vzhledem, tak přehledností. Velmi jednoduchá je také administrace jednotlivých částí. Například je možná administrace více obchodů pomocí jedné instalace. Opět zde najdeme placené i neplacené grafické předlohy, které můžeme libovolně využívat.

Obchod nabídne svým uživatelům vložení neomezeného počtu produktů, kategorií a výrobců. Také pro OpenCart existuje české diskusní fórum. Především díky uživatelům a aktivní komunitě si již můžeme dodatečně nainstalovat češtinu, která však stále není kompletní. Také celá komunita není příliš rozšířena, jako u jiných systémů.

3.3.5 Virtuemart

Virtuemart je open source rozšíření/komponenta vytvořená pro redakční systém Joomla! (systém pro zprávu obsahu) a Mambo, která umožňuje tvorbu e-shopu. Druhou možností je použít ho jako katalog zboží. E-shop využívá MySQL databázi a jako ostatní je napsán v jazyce PHP.

Jedná se o třetí nejpoužívanější e-shopový systém na světě. Podporuje různé funkce, například:

- dárkové kupony,
- mnoho platebních bran,
- více světových měn,
- multijazyčnost atd.

V rámci českého prostředí nabízí export nabízeného zboží na stránky, které nabízejí srovnání cen, jako Heureka.cz nebo Zbozi.cz.

3.3.6 PrestaShop

PrestaShop je open source systém pro internetový obchod, vytvořený v PHP a využívající databázi MySQL. Domovská stránka projektu je na adrese http://www.prestashop.com.

Systém funguje na našich serverech s PHP 5.3. Uvedené informace se týkají verze 1.3.3.0 (jiné verze a rozšíření systému mohou pracovat jinak), další případně najdete na stránkách projektu.

Prestashop je open source řešení pro internetový obchod. Prestashop je zcela zdarma. Podporuje různé možnosti plateb jako PayPal, GoPay, Google Checkout, spoustu dalších, které již můžou být zpoplatněny. [14]

Prestashop je uvolněn pod otevřenou licencí. Oficiálně byl uvolněn v srpnu 2007. Systém je postaven na PHPSmarty funkcích. Celosvětově Prestashop používá více jak 150 000 obchodníků. Prestashop vyhrál v roce 2010 a 2011 ocenění Nejlepší open-source e-shopová aplikace (Best Open-source Business Application). Skupina vývojářů byla založe-

na v Paříži, druhou kancelář má v Miami od roku 2011. Prestashop je přeložen do 56 světových jazyků. Technická podpora je poskytována na oficiálním fóru tohoto systému, kde si pomáhá skupina více jak 500 000 aktivních členů. [14]

Systém využívá rozsáhlé možnosti AJAX pro snadnou úpravu v administraci, slouží např. k snadnému přidání různých modulů (rozšíření) do obchodu pro přidání různých funkcionalit, spousta je poskytována různými vývojáři bezplatně. Různé témata a moduly si můžete najít/zakoupit na oficiálním trhu Prestashop Addons. Spousta dobrovolníků neustále přidává různá nová rozšíření. Každý si tak může přidat rozšíření dle svých požadavků. Ve standardním balíčku již je 275 dostupných funkcí. [14]

3.3.7 Nejčastější chyby

- Nekontrolování vstupu od uživatele
    - Všechna data, která web načte od uživatele, by měla být před použitím prověřena. Webová aplikace musí být připravena na to, že se útočník bude chtít záměrně nabourat do aplikace nebo na možnost, že uživatel omylem udělá chybu.
    - Uživatelská data je nutné ověřit na straně serveru, protože kód, který běží na klientovi, uživatel může měnit (například dynamické AJAXové aplikace).
    - Data, která může uživatel měnit:
        - obsah formulářových polí
        - URL adresa požadavku
        - cookies
        - HTTP hlavičky [15]
- Použití neinicializovaných proměnných
    - Starší verze PHP a některé další programovací jazyky automaticky uloží do proměnných data z požadavku.
    - Taková data však mohou být nebezpečná, protože mohou změnit obsah neinicializované proměnné.
    - U nových verzí PHP již kvůli snížení rizika podobných chyb nenačítají neověřená data z formulářů do proměnných, ale vytvoří pro ně speciální pole
        - pro metody GET, POST a cookies budou dostupná pole $_GET, $_POST, $_COOKIE [15]

- Krádež Session ID
    - Existuje jediný způsob, jak 100% zajistit ochranu session:
        - použití SSL (Secure Sockets Layer)
        - vypnutí posílání Referer HTTP hlavičky
    - Útok se provádí pomocí Cross-site scripting a je možný díky tomu, že se nedostatečně kontrolují data na vstupu
    - Jak zajistit bezpečnost:
        - všechny odkazy, které jsou voženy v příspěvcích, je nutné přesměrovat přes pomocnou stránku, která nebude mít v URL uloženo Session ID
        - u Session ID je také nutné kontrolovat shodu IP adresy, protože hodnota Session ID je měněna pro každou stránku [15]
- SQL injection
    - SQL dotaz je často vytvářen dynamicky na základě vstupů.
    - U těchto vstupů je nutné provést pečlivou kontrolu, aby nebylo chybným vstupům umožněno spustit libovolný SQL příkaz. [15]

II. PRAKTICKÁ ČÁST

4 POUŽITÉ NÁSTROJE

V prvních kapitolách praktické části budou uvedeny přímo programy a různé nástroje, se kterými jsem pracoval. Nastíním zde možnost jejich využití a dále návody pro jejich instalaci a využití.

4.1 VirtualBox

Testování bezpečnosti je vždy výhodnější zpracovávat v linuxových operačních systémech. Také však budu potřebovat určité procesy provádět v operačním systému Windows. Proto jsem jako vhodné řešení zvolil použití programu pro spuštění jednoho operačního systému virtuálně ve druhém.

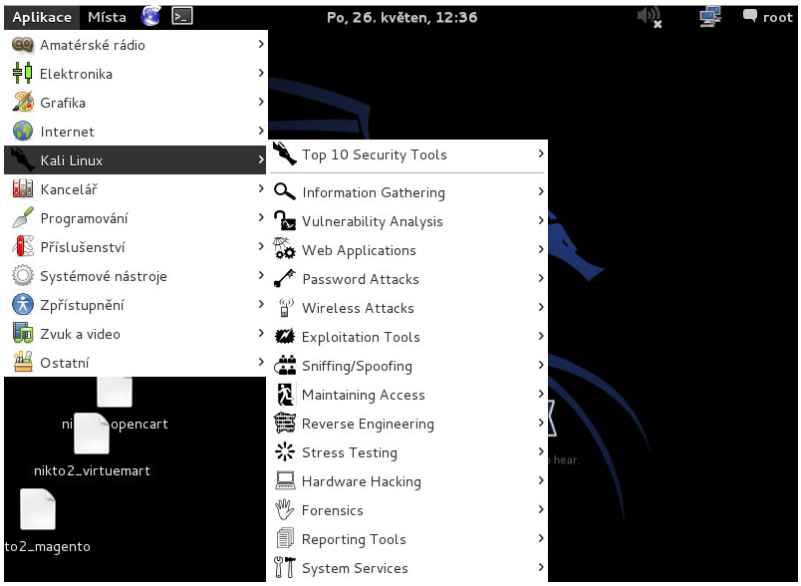
Využil jsem program VirtualBox, který je možné nainstalovat ve Windows a lze v něm plnohodnotně používat linuxový operační systém. Vyvíjí ho Oracle (dříve Sun Microsystems) a umožňuje vytvářet virtuální počítače, do nichž lze instalovat libovolné operační systémy (DOS, Windows, OS/2 Warp, Linux, BSD, NetWare, Solaris). Na jediném přenosném nebo stolním počítači pak můžeme snadno testovat a spouštět aplikace určené pro různé platformy. Po instalaci programu a vytvoření virtuálního PC s Windows 8, stačí stáhnout instalační ISO soubor, vypálit ho na optický disk, případně zkopírovat na flash disk a začít instalaci.

4.2 KALI Linux

Kali Linux je linuxová distribuce odvozená od Debianu. Je navržena speciálně pro penetrační testování a digitální forenzní analýzu. Vychází ze staršího systému BackTrack, ale přidává nové možnosti a nové nástroje. Díky tomu že je plně kompatibilní se systémem Debian je zde možné využívat plnou synchronizaci s příslušnými aktualizačními repozitáři Debianu. Kali je určen speciálně pro provádění bezpečnostních testů v rámci domácího i produkčního prostředí.

Jako ostatní linuxové distribuce lze Kali linux spustit i bez instalování s pomocí Live CD. Samozřejmostí je instalace do virtuálního PC nebo přímo na pevný disk počítače. Distribuci je možné sehnat ve 32 i 64 bitové verzi. Lze jej využívat jako běžný linux, ale jeho nesporná výhoda je v předinstalování aplikací, které slouží pro zjišťování odolnosti webů, serverů, počítačů, počítačových sítí, atd. Na obrázku 1 vidíme základní plochu Kali Linuxu

spolu s otevřeným seznamem testovacích aplikací, které jsou v systému již předinstalovány.



Obrázek 1: Seznam předinstalovaných aplikací v Kali Linux

4.3 Hosting

Při instalaci jakékoliv aplikace nebo programu je potřeba mít prostor pro uložení veškerých dat. Stejně tomu tak bude při instalaci e-shopů, avšak s jedním rozdílem. Na e-shopy se přistupuje pomocí prohlížeče zadáním doménového názvu nebo IP adresy, všechna data tak musí být uložena v daném umístění, aby prohlížeč mohla tato data bez problémů zobrazit. Takové ukládání dat se nazývá hosting. Hostingy mohou být lokální nebo webové.

- Lokální hosting (localhost) má veškerá data uložena pouze ve složce na pevném disku a mohou být zobrazena pouze z daného počítače.
- Webhosting již umožňuje zobrazení webové stránky skrze internet kdekoliv na světě. Data má uložena u sebe poskytovatel hostingu, daný prostor většinou za poplatek pronajímá.

4.3.1 Webhosting

Pro webhosting jsem hledal cenově rozumné řešení, které však bude vyhovovat nutným požadavkům. Vzhledem k velikosti některých e-shopů jsem potřeboval minimálně 130 MB volného prostoru. Dále bylo nutné najít hosting, který podporuje PHP 5, databázový server MySQL, možnost přístupu přes FTP. Dále jsem s hostingem potřeboval doménu 2.řádu (doména 3. řádu je nedostačující kvůli sdílené IP adrese).

Všechny tyto podmínky v kombinace s rozumnou cenou splňoval webhosting Endora.cz. Zaregistroval jsem si tedy vlastní doménu a k tomu zřídil webhosting Free, který je poskytován zdarma, ale pro mé potřeby je dostačující. Registrace byla bezproblémová a účet i s doménou jsem měl k dispozici prakticky ihned po zaplacení. Na obrázku 2 je vyobrazeno logo webhostingu Endora.

Obrázek 2: Logo hostingu Endora

Pro kopírování souborů na web jsem použil FTP klienta v programu TotalCommander. Veškerou správu souborů bylo také možné provádět přes administrátorské stránky hostingu https://webadmin.endora.cz.

4.4 Server

Server je počítač nebo program, který je vytvořen, aby poskytoval nebo realizoval určité služby. U webhostingů už jsou serverové služby automaticky poskytovány, u localhostu je nutné daný program nainstalovat. Ve většině serverových řešení je využíván softwarový server od firmy Apache.

4.4.1 Apache

Apache je open source software dostupný pro všechny operační systémy. Jeho vývoj začal v roce 1993 v americkém NCSA - Národním centru pro superpočítačové aplikace. Apache patří k nejrychlejším, ale také nejvýkonnějším webovým serverům. Server podporuje velké množství funkcí, libovolně lze doinstalovat velké množství přídavných modulů rozšiřujících jádro programu. Mimo jiné obsahuje externí modul pro kompresi dat webových stránek, které jsou posílány přes protokol HTTP (mod_gzip). Apache umožňuje také virtuální hosting, tedy možnost obsluhovat více webových stránek na jednu instalaci Apache.

4.4.2 Nginx

Nginx je konkurenční open source softwarový webserver. Umí pracovat s protokoly HTTP, HTTPS, SMTP, POP3, IMAP a SSL. Nginx má různé varianty pro všechny operační systémy, ale zaměřuje se především na unixové platformy.

Základním cílem je poskytovat vysoký výkon při velmi nízkých nárocích na paměť. Dokáže velmi rychle zobrazovat statický obsah díky rozložení zátěže na další servery. Je vhodný především pro větší firmy, je využíván například firmami Seznam.cz, Nokia, WordPress nebo Dropbox.

Webhosting Endora má nainstalován server Nginx ve verzi 1.5.10.

4.5 PHP

Jelikož všechny e-shopy, které jsem pro testování vybral, jsou napsány v jazyce PHP, je nutná instalace této technologie na webhostingu. U všech hostingových služeb již je instalace PHP samozřejmostí. V současnosti jsem využíval verzi 5.4.19. Při instalaci na lokálním serveru bývá většinou potřeba doinstalovat některé přídavné moduly, například php-curl. U webových řešení už jsou tyto hojně používané moduly nainstalovány automaticky.

4.6 Databáze

Všechny e-shopy potřebují svou vlastní databázi, do které si ukládají tabulky s daty. Většina hostingů již opět poskytuje alespoň jednu databázi ve freewarové verzi svého hostingu, ale narazil jsem i na takový, který technologii MySQL poskytoval až u placených variant. Proto je nutné při výběru poskytovatele hostingu pečlivě číst provozní podmínky.

Díky akční nabídce jsem na první týden používání získal od poskytovatele hostingu možnost vyzkoušet placenou verzi hostingu. Díky tomu jsem měl k dispozici dvě technologie ukládání dat v systému MySQL.

- MyISAM je nejpoužívanější formát ukládání dat pro MySQL databázový systém. Vychází z původního formátu ISAM. Použití úložiště MyISAM je velmi různorodé, není omezeno platformou a především za celou dobu svého vývoje již obsahuje velké množství doplňkových funkcí.
- InnoDB byl navržen především pro zpracování mnoha krátkodobých transakcí, které se anulují velmi zřídka. Tento typ úložiště je nutný pro e-shopový systém VirtueMart, neumožňuje nastavit ukládání do databáze MyISAM.

5 TESTOVÁNÍ

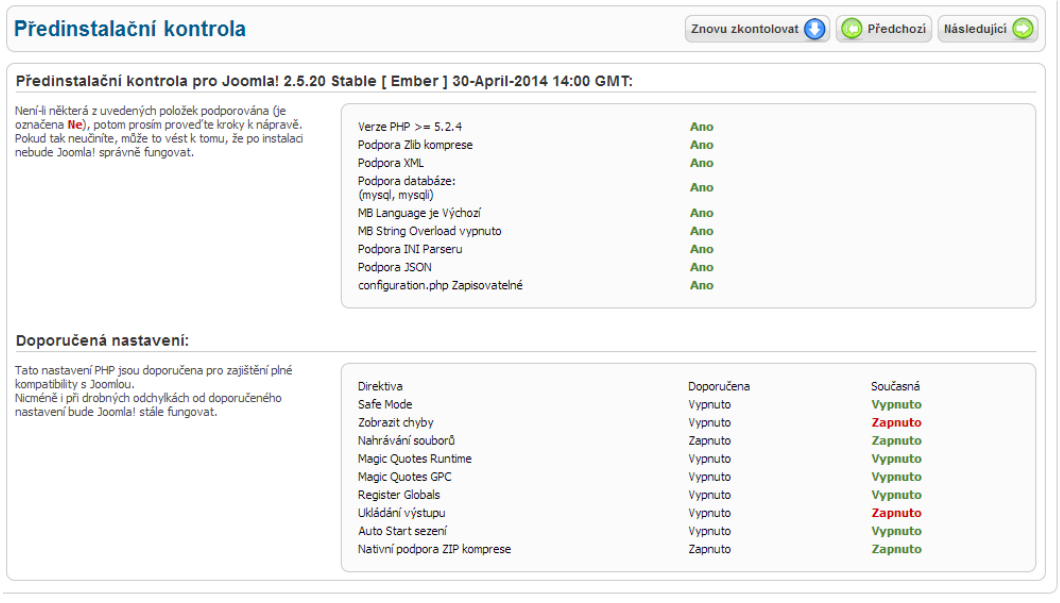
5.1 Výběr e-shopů pro testování

Mezi nejrozšířenější a nejznámější open source e-shopy patří například ZenCart, osCommerce, PrestaShop, Quick Cart, OpenCart, OpenSolution. Právě z hlediska rozšířenosti jsem hledal e-shopová řešení pro vlastní testování. Vybral jsem takové frameworky, které patří k nejpoužívanějším, tím pádem se u nich dá předpokládat pravidelná aktualizace a dostatečné zabezpečení. Jmenovitě se budu dále zabývat e-shopy Magento, OsCommerce, OpenCart, VirtueMart a PrestaShop. ZenCart jsem kvůli velké podobnosti s ostatními ze seznamu vypustil.

V kapitole ukáži základní postupy a nastavení, která jsou prováděna při instalaci jednotlivých frameworků.

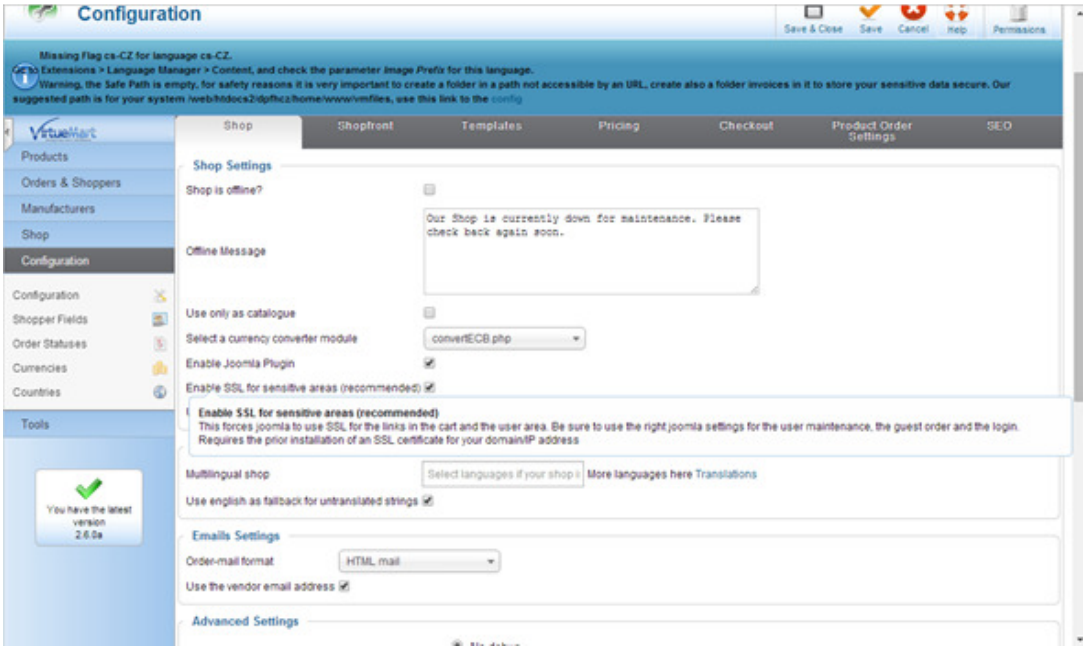
5.2 Instalace VirtueMart + Joomla

VirtueMart je pouze modul vytvořený do redakčního systému Joomla. Po nahrání všech souborů na web začne nejdříve instalace Joomly ve verzi 2.5. Na obrázku 3 vidíme předinstalační kontrolu, která nám ukáže, zda máme provedena správná nastavení pro instalaci. Tyto nastavení lze změnit v administrátorském účtu hostingu Endora. Po provedené kontrole zůstaly zapnuty služby *Zobrazit chyby a Ukládání výstupu.* Pomocí těchto doplňků může útočník jednoduše zjistit strukturu webu, proto je vhodné je vypnout.



Obrázek 3: Instalace systému Joomla! - předinstalační kontrola

Pokud se nechceme dále zabývat jednotlivými nastaveními, pokračujeme dalším krokem, kde zadáme údaje k databázi. Je vhodné mít pro každý e-shop vlastní databázi. Na obrázku 4vidíme již nastavení samotného e-shopu VirtueMart po dokončení instalace. Je důrazně doporučováno zaškrtnout možnost zapnutí SSL šifrování. Po ukončení instalace nám VirtueMart přímo v okně prohlížeče nabídce kliknutím smazat složku *install*, aby žádný útočník nemohl náš obchod přeinstalovat. Také se doporučuje přejmenovat složku *admin*.



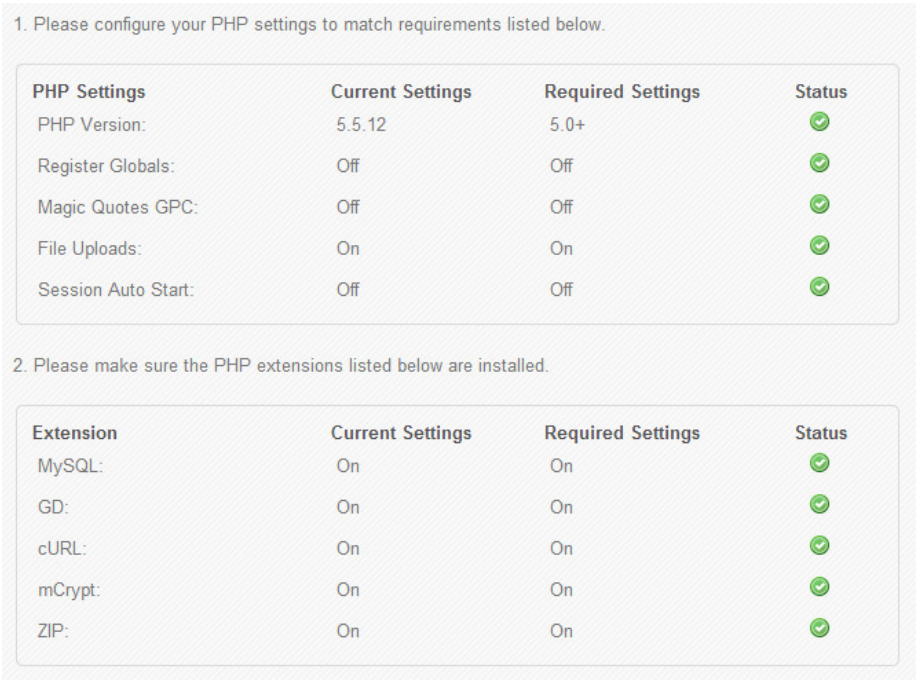
Obrázek 4: Nastavení e-shopu VirtueMart

5.3 Instalace OpenCart

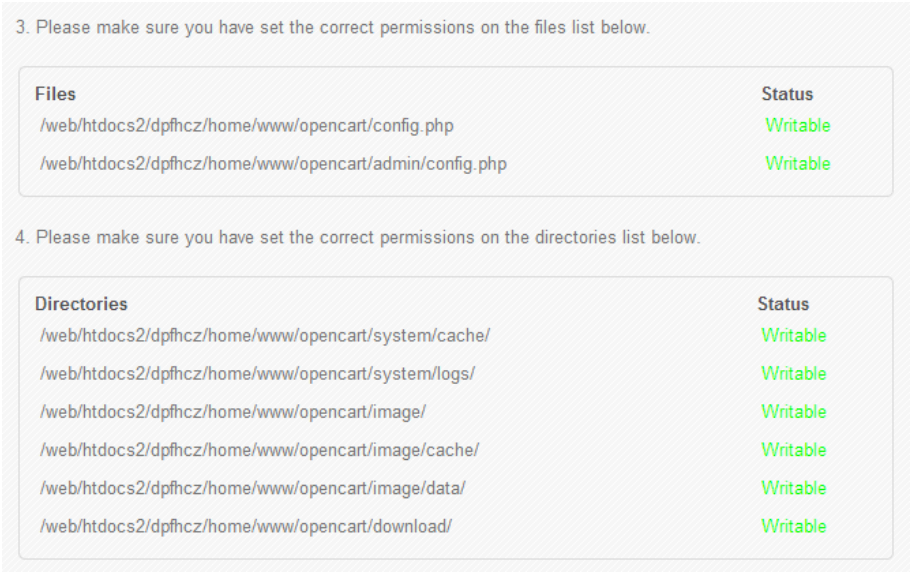
Při instalaci OpenCartu nám hned jako první okno vyskočí také podobná předinstalační kontrola, jako u VirtueMartu - viz. obrázky 5, 6. Po nahrání souborů na web je nutné ještě před instalací změnit název dvou souborů. Konkrétně:

- ~/www/opencart/config-dist.php na ~/www/opencart/config.php
- ~/www/opencart/admin/config-dist.php na ~/www/opencart/admin/config.php

Dále už pouze zkontrolujeme, zda jsou všechna nastavení v pořádku, můžeme si také ověřit verzi PHP a poté pokračovat dalším krokem k nastavením databáze a administračního účtu. Na konci instalace opět budeme upozorněni na nutnost smazání složky *install*. Poté již bude OpenCart připraven k použití.



Obrázek 5: Instalace OpenCart - předinstalační kontrola a)

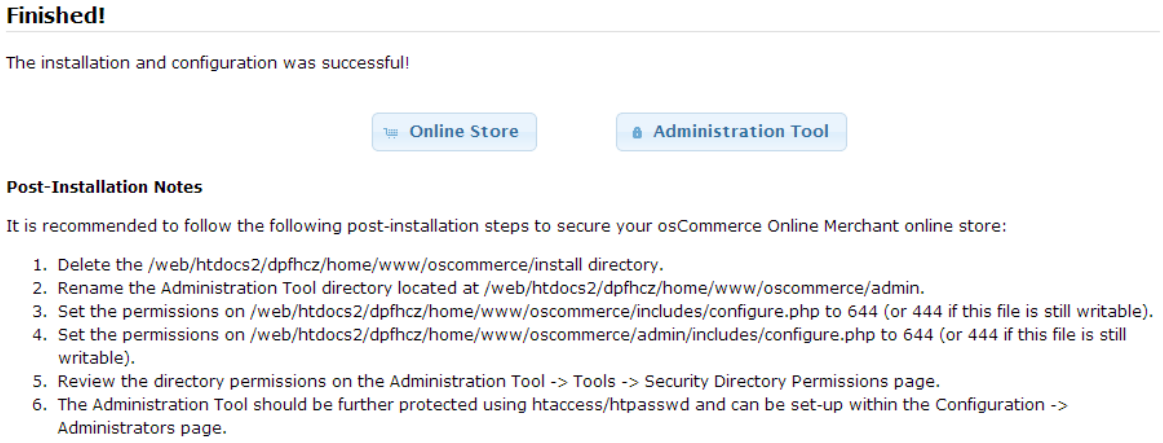


Obrázek 6: Instalace OpenCart - předinstalační kontrola b)

5.4 Instalace OSCommerce

Instalace OsCommerce probíhá téměř identicky jako předchozí e-shopy. Opět je nutné nastavit údaje pro vstup do databáze a do administrační části webu. Po dokončení opět dostaneme několik upozornění, která je potřeba vykonat. Kromě smazání složky *install* a přejmenování složky *admin* zde musíme nastavit práva souborů configure.php tak, aby nebylo

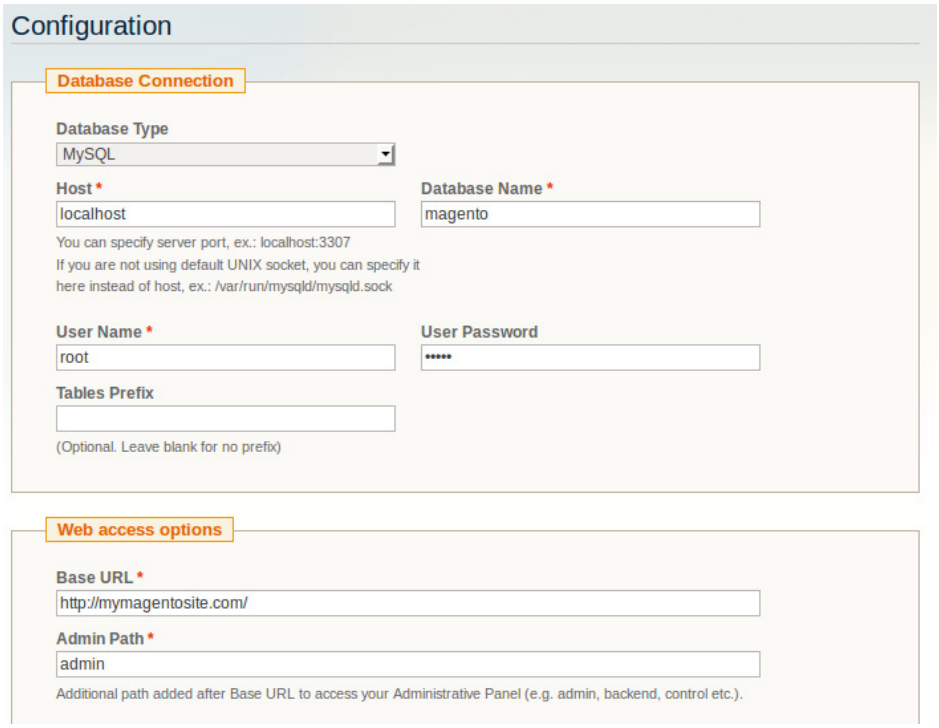
možné do nich zapisovat. Opět to slouží jako základní bezpečnostní opatření před útočníky. Na obrázku 7 jsou uvedeny všechny podrobnosti k dokončení instalace OsCommerce.



Obrázek 7: Informace k dokončení instalace OsCommerce

5.5 Instalace Magento

U instalace e-shopu Magento se můžeme podívat na obrázek na instalační nabídku pro nastavení databáze. Konkrétně zde nastavuji databázi MySQL na lokálním serveru.



Obrázek 8: Nastavení databázového serveru MySQL

6 TYPY ÚTOKŮ A JEJICH ŘEŠENÍ

Při vývoji jakékoliv aplikace by mělo být myšleno především na její bezpečnost. U webových aplikací to platí dvojnásob. V současné době již má přístup na internet dostupný v podstatě každý. Stačí objevit jakoukoliv aplikaci, která není řádně zabezpečena a může to způsobit velké škody. Útoky mohou být mířeny tak, aby smazaly co nejvíce dat, nebo naopak mohou na server vložit vlastní škodlivý soubor. Dnes jsou na všechny typy útoků vytvořeny automatizované aplikace nebo alespoň podrobné návody. Tím se vytrácí odbornost všech uživatelů a útoky může provádět každý, kdo umí kliknout myší.

Existuje nepřeberné množství různých webových útoků. Jedná se například o SQL Injection, Cross.site Scripting, Cross.site Request Forgeries, atd. Všechny bezpečnostní problémy však vycházejí z nekvalitního programování aplikací. Jednotlivé technologie mohou být velmi otevřené, ale jednoduchým příkazem je lze kvalitně zabezpečit. Bohužel ne všichni tvůrci si toto uvědomují.

Jednou z nejdůležitějších věcí při zajišťování bezpečnosti je ochrana hesla. Ukládat heslo jako čistý text je velká chyba. Sice je jednoduché zjistit heslo uživateli při zapomenutí a poslat mu ho, ale přinejmenším heslo uživatele bude znát administrátor a vývojář aplikace. Jedná se tedy o bezpečnostní riziko. Vhodným řešením je vytvořit pouze otisk hash z hesla. Otisk hash je posloupnost znaků, která má pevnou a neměnnou délku. Z jednoho hesla vznikne vždy jen jeden otisk. Veškerá autentizace tedy probíhá pouze na úrovni porovnávání dvou nicneříkajících posloupností znaků. Hashovací funkce jsou jednocestné, tudíž není možné získat z otisku zpětně heslo.

V následujících odstavcích této kapitoly jsou popsány jednotlivé druhy útoků a způsoby ochrany aplikace e-shopu proti výše zmíněným nežádoucím útokům. [16]

6.1 SQL Injection

Pod pojmem SQL Injection se skrývá podvržení vstupních dat takovým způsobem, že je určitým způsobem pozměněn výsledek SQL dotazu. Útočník toho docílí tím, že přes neošetřený vstup nepozorovaně „vsune“ svůj kód, kterým změní výsledný SQL dotaz. Velké škody mohou nastat, především pokud útočník zná strukturu databáze, jednotlivých tabulek a dotazů. V takovém případě je pro něj velmi jednoduché provést útok a získat citlivá data.

Jaké problémy mohou při takových útocích nastat? Útočník se může dostat k citlivým datům všech uživatelů, od uživatelských hesel až po e-mail. Může tak získat přihlašovací

údaje k administrátorským účtům a samozřejmě všem ostatním účtům. Je zde také možnost, že útočník smaže všechna data v tabulkách a tím databázi zcela vyřadí z provozu. Při získání administrátorských práv je samozřejmě také možné měnit popisy nebo ceny u jednotlivých produktů, případně vložit na stránky infikované soubory, které se poté budou šířit mezi návštěvníky webu.

SQL Injection se provádí pomocí neošetřených vstupů webové aplikace. Mezi takové vstupy patří formuláře, importy souborů, parametry URI nebo HTTP/XML/SOAP komunikace použité například u cookies.

Existují tři typy útoků, které se odlišují především tím, jakým způsobem se útočník dostane k ukradeným datům:

- Inband přímý útok
    - výsledky útoku jsou zobrazeny přímo na stránce v prohlížeči
- Out-of-band - nepřímý útok
    - útočník si například nechá zaslat výsledek na zadaný e-mail
- Blind - útok naslepo
    - k těmto výsledkům útočník nemá přístup

6.1.1 Ochrana před SQL Injection

Existuje dlouhá řada návodů, jak se SQL Injection vyhnout. Opatření se také mohou odlišovat podle typu SQL systému, ať už používáme řešení od Microsoftu, open-source MSSQL nebo jiné.

K základním ochranným prvkům patří především dostatečná znalost používání SQL. Největším problémem u naprogramovaných SQL systémů je především v samotném programování. Při dobré znalosti tohoto jazyka a vhodném použití jeho funkcí by k veškerým útokům vůbec nemuselo dojít. V první řadě všech bezpečnostních opatření by tedy měl být správný výběr daného řešení, případně vhodný výběr programátorů.

K obecným radám dále patří především dostatečné oddělení správcovských účtů od běžného provozu v databázi. Pro běžné účty by měla být samozřejmá úprava administrátorských práv tak, aby nebylo možné běžně mazat tabulky, případně celé databáze. V ideálním řešení má daný uživatel přístup pouze k předem vytvořeným a uloženým bezpečným procedurám, které může pouze spouštět. Veškeré vytváření, úpravy a mazání jsou v kompetencích hlavního správce.

Programátor webových aplikací by měl zajistit, aby nebyl všem uživatelům zobrazován přesný popis vyskytnuvší se chyby. Útočník jinak může velmi jednoduše zjistit i celé části SQL kódu a celý proces vytváření útoku a hledání skrytých cestiček je mu tak značně ulehčen.

Zde představím rady pro jednoduché zabezpečení internetových aplikací, jejichž použití je univerzální a zajistí základní ošetření vstupů:

- Nahrazení jednoduchých apostrofů za zdvojené
    - U všech textových vstupů je nutné provést náhradu apostrofů ´ za znak ´´ (dvojice apostrofů). Takové nahrazení je velmi jednoduché a přitom efektivní a znemožní použít jednoduchý apostrof pro ukončení vytvářeného SQL dotazu
    - Pro útočníka znalého takových úprav je však jednoduché tuto ochranu obejít tím, že použije dotaz v hexadecimální podobě. Při zřetězení takového dotazu se ovšem z hexadecimálního tvaru stává kód i s apostrofy.
- RegExp transformace
    - Tuto transformaci provedeme pro všechny textové vstupy. Tím budeme mít zajištěno, že vstupy budou obsahovat pouze takové znaky, které v nich mají být. Transformaci je možné použít i na všechny výstupy a tím je ošetřit před napadením pomocí XSS.
- Typová konverze
    - U netextových vstupů by měly proběhnout typové konverze. Například pokud víme, že vstupem budou pouze čísla v ne příliš velkém rozmezí, použijeme na daný vstup konverzi funkcí integer. Poté budeme mít jistotu, že nám neprojdou neočekávané znaky.
- Oddělené předávání dotazu
    - Tento bod patří v boji proti nevyžádaným návštěvníkům k nejúčinnějším. Základním problémem u všech vstupů je to, že umožňují zřetězení jednotlivých řádků kódu do výsledného dotazu a jeho následné odeslání na server. Za dostatečné řešení, které nelze obejít se považuje oddělené předávání dotazu a jeho parametrů databázovému serveru.

6.2 Cross-Site scripting (XSS)

Útoky XSS jsou takové, kdy útočník vloží na webovou nezabezpečenou stránku vlastní JavaScriptový kód, tím naruší konzistenci webových stránek a získá kontrolu nad prohlížečem návštěvníka. Kód je možné vložit přes formulářové vstupy, ale také přes tzv. hidden fieldy, hlavičky http protokolu nebo pomocí cookies. Možnosti útočníka jsou omezeny pouze omezeními samotného JavaScriptu, nad prohlížečem může získat absolutní moc. XSS umožňuje nejen vkládání škodlivých kódů, ale i provádění změn vzhledu i obsahu webů, popř. jejich funkčnosti. Může docházet i k získávání citlivých údajů nebo k obcházení bezpečnostních prvků.

XSS patří k nejstarším útokům na webové aplikace. Ale mnoho webů stále nemá vytvořenou spolehlivou ochranu, proto jsou tyto útoky tak časté. Navíc se neustále vyvíjí a přicházejí nové myšlenky, jak útok zamaskovat, aby nebyl odhalen. Pomocí útoku XSS byly napadeny i velmi známé webové stránky, například FBI, Yahoo, CNN, Apple či Microsoft. Nebezpečnost XSS je velmi vysoká, ale na druhou stranu obrana před ním je velmi jednoduchá. [17]

Pro Cross-site scripting se již delší dobu využívá zkratka XSS, která zcela nahradila původní zkratku CSS, protože zde často byly problémy se záměnou se zkratkou pro kaskádové styly CSS. Dle amerických průzkumů je sedm z deseti webů zneužitelných pomocí XSS. Proto je označován jako hrozba číslo jedna u webových aplikací.

Obecně se tedy jedná o vložení cizího kódu tam, kde s tím programátor nepočítal. U XSS rozlišujeme tři druhy útoků:

- Non - persistent - útoky provedené jako vsunutí nebo úprava URL odkazu
    - Local (DOM based) - Kód je vložen do URL adresy a webová stránka s takto neošetřeným přenesením proměnné z URL do Javascriptu jej zpracuje jako svoji součást. Útok nezasahuje samotnou webovou stránku, ale spočívá v tom, že stránka zpracuje nekorektní URL odkaz, který může být takto „podstrčen“ nic netušícímu uživateli. Tímto útokem mohou být napadeny i statické stránky.
    - Reflexed -Útok je velmi podobný předchozímu lokálnímu, ale týká se výhradně stránek, které jsou dynamicky generovány, tedy například vytvořené v jazyku PHP. Do kódu URL je vložen odkaz, který se stane

součástí stránky, většinou jako nadpis, tudíž není vůbec upravován, kontrolován či filtrován. [17]

- Persistent (stored) - nejnebezpečnější
    - Provádí úpravu přímo v kontextu stránky. Skript je vložen do stránky jako součást komentáře. Komentář je následně i se skriptem uložen do databáze a je zobrazován všem návštěvníkům stránky. [17]

6.2.1 Ochrana před XSS

- Nahrazení znaků < a > a dalších speciálních znaků
    - Vzhledem k tomu, že pro většinu webů se používá jazyk php, jsou všechny příkazy ohraničeny znaky < a >. Pokud by tedy útočník přes nezabezpečený vstup webové stránky chtěl vložit svůj kód, musel by jej vložit společně s těmito znaky. Proto je velmi vhodné tyto znaky na vstupu zakázat, respektive zaměnit za jiné znaky. Prohlížeč tak vložený kód nebude překládat a tím bude zabráněno většině útoků, které jsou prováděny tímto způsobem. Je však důležité, aby byly zabezpečeny všechny vstupy. Pokud si programátor bude myslet, že stačí ošetřit pouze některé, které jsou podle něj nebezpečné, je veškerá snaha o kvalitní zabezpečení zbytečná.
    - K dalším znakům, které je nutné nahradit, patří uvozovky a apostrofy. S použitím těchto speciálních znaků je totiž možné překonat předchozí ochranu. V PHP je možné pro tento účel použít speciální funkci *htmlspecialchars()*. Tato funkce kromě apostrofů a uvozovek nahrazuje také znaky & a dříve zmiňované < a >. Všechny vyjmenované speciální znaky takto nahradí pomocí bezpečných html entit.
- Použití whitelistů
    - Použití whitelistů se uplatňuje především u speciálních případů, kdy uživatelským vstupem není prostý text, tak jak zmiňujeme v předchozích případech, ale vstupem je HTML kód. Toho se využívá především u webového rozhraní e-mailu, kdy je potřeba zobrazovat různé obrázkové přílohy, či jiné grafiky. U takových vstupů nemůžeme jen tak zakázat veškeré speciální znaky, je však velmi vhodné vytvořit seznam, které znaky použít můžeme. Takovému seznamu znaků a jejich povolených bezpečných atributů se říká whitelist. Z celého rozsáhlého HTML kódu na vstupu tak

budou odstraněny veškeré nepovolené HTML značky a atributy a zůstanou jen ty, které jsou povoleny v seznamu.

- Whitelisty jsou uplatňovány také jako ochrana před použitím XSS u kaskádových stylů. Tato problematika je velmi podobná předchozí zmiňované a stejné je i její řešení. Kaskádové styly totiž obsahují určité možnosti, kudy útočit na webovou aplikaci, především se zde zaměřuje na možnost spustit JavaScript. Možnosti, jak provést takové útoky, se liší podle použití internetového prohlížeče, každý má totiž speciální vlastnosti, kterých lze při útocích využít. Jako bezpečné řešení se používá již zmíněný whitelist. [18]

6.3 Cross-Site Request Forgeries (CSRF nebo XSRF)

Cross-site Request Forgery (CSRF) je jedna z metod útoku do internetových aplikací. Princip útoku je dán slovem *forgery*, které v překladu znamená *padělání.* Cílem útočníka je donutit uživatele, který je přihlášený k webové aplikaci, aby kliknul na odkaz, který slouží k vykonání určité operace uvnitř aplikace. Přestože byla změna provedena bez vědomí uživatele, bude tato akce považována za oprávněnou. K úspěšnému vytvoření funkčního odkazu potřebuje útočník dobře znát strukturu napadené aplikace. Velký bezpečnostní problém je v tom, že oběť většinou neklikne na odkaz úmyslně. Tento padělaný odkaz je možné uschovat například jako URL adresu do skrytého obrázku, který je uložen na určité webové stránce. Druhou možností je použití automaticky odeslaného skrytého formuláře, jehož zdrojový kód je napsán v JavaScriptu. [19]

6.3.1 Ochrana před CSRF

Nejjednodušším a nejúčinnějším řešením je použití jednorázových autorizačních tokenů. Při výpisu stránky je náhodně vytvořen token, který se uloží do databáze, případně do session. Token je také přidán do skrytého formuláře, který obsahuje všechny parametry, které identifikují uživatelskou akci. Token pak bude spolu s parametry odeslán zároveň a před následným vykonáním požadavku se nejdříve ověří shoda odeslaného tokenu s očekávaným. Pokud se shodují, akce je schválena a token zahozen.

Níže nastíním jednoduché řešení implementace autorizačních tokenů. Nejprve token vytvoříme: [20]

```
<?php
$token = substr(md5(rand()),0,32);
```

Obrázek 9: Vytvoření autorizačního tokenu[20]

Následně při vypisování formuláře bude token uložen do databáze. Ve formuláři se doplní nový parametr *token:* [20]

```
mysql_query("INSERT INTO auth_tokens (token, validity) VALUES
('$token', NOW() + INTERVAL 1 HOUR)");
echo "<input type='hidden' name='token' value='$token' />\n";
```

Obrázek 10: Uložení tokenu o databáze[20]

Po provedení uživatelské akce se pomocí následujících řádků ověří platnost tokenu a aplikace provede příkazy, které jsou zapsány v podmínce: [20]

```
mysql_query("DELETE FROM auth_tokens WHERE validity <
NOW()");
mysql_query("DELETE FROM auth_tokens WHERE token =
'$_REQUEST[token]'");
if (mysql_affected_rows()) {
echo "Správný token";
}
?>
```

Obrázek 11: Ověření platnosti tokenu[20]

Pro zvýšení ochrany je možné použít dvojnásobné ověřování například pomocí captcha kódu nebo jiné autorizace požadavku. Ze strany uživatele je možná jediná ochrana a to taková, že bude vždy přihlášen pouze v jedné aplikaci. Při odchodu na jinou stránku je pak vždy nutné se odhlásit. [20]

6.4 Full Path Disclosure (FPD)

Anglický název Full Path Disclosure je možné přeložit jako odhalení nebo prozrazení úplné cesty. Cestou je myšlena úplná cesta právě spuštěného skriptu nebo právě používaného souboru. Při vyzrazení této informace útočníkovi je pro něj již velmi jednoduché se do daného adresáře dostat a cokoliv v něm změnit.

Co všechno lze z takto uniklé informace zjistit?

- Lze vyčíst plnou cestu k souborům a skriptům, především ke skriptu search.php, pomocí kterého právě můžeme vyhledat všechny skripty a cesty k nim
- Zjištění, zda web běží na PHP a zda má zapnuté zobrazování chybových hlášek. Právě z chybových hlášek poté můžeme zjistit všechny důležité informace.
- Podle adresářové struktury můžeme zjistit, zda je využit některý rozšířený framework (například Nette, Zend, atd.), protože každý má svou pevně danou strukturu a chování. Pokud není využit žádný framework a autor si vytvořil webové řešení sám, je velmi pravděpodobné, že útočník najde spoustu bezpečnostních chyb.
- U mnohých webových aplikací lze zjistit také verze použitých nástrojů, typ databázového serveru nebo kompletní seznam souborů a funkcí, které byly volány.

Útok FPD je poměrně běžný typ útoku a velmi často bývá úspěšný. V míře nebezpečnosti jej tak lze jmenovat na předních místech, především proto, že pomocí takto získaných dat pak útočník může jednodušeji provádět ostatní útoky jako SQL injection. Přitom jeho vytvoření není vůbec složité a v podstatě nepotřebuje žádný návod. Jde pouze o to, vyvolat chybovou hlášku PHP, která standardně obsahuje i celou cestu k souboru, ve kterém došlo k chybě.

Uvedu zde několik základních informací, jak by se mohly tyto útoky provést.

- Přetypování parametrů v URL nebo ve formuláři
    - Pokud za název parametru přidáme prázdné hranaté závorky, jazyk PHP automaticky ze vstupního parametru vytvoří pole. Vytvořené skripty ale dál pracují s touto proměnnou stejně a předají ji funkcím, které pracují pouze s řetězci. Tyto funkce však s polem pracovat neumějí a proto zahlásí chybu. Na následujícím obrázku je zobrazena chyba, která je hlášena po zadání příkazu *search.php?q[]=....* Následně je možné ve formulářích odesílaných metodou POST změnit název prvku INPUT tím, že k němu přidáme další hranaté závorky, například *name="search[]"*. Na obrázku 12 vidíme zobrazení chybové hlášky s úplnou cestou k souboru.

Obrázek 12: Zobrazení chybové hlášky s plnou cestou k souboru

- Použití prázdného identifikátoru session
  - U session identifikátoru platí určitá omezení ohledně nepovolených znaků a jejich počtu. Nesmí být použitý znak mezera a také nesmí být zadán prázdný řetězec. V opačném případě zahlásí chybu.
  - Identifikátor se dá změnit několika způsoby. První možnost je ho změnit přímo v prohlížeči pomocí cookie souboru *PHPSESSID*. Alternativním řešením je změnit ho pomocí Javascriptu. Je nutné zadat danou část kódu do řádku s adresou v momentě, kdy je načten náš web a následně tuto stránku obnovit.

```
javascript:void(document.cookie="PHPSESSID=");
```

Obrázek 13: Změna identifikátoru promocí PHPSESSID

- Přímé volání souborů
  - Pokud budeme chtít přímo zavolat soubory, které jsou pouze vloženy do jiných souborů a vyžadují tak určité nahrané knihovny k tomu, aby mohly být spuštěny, opět se to PHP nebude líbit a vypíše chybovou hlášku.
- Vynechání povinných parametrů
  - Pokud je skript vytvořen tak, že při jeho spouštění jsou nutné povinné parametry, tak pokud žádné parametry nezadáme, uvidíme další chybovou hlášku.

6.4.1 Ochrana před FPD

Základem vhodného zabezpečení před FPD je samozřejmě samotné programování webové aplikace, ovšem ani takto nikdy nedocílíme dokonalé ochrany, protože PHP je nevyzpytatelný jazyk a vždy se najde skulinka, kudy se dá protáhnout.

Jedinou možností, jak se efektivně bránit před FPD je vypnutí zobrazování chybových hlášek. Samozřejmě hlášení nesmí být zablokována úplně, pouze se musí zobrazovat pouze těm správným osobám, aby veškeré chyby mohly následně opravit.

Ochranu tedy provádíme v konfiguraci serveru úpravou souboru *.htaccess* tím, že přidáme tento řádek kódu: [21]

```
php_flag display_errors off
```

Obrázek 14: Úprava souboru .htaccess[21]

Dále je tedy nutné vytvořit místo, kam se chybová hlášení budou zapisovat. To zajistíme zapnutím direktivy *log_errors*, které provedeme také v souboru *.htaccess*. [21]

```
php_flag log_errors on
```

Obrázek 15: Zapnutí direktivy log_errors[21]

6.5 Krádež ID relace

Pokud chce útočník získat citlivá data oběti, většinou cílí své útoky na krádež Session ID. Pohyb uživatelů na webu je možné monitorovat pomocí relací (Sessions). Princip těchto relací je velmi jednoduchý - každý uživatel, který otevře stránku, dostane Session ID - jednoznačný identifikátor.

Relace vzniká po přihlášení uživatele k serveru. Je to množina proměnných, které uchovávají aktuální stav připojení klienta k serveru. Servery mohou obsluhovat mnoho připojení, proto spojují jednotlivé relace s konkrétními uživateli. Obvyklý postup je ten, že server vygeneruje cookie a zašle ji uživateli. Cookie obsahuje identifikační informace, které kli-

ent spolu s každým dotazem odesílá zpět serveru. Server podle přijaté cookie rozpozná klienta, pokračuje v relaci a odpovídá na dotazy. Po odhlášení klienta se cookie zruší nebo alespoň zneplatní. Jestliže útočník zcizí cookie a odešle spolu s dotazem na server, pak server nic nepozná a na dotaz odpoví. Takto útočník získá identitu a všechna práva oběti.

Existuje také možnost uložit identifikátor přímo do URL, v případě, že nejsou k dispozici cookies, ale vzhledem k tomu, že URL adresa je velmi používaný parametr, který se navíc ukládá na více místech, je toto řešení velmi nešťastné.

6.5.1 Ochrana před krádeží ID relace

Existuje několik možností obrany, ale žádná z nich není dostačující:

- Server může spojovat identifikátor uložený v cookie s některými dalšími informacemi o klientovi. Například použije IP adresu klienta. V tomto případě pak kontroluje, zda se nezměnila IP adresa, ze které byl zaslán identifikátor. Pokud ano, pak relaci okamžitě ukončí. Tato kontrola ale nebude fungovat v případě, kdy oběť i útočník používají stejný proxy server. [22]
- Obdobně může server použít informace z hlavičky HTTP. Jestliže dorazí identifikátor a sledovaný údaj v HTTP hlavičce se bude lišit od předchozí hodnoty, pak server opět okamžitě ukončí relaci. Nevýhodou je fakt, že útočník může HTTP hlavičku snadno editovat. [22]
- Dalším možným řešením je, že server bude s každou odpovědí zasílat klientovi nový identifikátor. V tomto případě existuje riziko, že útočník odešle dotaz na server dříve než klient. Server pak pošle odpověď a nový identifikátor útočníkovi a klientovi znemožní přístup k relaci. [22]

Zcizení cookie útočníkem nemůžeme ve své webové aplikaci ovlivnit, to si musí každý uživatel ohlídat sám, ale minimálním opatřením by mělo být alespoň vygenerování nové cookie vždy po zalogování uživatele do aplikace.

7 NÁSTROJE PRO TESTOVÁNÍ

Aplikací, kterými lze testovat bezpečnost webových stránek existuje velké množství. Existují programy použitelné pouze v operačním systému linux nebo naopak pouze v operačním systému Windows. Samozřejmě jsou programy pro použití napříč různými platformami. Z každé této kategorie jsem našel jeden program, který mi přišel nejvhodnější a nejvíce univerzální.

Ze všech různých aplikací mi přišla nejzajímavější Vega Web Vulnerability Scanner. Tento skener není omezen operačním systémem, obsahuje spoustu modulů pro testování a především je poskytován zdarma jako open source. Pomocí této aplikace jsem provedl většinu testů.

Druhým použitým programem byl Acunetix Web Vulnerability Scanner. Acunetix je dostupný pouze na platformě Windows a především je to program placený. Já jsem však vyzkoušel 15-ti denní verzi Trial, která mi umožnila provést testy, které jsem potřeboval.

Pouze na linuxových systémech je dostupná aplikace Nikto2. Díky jejímu využití v Kali linuxu ji nebylo nutné instalovat, stačilo zadat příkaz v terminálu. Tím se dostáváme k největšímu rozdílu Nikto2 od předchozích skenerů, že je dostupná pouze v textové formě bez grafického prostředí a ovládá se pouze z terminálu.

Všechny nástroje budu popisovat podrobněji v dalších kapitolách.

7.1 Vega Scanner

Webový testovací skener je open-source software vyvíjený kanadskou společností Subgraph. Pogram Vega je možné využít na všech operačních systémech. Na operační systém Windows je možné stáhnout instalační program, který jednoduše provede instalačním procesem. Druhou platformou, na které jsem program testoval, je Kali linux. Zde nebylo nutné nic instalovat, program již je v tomto systému předinstalován, stačí jej pouze spustit. Program je napsán v jazyku Java a umožňuje ověření odolnosti webových stránek například proti útokům SQL Injection, Cross-site scripting nebo testování před neoprávněným získáním citlivých informací, a spoustu dalších.

Skener jsem zkusil spustit nanečisto na náhodnou webovou stránku z obou operačních systémů a utvrdil jsem se v tom, že skener pracuje nezávisle na operačním systému, protože

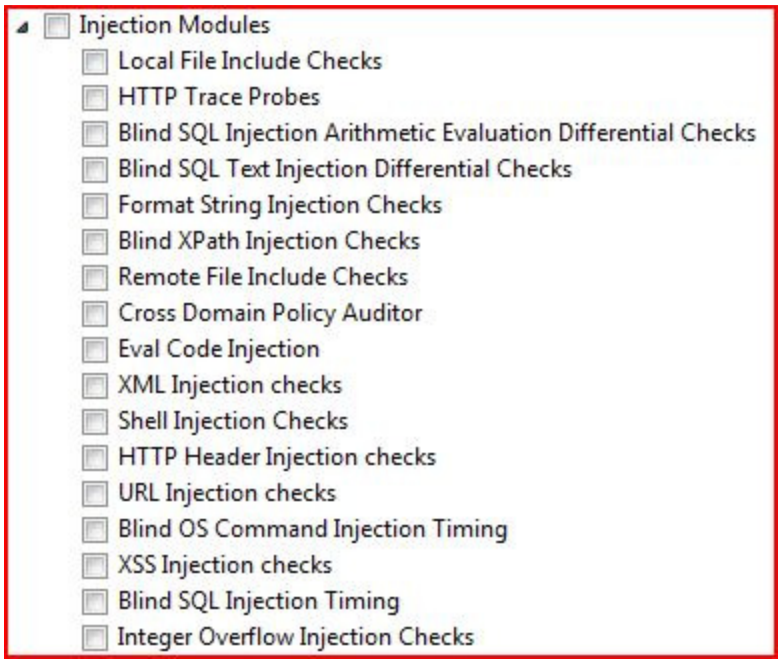
jsem v obou dosáhl totožných výsledků. Proto jsem následné testy prováděl na počítači s operačním systémem Windows 7, především kvůli většímu výkonu stroje.

Po spuštění se otevře hlavní okno programu, které je vytvořeno uživatelsky přívětivě, veškeré ovládání je velmi intuitivní. Po kliknutí na tlačítko *Start new scan* se otevře podokno se zadáním webové adresy pro testování. V další nabídce je již možné vybrat typy testů, které budou provedeny. Jsou rozděleny do dvou kategorií - moduly *Injection* a *Response Processing.*

Všechny hrozby, které jsou nalezeny pomocí testování, řadí Vega do čtyř kategorií podle míry zranitelnosti - *Vysoká, Střední, Nízká a Informativní*. Výběr jednotlivých modulů pro testování jsem prováděl pomocí pravděpodobnosti výskytu a míry zranitelnosti. Tudíž jsem provedl analýzu hrozeb, které jsou útočníky používány nejčastěji. Do konečného výběru jsem také vybíral pouze nejnebezpečnější útoky, přičemž jsem využil ohodnocení bezpečnosti podle skeneru Vega a zahrnul pouze útoky s označením *Vysoká a Střední.*

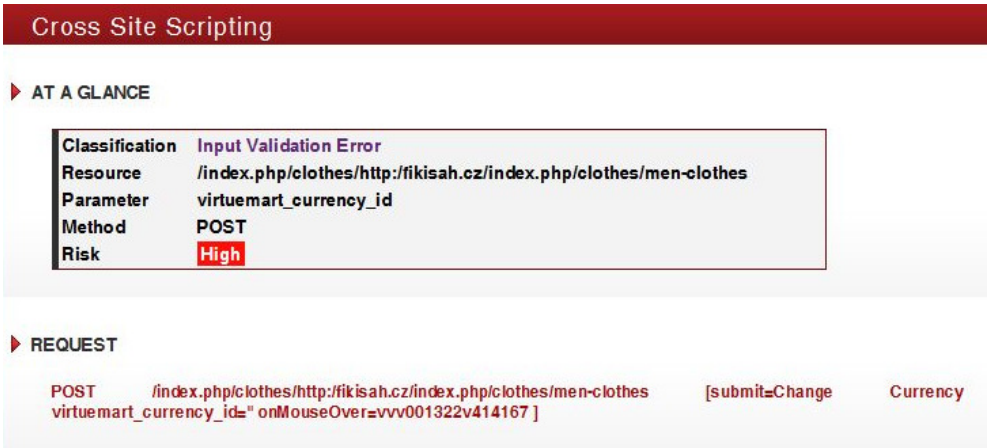
7.1.1 Injection Modules

Moduly Injection provádějí testování takzvaným „vsunutím“ nebezpečného kódu nebo skriptu do webové stránky. Takto je možné provést několik útoků a vložený škodlivý kód dokáže útočníkovi provést velké množství procesů. Vega skener obsahuje celkem 17 takových procesů, které jsou zobrazeny na obrázku 16. Pro potřeby testování jsem vybral 8 testovacích modulů, které dále podrobněji vysvětlím:



Obrázek 16: Injection Modules - Skener Vega

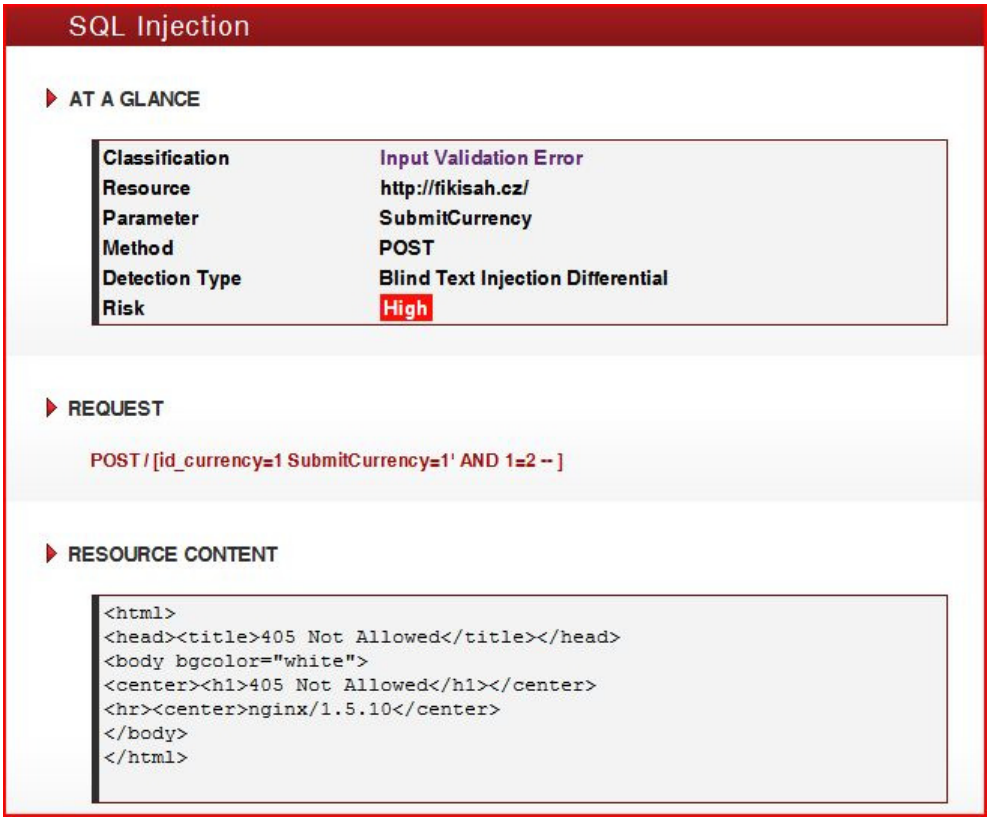
- Cross-site scripting
  - Míra nebezpečnosti - vysoká.
  - Odolnost vůči této nebezpečné hrozbě se testuje pomocí modulu *XSS Injection Checks*.



Obrázek 17: Úspěšný útok Cross-site scripting

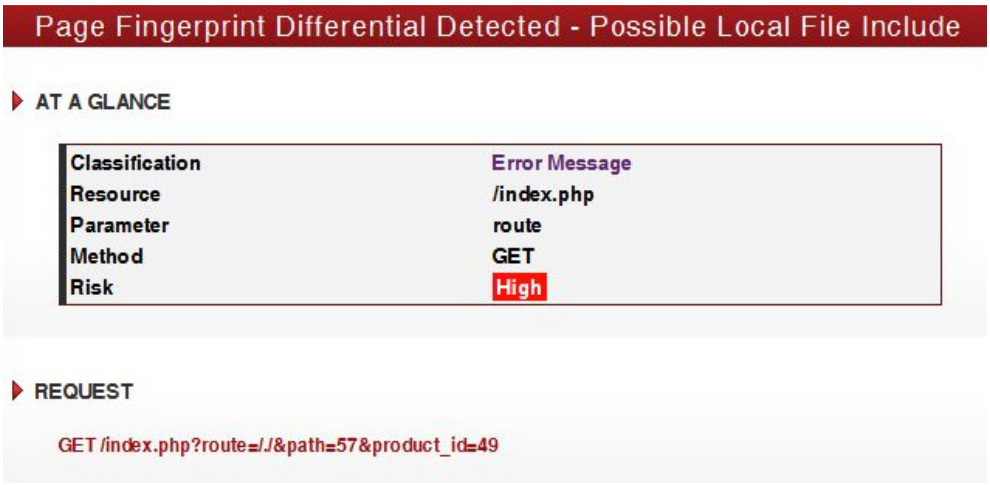
- SQL Injection
  - Míra nebezpečnosti - vysoká.

- Konkrétně se testuje útok typu Blind SQL, využity byly tři moduly *Blind SQL Injection Arithmetic Evaluation Differential Checks, Blind SQL Text Injection Differential Checks* a *Blind SQL Injection Timing.*



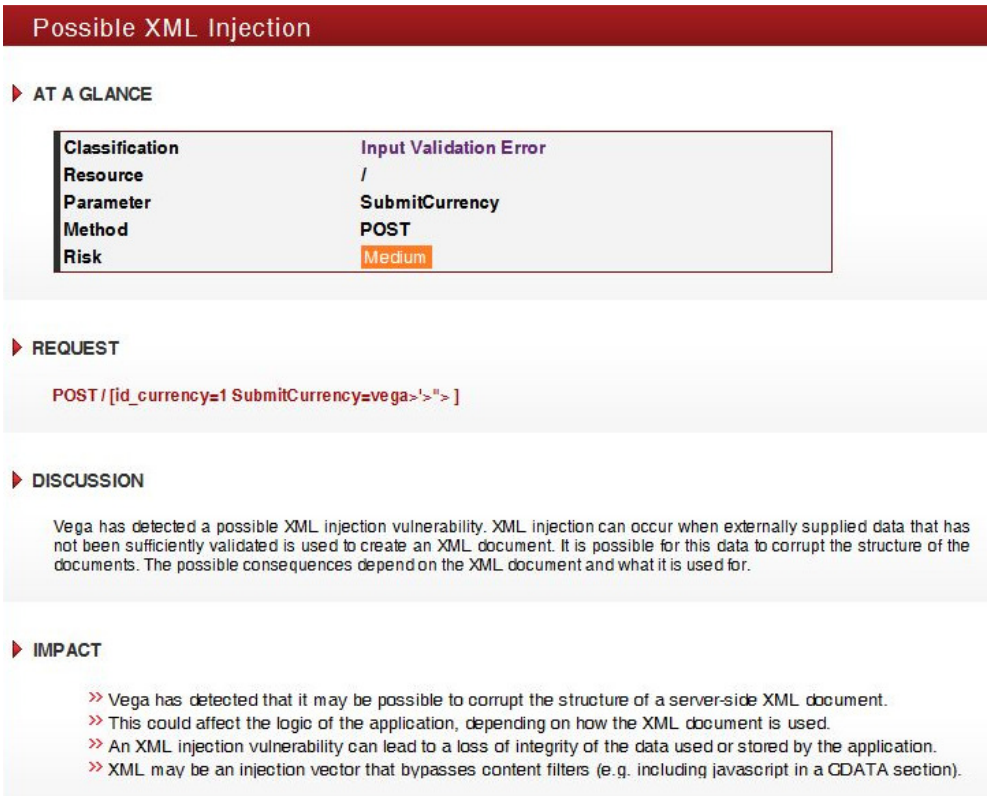
Obrázek 18: Úspěšný útok SQL Injection

- File Include
  - Míra nebezpečnosti - vysoká.
  - Testování odolnosti stránek vůči vložení souboru - například umožnění vložení celého souboru se skriptem a jeho následné použití. Vložením souboru může útočník dosáhnout ovládnutí celého webu, využití webu k útokům pomocí XSS nebo ovládnutí webu k následným DoS nebo DDoS útokům.
  - Vega skener obsahuje dva moduly - *Remote/Local File Include Checks*. Tyto útoky se projevují tak, že server buď načte vzdálený soubor zadaný útočníkem nebo lokální soubor.
  - Řešení: Důsledná kontrola souborů vkládaných do skriptů. U PHP je také možnost vypnout moduly *allow_url_fopen* a *allow_url_include* v nastavení *php.ini*.



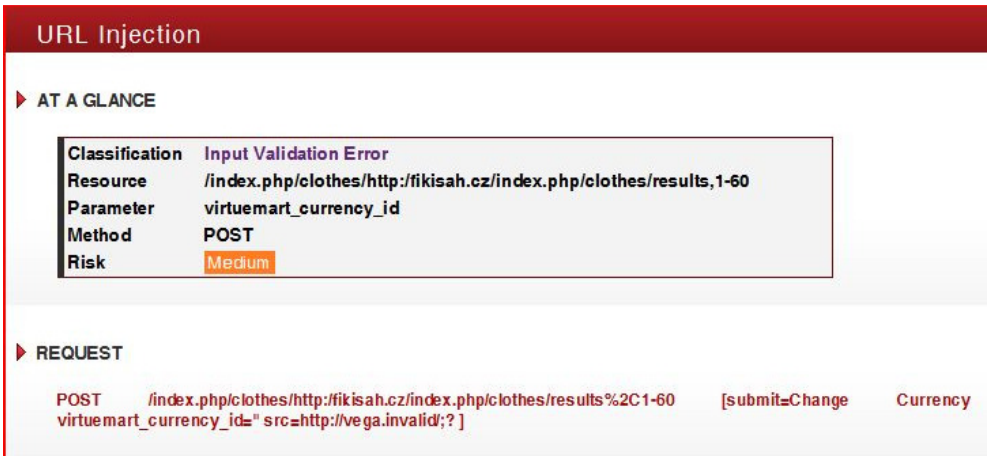
Obrázek 19: Úspěšný útok Local File Include

- XML Injection
  - Míra nebezpečnosti - střední.
  - Použití modulu *XML Injection Checks*.
  - Útoky mohou poškodit strukturu XML dokumentů na straně serveru. Tím může zničit celou logiku aplikace, porušit integritu dat nebo zničit veškerá data.
  - Řešení: Bezpečnostní řešení je velmi podobné jako u SQL Injection. To znamená především zabezpečit všechny vstupy a nahradit speciální znaky a bezpečné HTML entity.



Obrázek 20: Úspěšný útok XML Injection

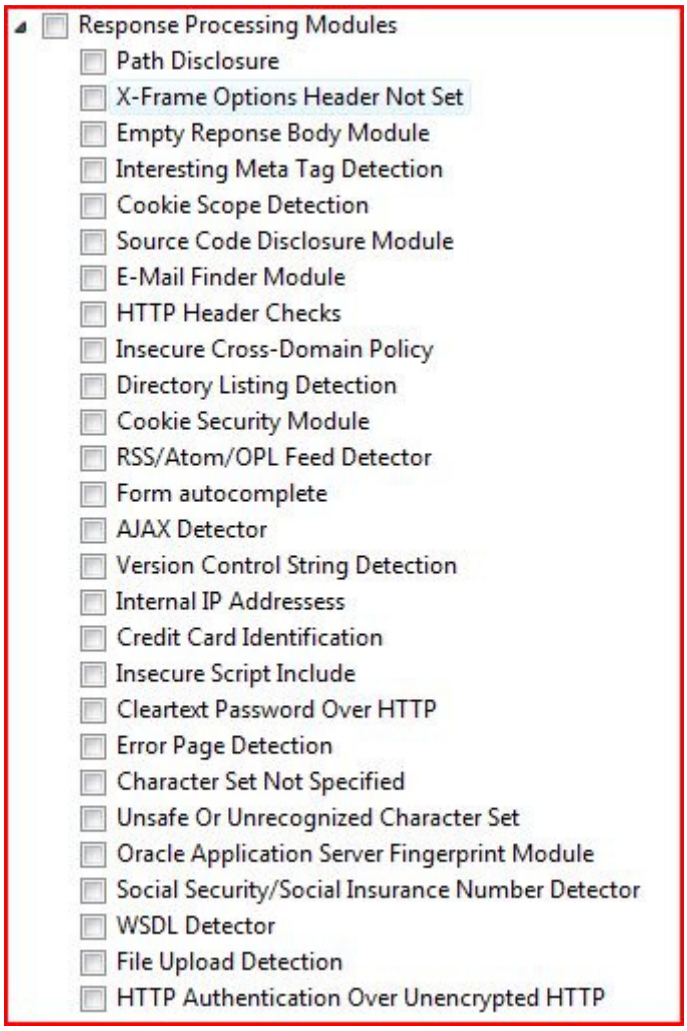
- URL Injection
  - Míra nebezpečnosti - střední.
  - Při neodladění různých vstupů je možné vsunout do kódu stránky odkaz na jinou nebezpečnou webovou stránku pomocí HTML tagu. Modul pro toto testování je nazván *URL Injection Checks*.
  - Řešení: Nahrazení speciálních znaků, např. <, >, “, ‘ za bezpečné HTML entity, které neumožní provedení vsunutí nebezpečného skriptu.



Obrázek 21: Úspěšný útok URL Injection

7.2 Response Processing Modules

Druhá skupina testovacích modulů je nazvaná jako *Response Processing*. Všechny takové útoky jsou vedeny za účelem zjištění co nejvíce informací o webové stránce, které poté může útočník využít ke snadnějšímu cílení jiných útoků. K výsledkům těchto útoků patří například vyčtené nezabezpečené heslo, zjištění struktury webu a názvů jednotlivých souborů. S takovými informacemi je pro útočníka mnohem jednodušší najít slabé místo stránky a například s pomocí útoků *Injection* z první skupiny modulů takových míst zneužít. Celkově v programu můžeme najít 27 takových modulů - zobrazených na obrázku 22, ze kterých jsem vybral 6, které jsem považoval za nejdůležitější a nejnebezpečnější.

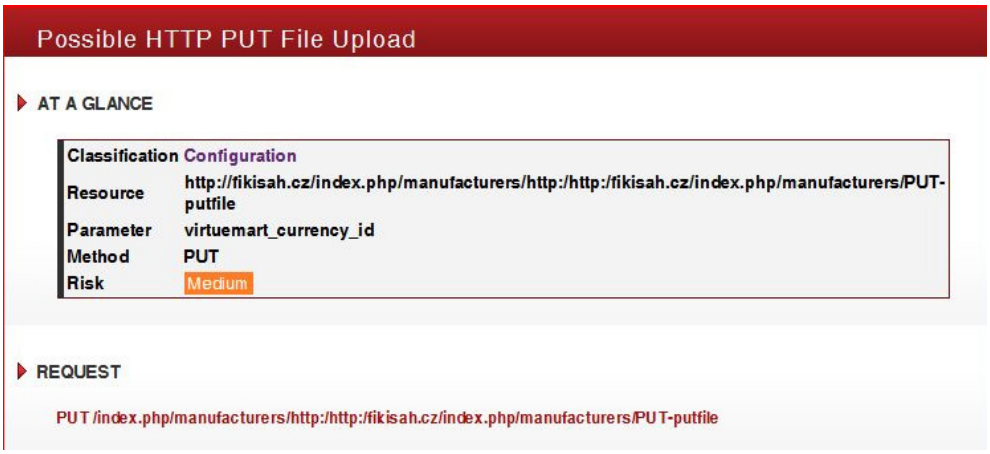


Obrázek 22: Response Processing Modules - skener Vega

- Full Path Disclosure
  - Míra nebezpečnosti - střední.
  - Test identifikuje strukturu webu a kompletní cestu k jednotlivým souborům. Modul je nazván *Path Disclosure.*

Obrázek 23: Úspěšný útok FPD

- AJAX detekce
  - Míra nebezpečnosti - střední.
  - Technologie AJAX slouží k interaktivnímu zobrazování webového obsahu, spolupracuje s JavaScriptem a technologií XML. Pokud není technologie správně zabezpečena, umožní například nahrát na web cizí soubor, který může porušit integritu celého systému, případně ovlivnit chod aplikací. Modul je nazván *AJAX detector*.
  - Řešení: Důsledná kontrola souborů vkládaných do skriptů. Zamezení vkládání souborů skrze vstupy a formuláře.



Obrázek 24: Úspěšný útok AJAX detection

- Nezabezpečené heslo
  - Míra nebezpečnosti - vysoká.
  - *Cleartext Password Over HTTP* je modul, který zjišťuje, jakým způsobem je zasíláno přihlašovací heslo po zadání a při jeho kontrole. Heslo nikdy nesmí být posíláno přes nezabezpečený protokol HTTP. Všechny operace s hesly a ostatními citlivými údaji by měly probíhat skrze bezpečný protokol HTTPS.

Obrázek 25: Úspěšný útok Cleartext Password

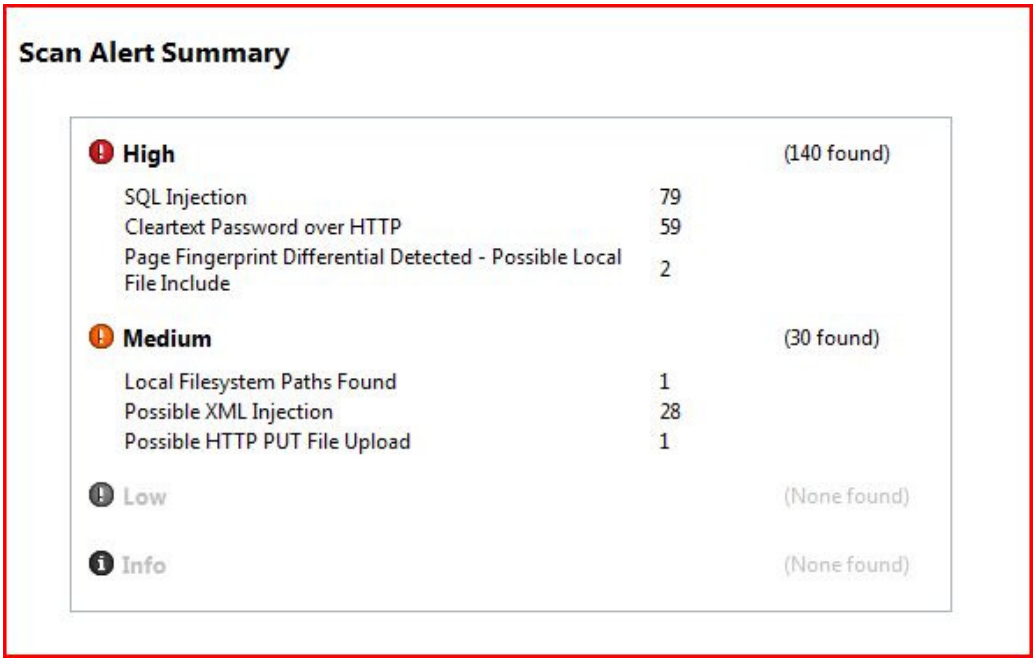
- Cookie soubory
  - Míra nebezpečnosti - nízká.
  - Cookie soubory jsou uloženy přímo v počítači a webové stránky si do ní můžou ukládat jakékoliv informace o návštěvníkovi daného webu - například jaké stránky pravidelně navštěvuje nebo jaké vyhledává informace. Tyto soubory lze poté využít v reklamních nabídkách, ale také při sociálním inženýrství. Míra nebezpečnosti není vzhledem k ostatním útokům příliš vysoká, vzhledem k rozšířenosti cookie souborů mne však zajímalo, zda weby dbají na jejich ochranu. Vega skener k tomu nabízí moduly *Cookie Scope Detection a Cookie Security Module.*
- Zobrazení výpisu chyb
  - Míra nebezpečnosti - střední.
  - Detekce souboru nebo stránky s výpisem chybových hlášek, které proběhly během běžného provozu je provedena pomocí modulu *Error Page*

Detection. Zobrazením chyb opět může útočník nalézt strukturu webu a jeho slabé místo.

7.2.1 Výsledky testování

Vzhledem k tomu, že každý e-shopový framework je jinak rozsáhlý, musel jsem stanovit testovací podmínky, aby byla výchozí pozice u všech testů stejná. Díky tomu mi vyšly výsledky, které je možné porovnat vzájemně vůči sobě.

Abych tedy docílil rovné podmínky pro všechny e-shopy, stanovil jsem pevný počet provedených útoků v rámci každého testu. Vycházel jsem z předchozích testů, které jsem prováděl kompletní na celé e-shopové frameworky, na všechny složky i podsložky. Vycházely mi však velice odlišné výsledky. Počet útoků jsem omezoval především kvůli e-shopu OpenCart, který je v porovnání s ostatními poměrně malý. K nejrozsáhlejším naopak patří PrestaShop a Magento, což lze vidět už z velikosti samotné instalační složky. Zatímco OpenCart zabírá velikost kolem 20 MB, složka Magenta má více než 100 MB. Počet testů na strukturu e-shopu jsem stanoval na 250. K tomuto číslu jsem došel jednoduchým porovnáním předchozích výsledků. Při nastavení nižšího počtu testů mohlo docházet k tomu, že některé problémy ještě nebyly nalezeny. Naopak větší číslo již nebylo možné použít vzhledem k tomu, že u e-shopu OpenCart již více testů nebylo možné provést.



Obrázek 26: Skener Vega - Kompletní test webu Magento

Co přesně jsem tímto číslem stanovil? Jednotlivé testy procházejí kompletní strukturu e-shopu a kontrolují náchylnost k bezpečnostním rizikům u všech nalezených odkazů a souborů. Tyto soubory jsou procházeny od nejvyšších adresářů směrem dolů, hlouběji ve stromové struktuře webu. I přesto, že jednotlivé testy nejsou provedeny kompletně na všechny soubory, vždy jsou nejdříve zkontrolovány ty nejdůležitější a nejnáchylnější.

Veškeré výsledky získané z testování jsem přehledně uvedl v tabulce . Na jednotlivé e-shopy bylo použito celkově 14 testovacích modulů, které jsem shrnul do 10 typů útoků. Jako nejvíce odolný vyšel z výsledků e-shopových frameworků OsCommerce, který byl náchylný pouze na tři testované hrozby. Také pouze tři útoky byly úspěšně provedeny u PrestaShopu, kde však byl nalezen větší počet chyb u jednotlivých testů. S poměrně dobrým výsledkem skončil také OpenCart, který však nedokázal zabránit mimo jiné útoku Cross-site scripting, který je velmi nebezpečný. Na samém konci, daleko za ostatními skončil VirtueMart, který je veden pouze jako doplněk redakčního systému Joomla. Tento e-shop dokázal odolat pouze 3 útokům.

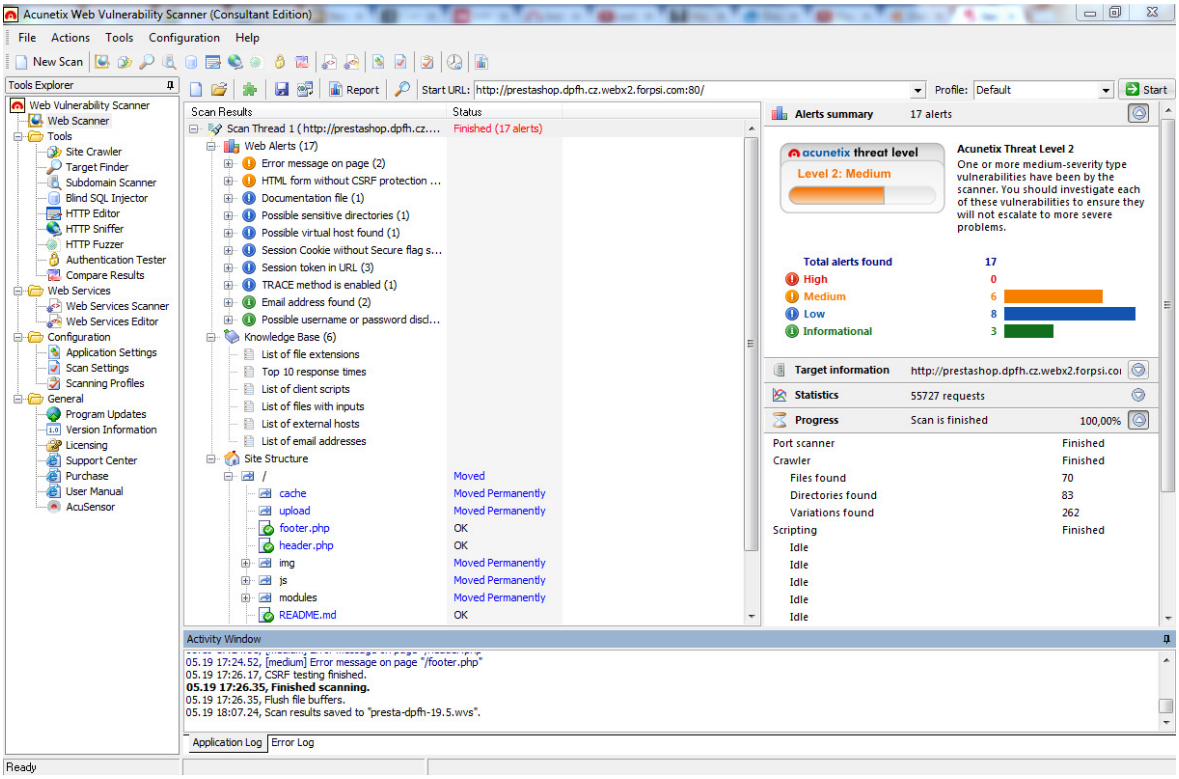
Při hodnocení bezpečnosti bez ohledu na typ e-shopu je na první pohled vidět velký bezpečnostní problém. Útok Blind SQL Injection byl nad síly všech ochranných prvků a s poměrně velkým procentem úspěšnosti všech provedených testů. Velmi podobně byl úspěšný útok pomocí XML Injection, proti kterému rovněž ani jeden nedokázal účinně chránit svá data. Zarážející je také odesílání hesla v nešifrované podobě pouze přes protokol HTTP u čtyř z pěti frameworků.

Tabulka 1: Skener Vega - Shrnutí výsledků

	PrestaShop	Magento	VirtueMart	OpenCart	OsCommerce
XSS Injection	0	0	3	1	0
SQL Injection	94	79	42	32	28
XML Injection	18	28	12	42	18
URL Injection	0	0	1	0	0
File Include	0	2	0	25	0
Full Path Disclosure	0	0	3	0	0
Detekce AJAX	0	1	1	0	0
Nezabezpečené heslo	1	59	58	0	2
Cookie soubory	0	0	1	0	0
Zobrazení výpisu chyb	0	0	0	0	0

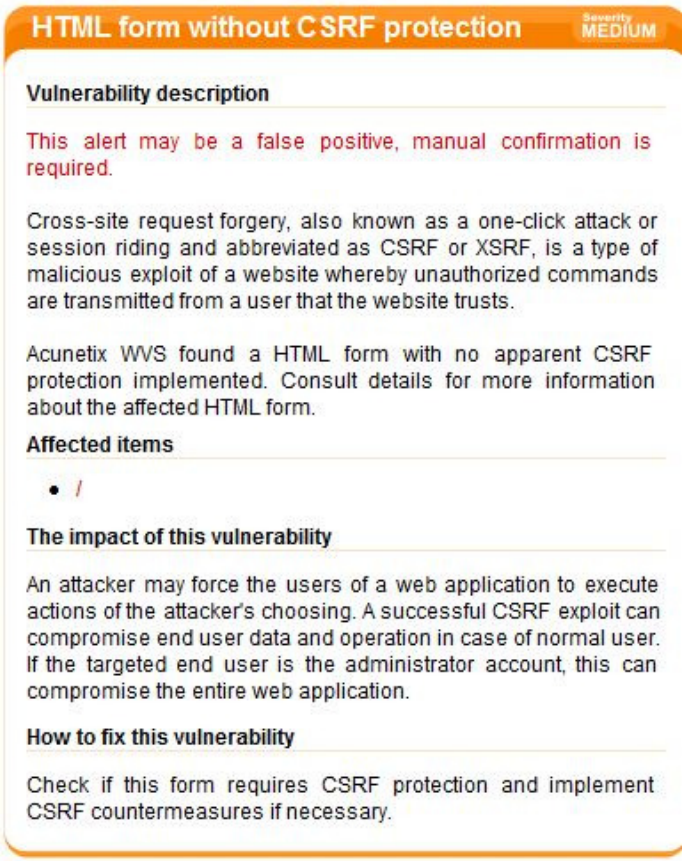
7.3 Acunetix Web Vulnerability Scanner

Acunetix je poměrně rozšířený testovací skener umožňující testování webových stránek, aby odhalil skryté zranitelnosti. Skener umožňuje ověřovat sílu hesel na autentizačních stránkách, zkoumá možnost „vsunutí“ nebezpečného skriptu do zdrojového kódu typu XSS nebo SQL. O nalezených zranitelnostech informuje v hlavním okně programu a také uvádí další informace o míře nebezpečnosti daného problému, rozsahu případných rizik při jeho použití a především návod, jak stránku zabezpečit, aby ji nebylo možné ohrozit. Vše je zobrazeno v přívětivém grafickém prostředí vytvořeném pouze pro operační systémy Windows.



Obrázek 27: Hlavní okno programu Acunetix

Díky tomu, že software testuje celý zdrojový kód a ne jenom samotné průnikové útoky, dokáže tak jistěji zobrazit danou chybu a dokonale tak minimalizuje hlášení falešných poplachů. Všechny tyto funkce jsou dostupné pouze v placené verzi. Ve zkušební 15-ti denní trial verzi jsou provedeny všechny testy, ale program nezobrazuje detailní hlášení o místu výskytu chyby a další podrobnosti. Z toho důvodu jsem tento skener použil pouze jako doplňkový a použil jsem testování pouze jednoho typu útoku, konkrétně CSRF, který jsem u konkurenčních skenerů nenašel, ale považoval jsem za důležité tento test provést.



Obrázek 28: Skener Acunetix - Úspěšný útok CSRF

7.3.1 Výsledky testování

Jak jsem již uvedl, skenování frameworků pomocí programu Acunetix jsem prováděl pouze na útok Cross-site request forgery. I přesto, že útok není zatím příliš známý, považuji ho za velmi nebezpečný a řadím na stejnou úroveň, jako útoky XSS a SQL Injection.

Bohužel opět musím konstatovat, že základní zabezpečení frameworků zcela propadlo a ani jeden nedokázal tomuto útoku odolat. V tabulce opět uvádím celkové shrnutí testů, čísla již však nesouvisí s předchozími testy u Vega skeneru. V nastavení Acunetixu nelze nastavit počet testů, proto jsou provedeny vždy na celou strukturu webu. Zde již tedy není srovnání tak jednoduché, ale opět musím vyzdvihnout OsCommerce s velmi malým počtem chyb. Nejmenší počet chyb (pouze 1) byl nalezen u e-shopu VirtueMart, bohužel se to však nedá považovat za úspěch, vzhledem k tomu, že jeho bezpečnostní štít opět utržil trhlinu a skóre útočníka proti obránci je navýšeno na 9:2.

Tabulka 2: Skener Acunetix - Úspěšný útok CSRF

	PrestaShop	Magento	VirtueMart	OpenCart	OsCommerce
CSRF Detection	6	11	1	8	2

7.4 Nikto2 Web Scanner

Další open source aplikace pro automatické testování zranitelnosti webových serverů se nazývá Nikto2 Web Scanner. Tento skener je již nainstalován v operačním systému Kali linux a jednoduchým příkazem jej ihned můžeme spustit. Na rozdíl od předchozích dvou skenerů je Nikto2 textová aplikace dostupná pouze v unixových platformách a je spouštěn přímo z terminálu.

Skener umožňuje komplexní testování serverů na 6 500 možných zranitelností a také je schopen ověřit zastaralost daného typu serveru, případně zda je možné jej aktualizovat. K dalším vlastnostem lze jmenovat kontrolu konfigurace serveru. Přitom je možné zjišťovat například vícenásobnou indexaci, samotnou identifikaci typu webového serveru nebo možnosti protokolu HTTP. Veškeré události lze ukládat do logu a následně si je zobrazit, případně vytisknout.

Skener jsem použil spíše jako doplňkový pro ověření testování zranitelnosti proti XSS Injection.

```
root@kali:~# nikto -host www.fikisah.cz
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          88.86.121.75
+ Target Hostname:    www.fikisah.cz
+ Target Port:        80
+ Start Time:         2014-05-26 02:23:37 (GMT2)
---------------------------------------------------------------------------
+ Server: nginx/1.5.10
+ Retrieved x-powered-by header: PHP/5.5.11
+ The anti-clickjacking X-Frame-Options header is not present.
+ Root page / redirects to: http://fikisah.cz/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server leaks inodes via ETags, header found with file /, fields: 0x5322e1c8 0x
219
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 6544 items checked: 0 error(s) and 5 item(s) reported on remote host
+ End Time:           2014-05-26 02:45:40 (GMT2) (1323 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Obrázek 29: Skener Nikto2

7.4.1 Výsledky testování

Skener Nikto2 vypisuje výsledky testů přímo v okně terminálu, což bohužel není úplně šťastné řešení, protože textový výpis je velice nepřehledný. Navíc zde nenalezneme žádné shrnutí výsledků, jako tomu bylo u výše uvedených skenerů. Textový výpis je tedy nutné číst pozorně, řádek po řádku. Zaměřil jsem se u něj tedy především na útok Cross-site scripting. Při pročítání výpisu však lze vidět další problémy, které již byly zmíněny, například nešifrovaný přenos hesla, možnost neoprávněného vložení souboru nebo viditelnost struktury souborů.

Při ověření výsledků útoku XSS můžeme kromě jedné výjimky potvrdit předchozí údaje. Výjimku můžeme vidět u frameworku Oscommerce, kde skener Nikto na rozdíl od Vegy nalezl zranitelnost. Údaje jsou opět zaneseny do tabulky, pro srovnání jsou ponechány výsledky z prvního testu.

Tabulka 3: Srovnání výsledků u XSS Injection

XSS Injection	PrestaShop	Magento	VirtueMart	OpenCart	OsCommerce
Vega skener	ne	ne	ano	ano	ne
Nikto2 skener	ne	ne	ano	ano	ano

7.5 Srovnání a doporučení

Bohužel musím říci, že výsledky testování mne velmi zklamaly. Vzhledem k tomu, že jde o nejpoužívanější e-shopové frameworky, výsledky testů zranitelností jsou hodně tristní. Přitom si myslím, že zabezpečení jednotlivých hrozeb není vůbec složité. Nejvíce překvapivý výsledek je pro mne u testu na SQL Injection. Počet možných míst průniku je alarmující. A to u všech frameworků.

Druhý velký problém byl u XML Injection. Zřejmě musíme vzít v potaz, že tyto útoky jsou na sebe poměrně podobné. Z toho vyplývá, že by se e-shopy měly zaměřit na všechny útoky, které útočí pomocí vsunutí souboru nebo části kódu přes nezabezpečené vstupy. Myslím, že by měla být od jednotlivých e-shopů přijata přísnější bezpečnostní opatření, protože tento problém vychází ze samotné implementace kódu.

Všechny typy útoků, které jsem prováděl, jsem podrobně popsal. Dále jsem u každého testu uvedl, jaké hrozby mohou nastat a v neposlední řadě, jak dané hrozby vyřešit. Jde tady

vidět, jak lehce lze obejít bezpečnostní pravidla. Skenery pro testování bezpečnosti jsou dostupné komukoliv a určitě nebude problém sehnat i jiné utilitky, které už si na útoky nebudou pouze hrát, ale budou je provádět plnou silou.

Pokud bych měl zhodnotit jednotlivé e-shopy a doporučit nejvhodnější řešení, musím uvést dva názvy. Pro menší a jednodušší projekty určitě doporučím OsCommerce. Pracovalo se s ním poměrně intuitivně a především není tak rozsáhlý jako všechny ostatní. Už od samotného přenosu souborů na web přes instalaci až po užívání obchodu vše proběhlo velmi rychle a bez problémů. A především je nutné zmínit, že ze všech frameworků dopadl v bezpečnosti daleko nejlépe.

Jako druhý bych doporučil PrestaShop. Tento e-shop již je rozsáhlejší, tudíž bude vhodnější pro větší projekty. Především mne zaujal instalační proces, který v každém kroku napovídal, jak pokračovat dál. I celkové prostředí obchodu bylo velmi líbivé. Z hlediska bezpečnosti na tom byl podobně jako OsCommerce, byly na něj úspěšné pouze 4 typy útoků, ale u každého útoku bylo zasaženo mnohem více souborů.

ZÁVĚR

V teoretické části byly popsány základní pojmy z oblasti bezpečnosti a z oblasti PHP e-shopových frameworků, aby byly pochopeny jednoduché příklady uvedené v praktické části práce. Dále byly ukázány nejpoužívanější e-shopové systémy a nejpoužívanější PHP frameworky, na kterých jsou jednotlivé e-shopy postaveny.

Práce se dále věnovala všeobecnému popisu a rozdělení bezpečnostních penetračních testů. Pomocí automatizovaných programů byly jednotlivé e-shopy testovány na jejich náchylnost vůči různým útokům. Testy byly rozděleny do dvou kategorií, na takzvané Injection Moduly, které útočí na weby takovým způsobem, že se přes nedokonale zabezpečené vstupy snaží „vsunout“ určitou část kódu nebo skriptu nebo celého souboru. Takové útoky mají různé následky. Od těch nejmenších, kdy se útočníkovi maximálně ukáže část struktury souborů, až po největší, které mohou způsobit ovládnutí celého webu, případně likvidace všech dat, které na něm byly uloženy. Obecně jsou tyto útoky vedené jako velmi vysoká hrozba. A bohužel jsem zjistil, že oprávněně. E-shopy si s těmito útoky vůbec nedokázaly poradit. Dovolím si říct, že všechny e-shopy z hlediska zabezpečení propadly. Ale samozřejmě nemůže být všechna práce ohledně zabezpečení na bedrech tvůrců. Uživatel by si také měl zjistit určité informace o tom, jak dosáhnout větší bezpečnosti. Určitě každý nalezne spoustu informací a možných nastavení, jak se takovýmto problémům vyhnout.

Druhou skupinou útoků jsou Response Processing Modules, které se soustředí především na vytěžování důležitých informací. Testy mají za úkol zjistit co nejvíce detailů, aby potom měl útočník co nejlepší přehled o struktuře webu oběti. Protože s těmito informacemi pro něj bude velmi jednoduché použít jiné testy, které již budou provádět škodlivé akce. Odolnost vůči těmto útokům byla o hodně lepší, než u první skupiny. Ale i přesto se našel útok, který nám ukázal, že je potřeba se zabezpečením něco dělat. Útok byl prováděn tak, aby mohl útočník zjistit heslo do citlivých částí webu. Cože se ve většině případů bohužel povedlo.

Největší problémy se zabezpečením mají e-shopy Magento a především VirtueMart. Tento e-shop, který vychází ze systému Joomla propadl téměř ve všech prováděných testech. Proto jej musím označit jako nejhorší možnou volbu.

Hlavním cílem práce bylo testování bezpečnosti e-shopových frameworků a navrhnutí vhodných opatření. U všech útoků jsem uvedl řešení, které by mělo tato bezpečnostní rizika vyřešit.

Bohužel všichni útočníci jsou velmi nevyzpytatelní a nikdy nemůžeme tušit, co všechno nás může čekat. Bezpečnost se stále vyvíjí, ale vyvíjí se především kvůli tomu, že se má vyvíjet kvůli čemu. Protože útočníci vždy budou o krok napřed, vždy dokáží sami nalézt volnou neza-

bezpečenou cestičku skrze sebelepší ochranu. Nezbývá než doufat, že se to nestane zrovna nám.

SEZNAM POUŽITÉ LITERATURY

[1] KOSEK, Jiří. PHP. Hypertextový preprocesor. In: Kosek [online]. Feb 12, 2009, 3 pm [cit. 2014-05-20]. Dostupné z: http://www.kosek.cz/php/

[2] LACKO, Ľuboslav. PHP 5 a MySQL 5: hotová řešení. Vyd. 1. Brno: Computer Press, 2007, 320 s. ISBN 978-80-251-1695-1.

[3] TVORBA-WEBU.CZ. Java script [online]. © 2003 - 2008 [cit. 2014-05-22]. Dostupné z: http://www.tvorba-webu.cz/javascript/

[4] SLEZÁK, Zbyněk. Bezpečnostní audity a pentesting ve firemním prostředí. Zlín, 2013. Diplomová práce. Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky.

[5] PASTUCHOVÁ, Markéta. Open source přebírá v oblasti softwaru klíčovou roli. In: ICT manažer [online]. 2011 [cit. 2014-05-20]. Dostupné z: http://www.ictmanazer.cz/2011/11/open-source-prebira-v-oblasti-softwaru-klicovou-roli/

[6] OPERAČNÍ SYSTÉM GNU. Definice svobodného software [online]. 2011 [cit. 2014-05-21]. Dostupné z: http://www.gnu.org/philosophy/free-sw.cs.html

[7] GREZO, Peter. Zend Framework: Úvod do frameworku. In: Zdroják.cz [online]. Aug. 6, 2012 [cit. 2014-05-21]. Dostupné z: http://www.zdrojak.cz/clanky/zend-framework-uvod-do-frameworku/

[8] AICHINGER, Michal. PHP framework Prado. In: Aichiho blog [online]. Apr. 27, 2008. [cit. 2014-05-22]. Dostupné z: http://www.czechdesign.cz/blogs/aichi/php-framework-prado

[9] KOUTNÝ, Jiří. Symfony2, těší mě!. In: Zdroják.cz [online]. Jun. 26, 2013. [cit. 2014-05-23]. Dostupné z: http://www.zdrojak.cz/clanky/symfony2-tesi-me/

[10] NETTE. Seznámení s Nette Frameworkem [online]. © 2008, 2014 [cit. 2014-05-20]. Dostupné z: http://doc.nette.org/cs/2.1/getting-started

[11] JANOVSKÝ, Dušan. Jakpsatweb.cz [online]. c2010 [cit. 2014-05-19]. Domény. Dostupné z: http://www.jakpsatweb.cz/domeny.html

[12] MAGENTO.CZ. Magento se představuje [online]. 2014 [cit. 2014-05-20]. Dostupné z: http://prirucka.magento.cz/magento-se-predstavuje

[13] HEJDA, Václav. ZenCart – tak trochu rozporuplný e-shop. In: Linuxexpres [online]. May 13, 2013 [cit. 2014-05-21]. Dostupné z: http://www.linuxexpres.cz/software/zencart-tak-trochu-rozporuplny-e-shop

[14] PRESTASHOP. About PrestaShop: The world's best free e-commerce software! [online]. © 2007-2014 [cit. 2014-05-22]. Dostupné z: http://www.prestashop.com/en/about-us

[15] KOSEK, Jiří. Bezpečnost webových aplikací [online]. © 2000-2013 [cit. 2014-05-22]. Dostupné z: http://www.kosek.cz/vyuka/4iz228/prednasky/bezpecnost.pdf

[16] ŠEBEK, Jiří. Webová aplikace (e-shop) pro využití v obchodní organizaci. Praha, 2012. Bakalářská práce. České vysoké učení technické v Praze. Fakulta elektrotechnická.

[17] PŘIBYL, Tomáš. Sedm z deseti webů je zranitelných. XSS – skriptování napříč servery. In: SystemOnLine [online]. 2008 [cit. 2014-05-22]. Dostupné z: http://www.systemonline.cz/it-security/xss-skriptovani-napric-servery.htm

[18] PEJŠA, Jan. Co je Cross-site scripting jak mu předcházet. In: Zdroják.cz [online]. Mar. 5, 2009 [cit. 2014-05-22]. Dostupné z: http://www.zdrojak.cz/clanky/co-je-xss-jak-mu-predchazet/

[19] MALÝ, Jan a Jan Kacálek. Zabezpečení webových aplikací I. - klientské skriptovací jazyky. In: Access Server [online]. Aug. 15, 2007 [cit. 2014-05-22]. Dostupné z: http://access.feld.cvut.cz/view.php?cisloclanku=2007090001

[20] NOVÁK, Tomáš. Bezpečnost open source redakčních systémů se zaměřením na CMS Joomla. Zlín, 2013. Diplomová práce. Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky

[21] ŠPAČEK, Michal. FPD aneb Full Path Disclosure. In: DevBlog [online]. Apr. 29, 2012 [cit. 2014-05-23]. Dostupné z: http://devblog.cz/2012/04/fpd-aneb-full-path-disclosure/

[22] JURÁK, Martin. Zabezpečení a optimalizace webových stránek v internetu. Zlín, 2010. Diplomová práce. Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AJAX	Asynchronous JavaScript and XML
CSRF	Cross-Site Request Forgery
CSS	Cross-Site Script
DDoS	Distributed Denial-of-Service
DoS	Denial-of-Service
HTML	HyperText Markup Language
FPD	Full Path Disclosure
ID	Identification Number
PHP	PHP: Hypertext Preprocessor
SQL	Structured Query Language
URL	Uniform Resource Locator
XML	Extensible Markup Language
XSRF	Cross-Site Request Forgery
XSS	Cross-Site Script

SEZNAM OBRÁZKŮ

SEZNAM TABULEK