

Testování bezpečnosti webových aplikací

Security testing of web application

Petra Holbíková

Bakalářská práce
2012



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2011/2012

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Petra HOLBÍKOVÁ**
Osobní číslo: **A08224**
Studijní program: **B 3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**

Téma práce: **Testování bezpečnosti webových aplikací**

Zásady pro vypracování:

1. Proveďte analýzu současných rizik, týkajících se webových aplikací.
2. Vytvořte přehled dostupných testovacích nástrojů.
3. Zhodnoťte možnosti automatizovaného a manuálního testování.
4. Vytvořte ukázkovou webovou aplikaci (open-source technologie PHP, MySQL) pro demonstraci nejčastějších rizik a scénářů penetračních testů.

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. HOWARD, Michael a David LEBLANC. **Bezpečný kód: Ittechniky a strategie tvorby bezpečných webových aplikací**. Vyd. 1. Brno: Computer Press, 2008, 895 s. ISBN 978-802-5120-507.
2. CROSS, Michael. **Developer's guide to web application security**. Rockland, MA: Syngress Publishing, c2007, 489 s. ISBN 159749061X.
3. STUTTARD, Dafydd a Marcus PINTO. **The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws**. Indianapolis: John Wiley & Sons, 2011, 489 s. ISBN 1118079612.
4. HOFFMAN, Billy a Bryan SULLIVAN. **Ajax Security**. Boston: Addison-Wesley Professional, 2007. ISBN 0132701928.
5. SOLARI, Carlos Curtis. **Security in a Web 2.0+ World: A Standards-Based Approach**. Chichester: John Wiley & Sons, 2011. ISBN 0470971088.
6. SHIFLETT, Chris. **Essential PHP security**. Sebastopol: O'Reilly, 2006, 109 s. ISBN 05-960-0656-X.
7. OWASP FOUNDATION. **OWASP Foundation [online]**. 13. 1. 2012 [cit. 2012-01-19]. Dostupné z: www.owasp.org

Vedoucí bakalářské práce:

Ing. Radek Vala

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

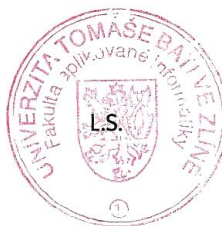
24. února 2012

Termín odevzdání bakalářské práce:

8. června 2012

Ve Zlíně dne 24. února 2012

prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Holbiková, P. Testování bezpečnosti webových aplikací. Bakalářská práce. Zlín, 2012.

Hlavním cílem bakalářské práce je pojednání o současných bezpečnostních rizicích, dostupných testovacích nástrojích a také o testování bezpečnosti webových aplikací. V teoretické části práce nalezneme analýzu bezpečnostních rizik a testovacích nástrojů, dále pak zhodnocení manuálního a automatizovaného testování webových aplikací. V druhé části práce, praktické části, nalezneme popis vytvořené webové aplikace, jež demonstruje nejčastější bezpečnostní slabiny a způsoby obrany proti nim.

Klíčová slova: bezpečnost, webová aplikace, útoky, bezpečnostní rizika, testovací nástroje, hacking,OWASP

ABSTRACT

Holbiková, P. Security testing of web applications. Bachelor's thesis, Zlín, 2012.

The main topics of this thesis are the today's security risks, available testing tools and also security testing of web applications. The theoretical part of this thesis contains analysis of security risks and their testing tools. Manual and automatic web application testing is also included. The self made web application is described in second part of the thesis (practical part) It is demonstrating the most common security weaknesses and ways to defense against them.

Keywords: security, web application, attack, security risks, testing tools, hacking,OWASP

Na tomto místě bych ráda poděkovala mému vedoucímu bakalářské práce Ing. Radku Valovi za jeho podnětné připomínky, ochotu při konzultacích, podporu a trpělivost při psaní této bakalářské práce.

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

OBSAH

ÚVOD	10
I TEORETICKÁ ČÁST	11
1 ANALÝZA SOUČASNÝCH RIZIK TÝKAJÍCÍCH SE WEBOVÝCH APLIKACÍ	12
1.1 INJECTION	15
1.2 CROSS-SITE SCRIPTING	16
1.2.1 Stored (Persistent)	16
1.2.2 Reflected (Non-persistent)	18
1.2.3 DOM based XSS	19
1.3 BROKEN AUTHENTICATION AND SESSION MANAGEMENT	19
1.4 INSECURE DIRECT OBJECT REFERENCES	20
1.5 CROSS – SITE REQUEST FORGERY	21
1.6 SECURITY MISCONFIGURATION	22
1.7 INSECURE CRYPTOGRAPHIC STORAGE	23
1.8 FAILURE TO RESTRICT URL ACCESS	23
1.9 INSUFFICIENT TRANSPORT LAYER PROTECTION	24
1.10 UNVALIDATED REDIRECTS AND FORWARDS	24
1.11 SHRNUÍ KAPITOLY	25
2 ANALÝZA DOSTUPNÝCH TESTOVACÍCH NÁSTROJŮ	26
2.1 OWASP ZAP - ZED ATTACK PROXY	27
2.2 SELENIUM IDE.....	29
2.3 SKIPFISH	29
2.4 NIKTO2.....	30
2.5 WEBSECURIFY - WEBSITE SECURITY TESTING TOOL.....	31
2.6 DALŠÍ TESTOVACÍ NÁSTROJE.....	32
2.6.1 Nmap.....	32
2.6.2 WireShark.....	32
2.6.3 Backtrack 5	32
3 MOŽNOSTI AUTOMATIZOVANÉHO A MANUÁLNÍHO TESTOVÁNÍ.....	34
3.1 MANUÁLNÍ TESTOVÁNÍ.....	35
3.2 AUTOMATIZOVANÉ TESTOVÁNÍ.....	35
3.3 HODNOCENÍ RIZIK	36
3.3.1 DREAD.....	36
3.3.2 OWASP Risk Rating Methodology	36
3.3.2.1 Identifikace rizika.....	36
3.3.2.2 Faktory pro odhad pravděpodobnosti.....	36

3.3.2.3	Faktory pro odhad dopadu	37
3.3.2.4	Určení závažnosti rizika	38
3.3.2.5	Rozhodování o tom, co opravit	38
II	PRAKTICKÁ ČÁST	39
4	ÚVOD DO PRAKTICKÉ ČÁSTI.....	40
5	VYUŽITÉ PROGRAMOVACÍ JAZYKY A TECHNOLOGIE.....	41
5.1	HYPERTEXT MARKUP LANGUAGE	41
5.1.1	Historie.....	41
5.2	PHP.....	42
5.2.1	Historie.....	42
5.3	KASKÁDOVÉ STYLY	43
5.3.1	Historie.....	43
5.4	MYSQL.....	43
5.5	PHPMYADMIN.....	44
5.6	APACHE HTTP SERVER.....	44
6	VLASTNÍ POPIS APLIKACE	45
6.1	POPIS LAYOUTU.....	45
6.2	STRUKTURA WEBOVÉ STRÁNKY	46
6.3	POPIS JEDNOTLIVÝCH STRÁNEK APLIKACE.....	46
6.3.1	SQL Injection	47
6.3.2	Cross – Site Scripting.....	49
6.3.3	Failure To Restrict URL Access	51
6.3.4	Unvalidated Redirect.....	51
6.3.5	Insecure Objects.....	52
6.3.6	CSRF.....	53
6.4	OBNOVA DATABÁZE	54
7	PENETRAČNÍ TESTOVÁNÍ WEBOVÉ APLIKACE	55
7.1	SCÉNÁŘ PENETRAČNÍHO TESTOVÁNÍ	55
7.1.1	Sběr informací.....	55
7.1.2	Mapování sítí	55
7.1.3	Automatizované testy slabin.....	56
7.1.4	Manualní testy.....	56
7.1.5	Vyhodnocení bezpečnostních slabin.....	56
	ZÁVĚR	57
	CONCLUTION	58
	SEZNAM POUŽITÉ LITERATURY	59
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	62
	SEZNAM OBRÁZKŮ	64
	SEZNAM TABULEK	65

SEZNAM PŘÍLOH 67

ÚVOD

S velkým rozvojem programovacích jazyků přicházejí nové možnosti jak prezentovat a spravovat informace. Oblast programování webových aplikací je poměrně novým oborem jež se vyznačuje velmi progresivním a rychlým rozvojem. Česká Republika byla do internetové sítě zapojena začátkem roku 1992, kdy jako první byly připojeny univerzitní sítě. K dnešnímu datu, tedy o 20 let později, má v ČR přístup k internetu prakticky kdokoliv. Jen v naší republice bylo v roce 2011 připojeno přes 7,2 miliónu uživatelů. Víceméně skoro každá firma má svou prezentaci na internetu, spousta uživatelů provozuje své osobní stránky, můžeme na internetu nakupovat nebo spravovat bankovní transakce a další.

Tyto obrovské možnosti nabízejí také velké riziko zneužití. Nelegální průniky do aplikací a neoprávněné přístupy k citlivým datům jsou v poslední době diskutované téma. O bezpečnost svých webových aplikací se zajímá stále více osob. Testování bezpečnosti je odvětví, jehož úkolem je testovat aplikace a doporučovat způsoby jak se rizikům bránit. Mezi důsledky těchto rizik můžeme zařadit škody s více či méně zásadními dopady od odcizení citlivých dat, přes zcizení či zničení aplikace, až po velké finanční ztráty či poškození dobrého jména.

Cílem této bakalářské práce je představit a poukázat na některá bezpečnostní rizika. Práce nastiňuje problematiku bezpečnosti aplikací v prostředí WWW. V jednotlivých kapitolách se dozvídáme o nejčastějších bezpečnostních rizicích a chybách, jaké jsou možnosti zneužití a doporučení, jak se proti nim bránit. Dále se dovídáme o dostupných testovacích nástrojích a jejich využití v oblasti testování. V praktické části práce je vytvořena a popsána ukázková aplikace, ve které je poukázáno na některá bezpečnostní rizika. Praktická práce je naprogramována tak, aby ukazovala reálné možnosti zneužití bezpečnostních chyb a také návrh, na jejich opravu.

Tato práce rozšiřuje vědomosti získané během studia na vysoké škole, zejména z předmětů o databázích, webových technologiích a kryptologii. Znalosti získané během vytváření bakalářské práce rozšiřují základní možnosti programování řádně zabezpečených webových aplikací, které jsou pro trh velmi žádané.

I. TEORETICKÁ ČÁST

1 ANALÝZA SOUČASNÝCH RIZIK TÝKAJÍCÍCH SE WEBOVÝCH APLIKACÍ

V dnešní době (k roku 2012) se k veřejnosti skrz massmedia dostávají informace o útocích na webové aplikace stále častěji. V čele těchto útoků jsou nejznámější hackerské skupiny – Anonymouse, LulzSec a Legion of Doom. Jejich masové základny hackerů ale využívají spíše DDoS útoky. Tento útok nespočívá ani tak v preciznosti hledání chyb vývojářů, jako spíše v jednoduchosti odeslání velkého množství jednoduchých dotazů. Webový server neunes obrovské množství TCP/IP spojení a aplikace se stává nedostupnou ostatním uživatelům.

K posledním nejznámějším DDoS útokům patří Operation Payback, útoky na servery FBI, Bílého Domu, Universal Studios. [9] V České Republice se skupina Anonymouse přihlásila k útokům na weby OSA, České vlády a weby politických stran. [8]



Obr. 1. Vlajka hackerské skupiny Anonymouse [9]

LulzSec nevyužívá prvoplánově DDoS útoky, na rozdíl od Anonymouse. Mezi jejich známé počiny patří prolomení databází PBS, CIA, Sony Pictures, Sony Playstation Network a Sony Music Entertainment. [10]



Obr. 2. Logo hackerské skupiny

LuSec [10]

Tyto činy hackerských skupin nejsou prováděny za účelem finančního profitu, ale jako protest a odpověď na cenzuru internetu v dokumentu ACTA, pozastavení serveru Megaupload, The Pirate Bay a další akce a dokumenty spojené s internetovým prostředím.

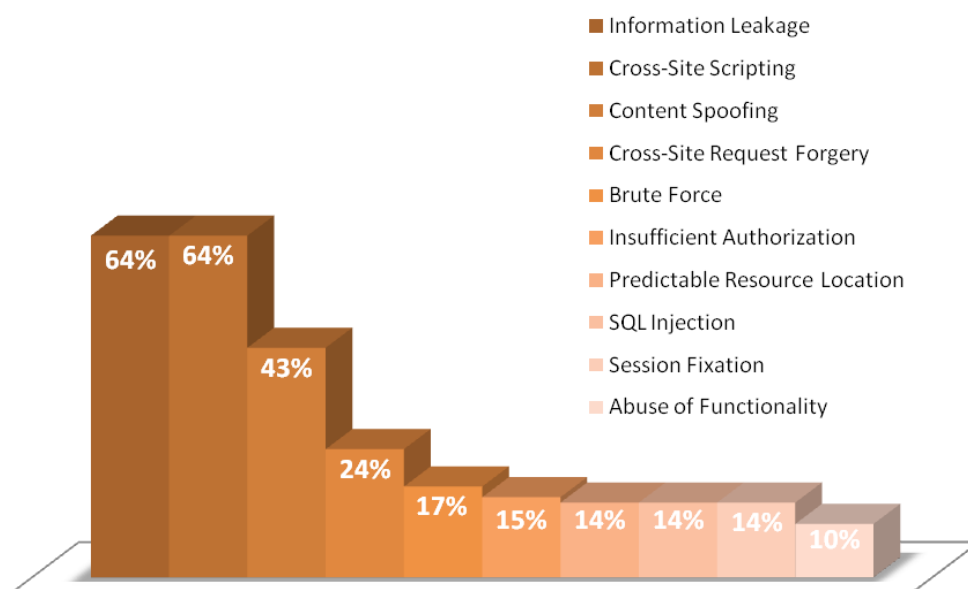
Pokud opomeneme DoS útoky, které jsou zaměřeny spíše na diskonfiguraci HW než na webovou aplikaci jako takovou, existuje řada slabín v bezpečnosti webových aplikací, které jsou povětšinou chybou v kódu, nebo nevědomost vývojáře. Tato rizika analyzuje projekt OWASP, který v tříletých intervalech (2004, 2007, 2010) vydává žebříček nejrizikovějších chyb.

Tab. 1. Žebříčky OWASP Top 10 pro roky 2004, 2007 a 2010

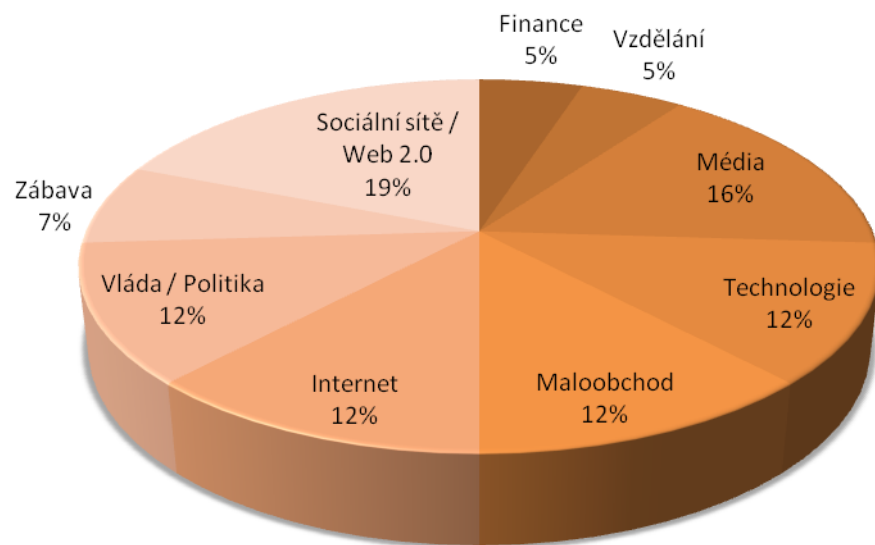
	Top 10 - 2004	Top 10 - 2007	Top 10 - 2010
1	Injection	Cross Site Scripting	Unvalidated Input
2	Cross Site Scripting	Injection Flaws	Broken Access Control
3	Broken Authentication and Session Management	Malicious File Execution	Broken Authentication and Session Management
4	Insecure Direct Object References	Insecure Direct Object References	Cross Site Scripting
5	Cross Site Request Forgery	Cross Site Request Forgery	Buffer Overflow
6	Security Misconfiguration	Information Leakage and Improper Error Handling	Injection Flaws
7	Insecure Cryptographic Storage	Broken Authentication and Session Management	Improper Error Handling
8	Failure to Restrict URL Access	Insecure Cryptographic Storage	Insecure Storage
9	Insufficient Transport Layer Protection	Insecure Communications	Application Denial of Service
10	Unvalidated Redirects and Forwards	Failure to Restrict URL Access	Insecure Configuration Managemen

Zběžným porovnáním lze pozorovat, že se v průběhu let a vývoje programovacích jazyků rizika mění. Jestliže porovnáme roky 2010 s 2007, zjistíme, že se zvyšuje riziko Broken Authentication and Session Management stejně jako Failure to Restrict URL Access, pozice si vyměnily Injection a XSS. V podstatě se ale v Top 10 vyskytují až na dvě položky stejná rizika, avšak na různých příčkách. Při porovnání roku 2007 s rokem 2004 najdeme rozdílů více. Jednak je to způsobeno vývojem SW a HW a stejně tak i rozhodnutím OWASPU zaměřit se v žebříčku více na rizika než na slabiny webových aplikací.

Ze statistiky, vypracované společností WhiteHat Security, z roku 2011 – WhiteHat Website Security Statistic Report, se můžeme dozvědět, která odvětví web aplikací jsou více či méně náchylná k útokům, a jejich procentuální zastoupení. Uvádí také průměrné časy oprav bezpečnostních chyb v jednotlivých odvětvích a další podrobnější data. V grafu uvedeném níže, kde jsou zahrnuta všechna odvětví, se dozvídáme, že nejčastějšími chybami ve webových aplikacích je náchylnost na XSS a únik interních informací. V grafu po něm následujícím jsou zobrazena jednotlivá odvětví a procentuální náchylnost na bezpečnostní rizika. [32]



Graf 1 Top 10 zranitelností webových aplikací z roku 2010 (všechna odvětví) [32]



Graf 2 Přehled náchylnosti na zranitelnost webových aplikací podle odvětví z roku 2009 [32]

1.1 Injection

Aplikace je napadena přes neošetřený vstup, kde potenciální útočník je každý, kdo ovládá syntaxi. Nejběžnějším útokem je SQL Injection, dále pak LDAP, XPath a OS Injection. Je to bezpečnostní chyba, při které je možno manipulovat s daty bez legitimního přístupu.

Při injection spočívá princip ve vkládání nových či rozšiřujících částí dotazu do již existujících dotazů tak, aby překladač tuto část nového kódu interpretoval jako správný příkaz, popř. oznámil chybu, kterou lze zneužít k dalším útokům. Zneužitím této bezpečnostní chyby je získání citlivých dat z databáze jako např. přihlašovací údaje, osobní údaje, čísla bankovních účtů, telefonní čísla apod. Data lze poškodit, měnit či mazat. V některých případech útočník získá schopnost ovládat celou databázi.

Injection není problém jen webových aplikací, ale v podstatě každé aplikace, která s databází pracuje.

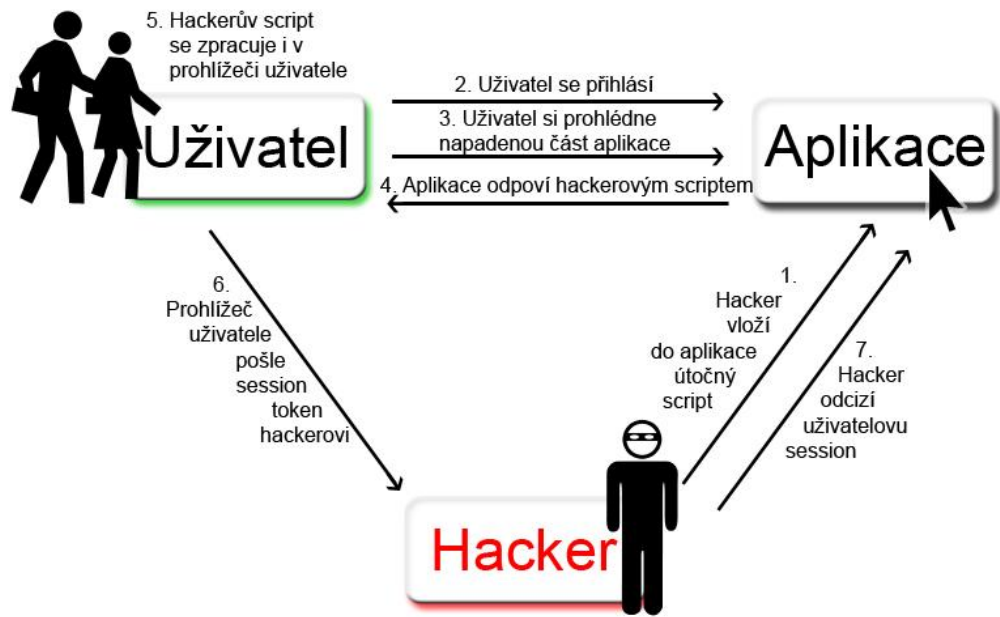
1.2 Cross–Site Scripting

K útoku opět dochází přes nedostatečně kontrolovaný vstup. Princip, podobně jako u injection, spočívá v podstrčeném skriptu, který útočník odešle. Tato část scriptu je pak při nedostatečné kontrole a validaci prohlížečem interpretována. Dopadem takového útoku pak může být poškození layoutu, disfunkce či obcházení zabezpečení webových stránek nebo získání osobních údajů. Často je s XSS také spojován se sociálním inženýrstvím. Tato kombinace, kdy útočník donutí uživatele spolupracovat, bývá mnohonásobně úspěšnější.

Cross–Site Scripting lze rozdělit na tři základní typy.

1.2.1 Stored (Persistent)

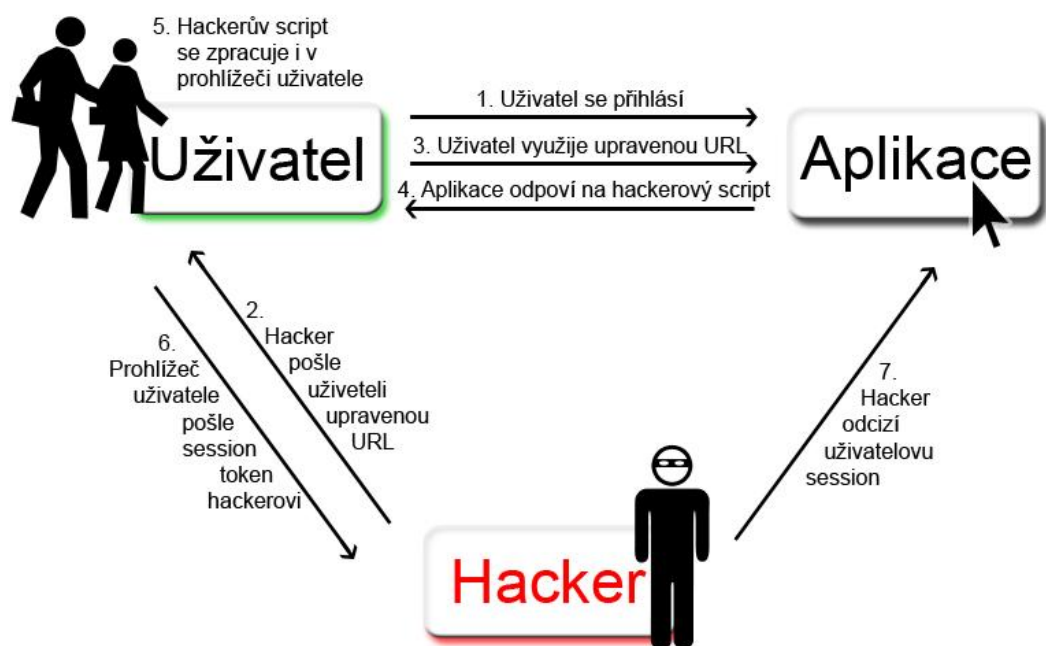
Tento typ XSS vzniká ve webových aplikacích, ve kterých je zahrnut dynamický obsah. Útok typicky zahrnuje nejméně dva požadavky na aplikaci. V prvním kroku je útočník schopen odeslat script, například přes formulář typu textarea, do aplikace (databáze s komentáři, fórum apod.) a pokud nejsou tato data dostatečně validována, uloží aplikace tento závadný script do databáze. V druhém kroku je napadena oběť, která shlédne útočnickova data. Každý, kdo pak navštíví napadenou stránku s tímto dynamicky načítaným obsahem z databáze, je ohrožen. Díky těmto krokům je tento útok někdy označován taktéž jako second–order cross–site scripting. Typ Stored je nejvíce nebezpečný ze tří jmenovaných typů útoku XSS. [3]



Obr. 3. Vývoj útoku XSS typu stored

1.2.2 Reflected (Non-persistent)

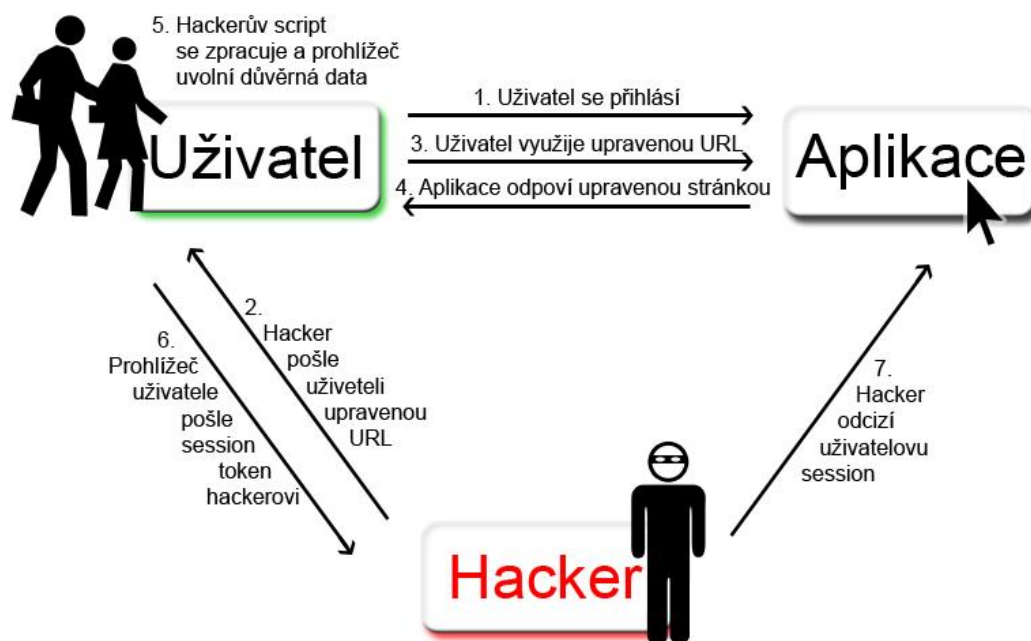
Princip spočívá v úpravě URL adresy. Parametry v ní obsažené se nahradí či doplní o útočný script. Pokud oběť na tento odkaz klikne, script se spustí a útočník může odcizit sessions nebo jiná důvěrná data. Oproti předchozímu typu se script nikde neukládá a je vykonán okamžitě. Typ reflected je nejběžněji používaný útok, vyžaduje však od oběti přístup na napadenou aplikaci přes upravenou URL. Pokud uživatel nepřistupuje na aplikaci přes tuto upravenou URL, není napaden. [3]



Obr. 4. Vývoj útoku XSS typu reflected

1.2.3 DOM based XSS

Označuje se též jako lokální nebo type 0. Způsob napadení je podobný jako u typu reflected. Na rozdíl od něj, kdy byl útok zprostředkován serverem generujícím script do aplikace, využívá DOM metoda přímo prohlížeč napadeného uživatele.



Obr. 5. Vývoj útoku XSS typu DOM based

1.3 Broken Authentication and session management

U rozsáhlých aplikací je session management důležitým stavebním prvkem. Není jednoduché zabezpečit autentizaci a session po celou dobu sezení uživatele, a proto se s touto problematikou musí počítat již v návrzích aplikace. Dopadem po ukradnutí session nebo prolomení logování bývá většinou odcizení účtu. Po prolomení může útočník provádět s účtem vše, co jeho majitel, nebo přinejmenším jen prohlížet citlivé údaje. U lépe zabezpečených aplikací se proto autentizace doplňuje silnějšími metodami jako například kryptografickými tokeny nebo biometrickými prvky.

Útočník opět hledá slabiny v neošetřených logovacích formulářích, ve funkcích se session managementem apod. Toto bezpečnostní riziko session nemá přesnou definici jako je tomu u injection nebo XSS. Název pokrývá řadu způsobů, jak ohrozit aplikaci či uživatele. Může zde být zahrnuto například injection, kdy se hacker oboje logování SQL kódem, XSS nebo

session ID v URL. Výjimkou není ani špatné ošetření timeoutu. Jestliže uživatel sdílí počítač i s jinými osobami a neukončí session odlášením, nastává riziko, že osoba využívající počítač po tomto uživateli vstoupí do neukončené session. Tímto způsobem jednoduše odcizí uživatelský účet. [7]

1.4 Insecure direct object references

Přímý odkaz na objekt vznikne, jestliže vývojář zveřejní odkaz na interně implementovaný objekt, jako je například soubor, adresář nebo klíč k přístupu do databáze. Bez kontroly přístupu k těmto datům nebo jiné ochrany pak může útočník modifikovat tyto odkazy pro získání přístupu k neoprávněným datům. [7]

Útočníkem může být i řádný uživatel systému, který jednoduše změní některý z parametrů, který přímo odkazuje na nějaký objekt, na který by jinak uživatel neměl oprávnění. Při neověření přístupu k cílovým datům, pak aplikace data zobrazí. Logika, že pokud nikomu nesdělíme, že data existují, nebude je nikdo hledat, je nesmyslná.

V dnešní době ani není nutné používat násilné techniky. I díky jednoduchému vyhledávání pomocí vyhledávače Google lze najít obsah, který není veřejně vystavován. Dokonce i takový, který není skrytý za uživatelsky přívětivou doménou.

Pro účelné vyhledávání a Google hacking je potřeba zvládnout syntaxi. Její základ tvoří Booleova logika, dále pak na něj navazují pokročilé operátory ve tvaru operátor:argument. Se znalostí těchto operátorů není problém nalézt výpisy adresářů, interní dokumenty, ale i síťový hardware jako jsou tiskárny, kopírky, webové kamery atd.

Ukázkovým příkladem může být vyhledávání nezabezpečených síťových tiskáren. Mnoho takovýchto HW zařízení dnes disponuje webovým rozhraním. Pomocí operátoru pro hledání řetězce v URL tak můžeme vyhledat zařízení například od značky HP. Vzniklým příkazem `inurl:hp/device/this.LCDispatcher` se pomocí vyhledávače dostaneme k nezabezpečené tiskárně, kde se můžeme dozvědět např. o stavu tiskárny nebo i poslat nějaký dokument na tisk. [12]

Tab. 2. Vybrané pokročilé operátory pro vyhledávání pomocí vyhledávače Google

Operátor	Význam
intitle	vyhledávání v titulu stránky
allintitle	vyhledávání v titulu jednotlivě všechna slova a fráze uvedené jako argumenty
inurl	vyhledávání řetězce v URL
allinurl	Vyhledávání v URL slovo nebo frázi uvedené jako jeho parametry
site	vyhledávání ve specifikované doméně
filetype	Vyhledávání klíčových slov v souborech s příslušnou příponou
link	vyhledávání stránky obsahující odkazy směřující na daný cíl
numrange	vyhledávání informací s čísly v určitém rozsahu
intext	vyhledávání řetězce přímo v textu webové stránky, nikoli v nadpisu ani v URL či souborech

Vyhledávač Google je špičkou ve vyhledávání jakékoliv obsahu na internetu. Usnadňuje práci regulérním uživatelům. Bohužel, lze ho snadno využít i jako pomůcku pro hackery. Pro ně skýtá taktéž malou výhodu v částečné anonymitě.

1.5 Cross – site request forgery

Cross – site request forgery bývá často zaměňován za XSS. Jedná se spíše ale o opačný typ útoku. V případě XSS je obětí uživatel. U CSFR je tomu naopak, tedy uživatel je útočníkem nebo alespoň spolupachatelem. Nejčastějšími cíly CSRF útoků jsou např. nevědomé hlasování v anketách, vkládání příspěvků do diskusních fór, nákupy v e-shopech, změny v administraci apod. Jeden ze základních předpokladů pro úspěšný útok je znalost aplikace. Nejčastější útoky jsou realizovány přes zneužití metody GET a POST.

Účinná obrana proti CSRF útokům je například posílání skrytých polí ve formulářích s vygenerovanou hodnotou a následnou její kontrolou nebo pomocí proměnného URL. Obrana na straně klienta v podstatě neexistuje. Doporučuje se však bezpečněji nastavit prohlížeč a firewall, nepoužívat automatické logování a zvažovat, zda odkaz, který se uživatel chystá navštívit, není závadný. [11]

1.6 Security misconfiguration

Tento typ útoků využívá nedostatků v konfiguraci aplikačních, webových a databázových serverů. Možné příčiny mohou být v ponechání výchozích hodnot, neaktuálnost SW i chyba na straně administrátora. Častým případem je, že provoz aplikace zaštiťuje jiná osoba, než osoba, která aplikaci vyvíjela. Tímto způsobem může dojít k nedorozumění a mohou nastat bezpečnostní chyby, kterých hacker využije. [7]

Nejvíce se vyskytujícími chybami jsou:

- Neopravené bezpečnostní chyby v softwaru serveru
- Diskonfigurace softwaru, které umožňují výpis adresáře nebo útok na directory traversal
- Zbytečné výchozí soubory, zálohy, vzorové soubory obsahující konfigurační skripty, aplikace, kusy kódu a webové stránky
- Nesprávné nastavení oprávnění pro soubory a adresáře
- Zbytečně zapnuté služby, včetně vzdáleného přístupu a správy obsahu
- Výchozí účty, které využívají výchozích hesel
- Administrátorské funkce nebo debugging funkce, které jsou přístupné nebo povolené
- Chybové hlášky, které obsahují příliš mnoho informací
- Špatně nakonfigurované SSL certifikáty a šifrovací funkce
- Použití self-signed certifikátu pro ověřování a man-in-the-middle ochranu
- Použití výchozích certifikátů
- Nesprávné ověřování externími systémy [7]

Mezi potenciální útočníky patří jak anonymní vnější návštěvníci aplikace, tak uživatelé s účty. Je nutností, aby všichni uživatelé byli kontrolováni a měli svá omezená práva. Jestli je aplikace náchylná na špatnou konfiguraci, je možné zjistit pomocí manuálního testování i pomocí různých scannů, které prověřují známá a častá slabá místa. Tyto nástroje by se měly používat interně i externě a hlavně pravidelně.

1.7 Insecure cryptographic storage

Ochrana citlivých dat je dnes již součástí většiny webových aplikací. Ne vždy je však dostatečná, ať už použitím nedostatečných šifrovacích algoritmů nebo dokonce neošetřením citlivých dat. Pro ochranu by se měly používat silné šifrovací funkce jako například AES, RSA, SHA-256 nebo lepší. Mezi slabé algoritmy se v dnešní době počítá i oblíbené MD5 nebo SHA-1.[13] Pokud opravdu vývojář aplikace chce tyto šifry použít, doporučuje se alespoň je „osolit“, což znamená přidat náhodný řetězec k původnímu heslu a tento nově vzniklý řetězec zašifrovat. Sůl se taktéž ukládá do databáze k přihlašovacímu jménu a zahashovanému heslu. Krok „osolení“ pak útočníkovi, který odcizil seznam přihlašovacích jmen a hashí, znemožní využít „rainbow table“. [1] Ani používání vlastních šifer není bezpečné vzhledem k náchylnosti chybě lidského faktoru.

Snaha uchránit data koresponduje i s ostatními riziky jako například prolomení ověřování identity, zcizení session, injection apod. Citlivá data uživatelů, čísla kreditních karet, hesla apod. by měla být dostatečně zabezpečena i po prolomení ostatních bezpečnostních zábran, jelikož se v podstatě jedná o poslední obranný mechanismus. Již při registraci, by měl být uživatel nucen, aby využil dostatečný počet znaků a v ideálním případě i různé jejich kombinace. Pokud se nevyužívá jednosměrných hashí, ale šifrování jako např. AES, je nutné dobře zabezpečit klíč.

1.8 Failure to restrict url access

Failure to restrict url access je častým problémem ve web aplikacích, pokud stránka nemá korektní kontrolu přístupu. Obsah je pak zobrazen i neoprávněným uživatelům, pro které by měl zůstat skryt. Typickým příkladem je zobrazení odkazu na určitý obsah pro zalogované uživatele, který se nezalogovaným uživatelům neukazuje. Ovšem při znalosti URL se obsah zobrazí. Stránka jako taková nemá dostatečně kontrolované přístupy, pouze „zviditelnění“ odkazu na tuto stránku se ověřuje. Pro vyhnutí se této chybě je nutné mít u každé stránky kontrolu přístupu.

Nejčastější metoda, jak jsou tyto bezpečnostní chyby odhaleny, je tzv. „forced browsing“. Cílem této metody je najít a přistoupit hrubou silou ke zdrojům, na které aplikace neodkazuje, ale jsou přístupné. Útok se většinou provádí manuálně, pokud je předpoklad, že stránky či indexy jsou založené na generování nebo předvídatelných hodnotách.

1.9 Insufficient Transport Layer Protection

Komunikaci, která je spojena především mezi prohlížeč – web server, web server – aplikace, aplikace – databázový server, apod. by měl zaštitovat bezpečnostní protokol (HTTPS - SSL/TLS). Zaručuje to, že transportní vrstva je chráněna, šifrována a že nebylo s daty manipulováno při přepravě. Útočníkem může být kdokoliv, kdo dokáže sledovat síťový provoz. Takto zabezpečené by měly být všechny privátní stránky a aplikace, logovací skripty, atd.

Častými chybami jsou:

- Zabezpečený je pouze proces autorizace, následující stránky již ne
- Nepoužití secure cookies
- Nepoužití silných algoritmů
- Prošlé certifikáty
- Mix zabezpečeného a nezabezpečeného obsahu (TLS vs Non-TLS)
- Přesměrování z HTTP na HTTPS

Tyto chyby mohou vést jako většině případů k odcizení uživatelského účtu nebo dokonce administrátorského účtu, a tím ke ztrátě kontroly nad aplikací. Špatné či nedostatečné SSL nastavení pak usnadňují phishingové útoky a útoky man-in-the-middle. [7]

1.10 Unvalidated redirects and forwards

Útoky přesměrováním na jiné stránky jsou velmi podobné XSS, avšak daleko jednodušší. Pokud se na webových stránkách objevuje odkaz přesměrování, je velmi snadné odkaz přepsat. A jestliže vývojář nezahrne do kontroly whitelisty url adres, na které je možno přesměrovávat, může se pak uživatel, který se k takovému odkazu dostane, např. na sociálních sítích, a klikne na něj, stát obětí. Odkaz, který při zběžném prohlédnutí, neodkazuje na výslednou stránku, se zdá být v pořádku. V nejčastějších případech se po přesměrování může jednat o navazující phishingový útok se vzhledově identickou stránkou a uživatel, který nebude kontrolovat URL adresu, se může, v dobré vůli, zalogovat. Útočník pak získává přihlašovací údaje. Výjimkou není také přesměrování na stáhnutí škodlivého malwaru. Tento typ útoku je nejúčinnější v kombinaci se sociálním inženýrstvím,

phisingovými útoky, malware a jinými podvodnými činnostmi. Jako obrana proti takovýmto zneužitím je kontrola přes whitelisky nebo se redirect a forward úplně vyvarovat. [14]

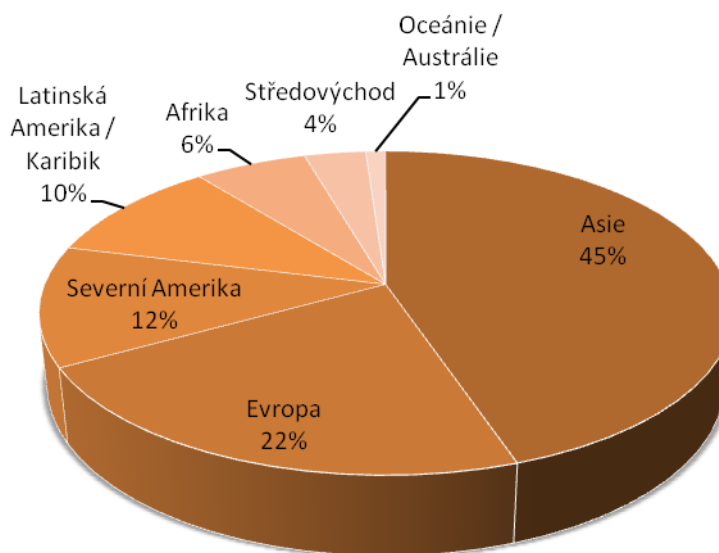
1.11 Shrnutí kapitoly

Již při návrhu aplikace by měla být snaha minimalizovat její zranitelnost. Propojením několika programovacích jazyků se potenciálně zvyšuje riziko vzniku bezpečnostní chyby. Dopady pak mohou být od odcizení osobních dat uživatelů a jejich zneužití či prodej, přes finanční dopady na firmu provozující webovou aplikaci, poškození dobrého jména, až po technické dopady - zničení celé aplikace, odcizení aplikace atd. Seznam nejčastějších potenciálních rizik se všeobecně považuje za minimum toho, co by měl vývojář otestovat na vlastní aplikaci. Existují však desítky dalších neméně závažných rizik, které mohou být zneužity.

2 ANALÝZA DOSTUPNÝCH TESTOVACÍCH NÁSTROJŮ

V roce 2011 mělo v České Republice k internetu přístup 7,22 milionu uživatelů (tj. 70,9%). [21] Podle Českého Statistického Úřadu to od roku 2006 znamená nárůst o 130% (1,4milionu) více připojených domácností. [22] V celosvětovém měřítku bylo k 31. 12. 2011 připojeno 2,26 miliardy lidí. [20] Pokud to porovnáme se stejným datem o 12 let zpět, tj s rokem 2010, zjistíme, že nárůst byl o 1,9 miliardy uživatelů. Vzhledem k rychlému rozvoji internetu, jeho webových aplikací a obrovskému nárůstu uživatelů, je zabezpečení webových aplikací velmi diskutované téma. Se zvyšujícími se počty uživatelů, se také zvyšuje riziko zneužití internetu a hackerských útoků. Jsou proto vyvíjeny testovací nástroje, které pomáhají programátorům webových aplikací odhalovat bezpečnostní díry. Rozdělovat je můžeme do několika kategorií, jako např. scannery, spidery, sniffery, fuzzery, port scannery atd. Většina nástrojů ale je multifunkční a testování pomocí výše zmíněného zahrnují.

Testování webové aplikace je důležitým krokem k udržení zabezpečení a ochraně dat. Na druhou stranu lze tyto nástroje snadno zneužít k zjištění slabín a následnému napadení webové aplikace hackerem.



Graf 3 Uživatelé internetu rozdělení podle světových regionů [20]

2.1 OWASP ZAP - Zed Attack Proxy

Penetrační testovací nástroj pro hledání zranitelností webové aplikace je vyvíjen v rámci OWASP projektu v čele se Simonem Bennettsem. Jeho první uvolnění pro používání bylo v září v roce 2010. Designován je pro vývojáře se zkušenostmi v oblasti webové bezpečnosti, stejně jako testery. Open Source nástroj podporuje jak automatizované testy, tak i nástroje pro manuální hledání bezpečnostních rizik. Přeložen byl do devíti světových jazyků a v roce 2011 získal cenu Toolsmith Tool of the Year. [7]



Obr. 6. Logo OWASP ZAP [7]

Automatizované nástroje:

- Spider – Nástroj na prohledávání odpovědí při procházení aplikace a upozorňuje také na nenavštívené odkazy. Doporučuje se až po manuálním zmapování aplikace.
- Pasivní scanner – Běží na pozadí, a proto nezatěžuje tolik aplikaci. Pasivně prohledává odpovědi testované webové aplikace a odpovědi nijak nemodifikuje, je proto bezpečnější pro použití. [27]
- Aktivní scanner – Nástroj na detekci potenciálních bezpečnostních chyb a případných známých útoků. Aktivní skenování aplikace se považuje za útok, měly by být používány pouze majitelem aplikace nebo s jeho svolením. [27]
- Brute – force scanner – Nástroj, který hrubou silou přistupuje k adresářům a souborům. Zkouší vyhledávat názvy těchto objektů. Pro tyto účely obsahuje rozsáhlou databázi častých a obvyklých názvů.
- Port scanner – Nástroj skenuje porty a následně zjistí, které jsou otevřeny.

Manuální nástroje:

- Zachycení proxy – Umožní vidět veškeré požadavky na webovou aplikaci a všechny odpovědi, které na požadavky odpovídají.
- Manual request editor – Editor na vytváření vlastních požadavků při testování webové aplikace.
- Fuzzer – Nástroj na testování parametrů aplikace. Typicky testuje přetečení zásobníku, chybové hlášení, zranitelnost formátování řetězců. Mezi pokročilejší funkce patří kontrola proti SQL Injection, XSS a útoky na Directory Traversal. [26]

2.2 Selenium IDE

Tento testovací nástroj, který lze získat jako Open Source doplněk pro prohlížeč Firefox, je napsán v jazyce JavaScript. Řadí se po bok dalších testovacích nástrojů od firmy SeleniumHQ. Vývoj těchto nástrojů začal v roce 2004, hlavním programátorem byl Jason Huggins.

Selenium IDE byl vytvořen vývojářem Shinya Kasatani v Japonsku. Ten jej implementoval jako testovací framework doplněk do prohlížeče. Dříve byl znám pod názvem Selenium Recorder. První verze byla vydána v roce 2006 a vývoj pokračuje do dnešních dní, kdy čítá velkou základnu i neoriginálních doplňků a rozšíření programujícími nadšenci. V roce 2007 zahájili i spolupráci s Googlem. Ve frameworku lze testovat aplikace psané v jazycích PHP, C#, Perl, Python, Ruby, Java běžících na MS Windows, Linuxu i Macintoshi. Lze v něm editovat testy a výstupy se ukládají ve formátu XHTML. Program simuluje běžné aktivity uživatele jako je klikání, psaní znaků, apod. Je užitečný při kontrole po dokončení aplikace, stejně tak i při změně hostingu či jiných úpravách aplikace. Spolu s ostatními produkty Selenium CORE, Selenium RC a Selenium GRID se jedná o oblíbené a často využívané nástroje pro kontrolu webové aplikace. [16]



Obr. 7. Logo Selenium IDE [16]

2.3 Skipfish

První verze tohoto Open Source testovacího nástroje, pocházejícího od společnosti Google (hlavní programátoři: Michal Zalewski, Niels Heinen, Sebastian Roschke), byla zveřejněna v roce 2010. Aktuální verze programu Skipfish je 2.05 beta a vývoj dále pokračuje. Je psaný čistě v jazyce C a tímto si zajišťuje vysokou rychlost. Po stažení se dostáváme k

nezkompilovanému kódu. Využití tohoto nástroje tudíž nemá user-friendly prostředí, a proto je využíván hlavně vývojáři ke kontrole své webové aplikace. Scanner komplexně kontroluje náchylnost na závažná bezpečnostní rizika, jako jsou Injection útoky, XSS, zcizení cookies, nastavení SSL obsahu, apod., stejně tak jako méně závažné problémy. [17]



Obr. 8. Logo Skipfish [17]

2.4 Nikto2

Testovací nástroj na automatické scanování slabých míst web severu je distribuován pod licencí Open Source. Byl napsán Chrisem Sullo a Davidem Lodgem. První verze Nikto 1.0 Beta bylo uvedeno na konci roku 2001. Během dvou let se stalo jedním z nejoblíbenějších scannerů. Verze 2.0 vyšla v roce 2007. Nikto je psané v jazyce Perl a postavené je na základě LibWhisker2. Spustitelný je na jakémkoliv systému, který podporuje instalaci Perlu. Mezi vybrané funkce patří podpora SSL a HTTP proxy, kontrola zastaralých serverových komponent, sken portů, hádání subdomén, testování nejčastějších zranitelností webu, sken adresáru atd. Uživatelským rozhraním je zde příkazová řádka.

Existují i obdobné projekty, které využívají databázi Nikta nebo s Niktem spolupracují. Mezi těmito projekty lze zmínit nástroj Wikto pocházející od Sensepost, který je určen primárně pro uživatele MS Windows. Wikto je psané v C#. Spolupracuje navíc i s Googlem. Dalším alternativním projektem pro uživatele Apple Macintosh je MacNikto. Vyvíjen je v AppleScript, Xcode a InterfaceBuilder. Oba zmiňované nástroje zahrnují grafické uživatelské prostředí. [15]



Obr. 9. Logo Nikto2 [15]

2.5 Websecurify - Website Security Testing Tool

Multiplatformní nástroj běží na všech třech nejdůležitějších OS – MS Windows, Linux a Mac OS. Jeho velké plus je v tom, že podporuje i testování na mobilní platformě iOS, v budoucnu i Android.



Obr. 10. Logo Websecurify [25]

Websecurify najdeme ve čtyřech verzích:

- Websecurify Browser Extensions (neplacená rozšíření pro Mozilla Firefox a Google Chrome)
- Websecurify Basic (neplacená verze se základními nástroji a funkcemi)

- Websecurify Mobile (placená verze pro testování na mobilní platformě)
- Websecurify Advanced (placená plná verze)

Pracovní prostředí je přívětivé pro většinu uživatelů a práce s nástrojem je jednoduchá, rychlá. Projekt zahrnuje mimo jiné testování bezpečnostních rizik OWASP Top 10. Výhodou oproti většině tomuto podobných nástrojů je i souběžné testování více aplikací. Vývoj byl započat v polovině roku 2008. První vydaná verze 0.2 byla vyvíjena pro Mac a testovala pouze XSS a SQL Injection. V následující verzi bylo přidáno i testování CSRF. Čtyřletým vývojem dostala aplikace dnešní podobu a testuje drtivou většinu závažných i méně závažných rizik. [25]

2.6 Další testovací nástroje

Existuje celá řada komerčních i nekomerčních nástrojů. Následující tři doplňují výše uvedené nástroje a zaměřují se spíše na síťový provoz.

2.6.1 Nmap

Nmap, taktéž Network Mapper, je port scanner určený ke správě sítí, scanningu otevřených portů a zjišťování služeb, OS či typ použitého firewallu a jeho pravidel. Program vyvinul Gordon Lyon a je distribuován pod licencí GPL. První verze byla vydána v roce 1997, avšak jeho vývoj pokračuje do dnešních dní. Nástroj pracuje na většině OS. Jeho user interface je příkazová řádka, je však možné od verze 2.50 využít grafické rozhraní NmapFE, které bylo později od verze 4.50 nahrazeno ZenMapem. [18]

2.6.2 WireShark

Nástroj WireShark je protokolový analyzátor a paketový sniffer. Vznikl z původního Ethereal, který byl vyvíjen od roku 1998. Je distribuován jako multiplatformní SW pod licencí GNU General Public License Nabízí jak grafické rozhraní, tak i terminálovou verzi pod názvem TShark. [19] Vhodné využití je například při problematice s CSRF.

2.6.3 Backtrack 5

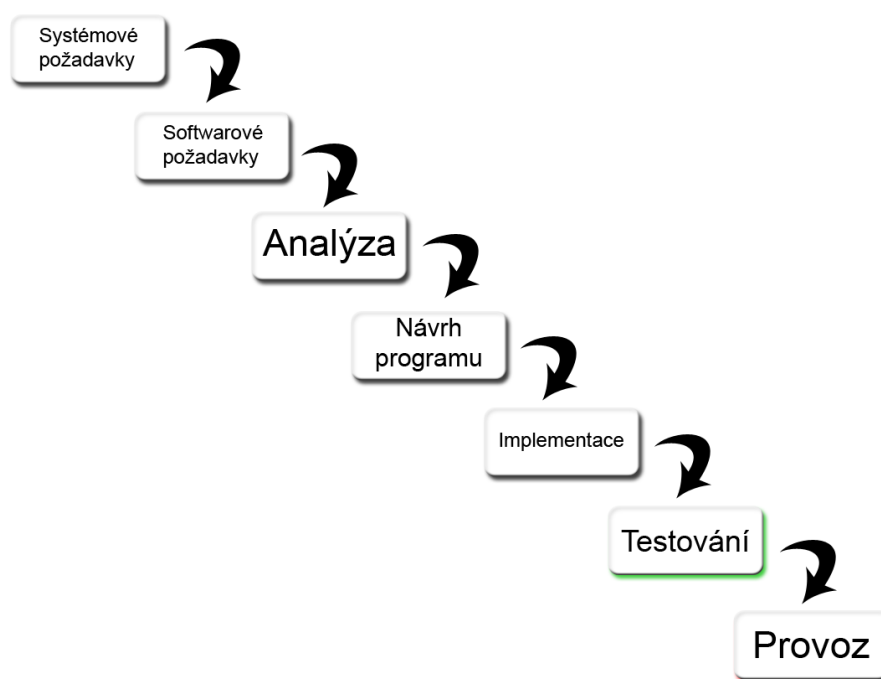
Backtrack5 je Linuxová Live CD distribuce založená na Ubuntu Linux, kterou vyvíjí skupina Offensive Security. Tato pentest distribuce obsahuje velké množství aplikací a

nástrojů pro průnikové testování sítí, OS a SW a HW. Jeho poslední verze vyšla v květnu 2012, je avšak průběžně neustále aktualizován.

3 MOŽNOSTI AUTOMATIZOVANÉHO A MANUÁLNÍHO TESTOVÁNÍ

Testování bezpečnosti webové aplikace je jedna z částí celkového testování webu před uvedením do provozu na internet. Vedle tohoto testování se řadí taktéž funkční testování, testování použitelnosti, kompatibility a výkonu. Vše se optimalizuje tak, aby aplikace běžela stabilně a bez chyb v různých prohlížečích, na různých OS a mobilních platformách.

Podle nejstaršího modelu životního cyklu SW, který se nazývá Vodopádový, se testování provádí jako poslední krok před uvedením do provozu. Poprvé byl definován v roce 1970 Winstonem W. Roycem. Myšlenka, že k další fázi vývoje se smí přistoupit až po úplném dokončení fáze aktuální a nemožnosti se vrátit do jakékoliv fáze vývoje, je v dnešní době vývoje nepraktikovatelná. Nevýhodou modelu je taktéž testování až po úplném vývoji aplikace. V praxi se tento model tedy příliš nevyužívá, lze je však využít u malých projektů. Vychází z něj však mnoho různých modelů, které se snaží odstranit nedostatky tohoto, a taktéž bývá často využíván k výukovým účelům. [23]



Obr. 11. Vodopádový model vývoje SW

Mnohem propracovanější model vývoje je například Spirálový model. Ten byl představen v roce 1986 a pokrývá většinu nedostatků modelu vodopádového. Cyklus je rozdělen do 4 hlavních fází. Testování zde probíhá po každé fázi vývoje, je proto vhodné použití

dostupných automatizovaných testů. Dnes bývá k vývoji aplikací většinou zvolen moderní přístup jako je agilní programování, kde se taktéž využívá automatizované testování. Doporučuje se klást důraz na testování bezpečnosti již během vývoje. Řešení chyb již v obsáhlých programech je jednak složitější, ale i finančně více nákladné.

Z pohledu hackerů jsou útoky na webovou aplikaci podobné jako techniky pro její testování. Základem je zmapování aplikace většinou pomocí nástrojů na automatizované testování za účelem zjištění slabín, jež jsou následně využity pro cílený útok. Ten může být prováděn již manuálně.

3.1 Manuální testování

Manuální testování bezpečnosti webové aplikace je prováděno člověkem, testerem, který by měl prokazovat určitou dávku kreativity. Výhodou je, že není zapotřebí žádných podpůrných programů, není náročné na přípravu, jako je tomu u automatizovaného testování. Nevýhodou je časová náročnost. Manuální testování se provádí v aplikacích, kde nelze použít automatizované testy, nebo tam, kde je zapotřebí větší flexibility. I přes obsáhlé databáze u automatizovaných nástrojů je někdy nutnost odzkoušet manuálně. Zkušenosti a vědomosti testera zde hrají velkou roli k úspěšně otestované aplikaci.

3.2 Automatizované testování

K automatizovanému testování webové aplikace se přistupuje tehdy, je-li potřeba u testů několikanásobné přesné opakování nebo preferujeme-li rychlost. Časová úspora je oproti klasickému manuálnímu testování asi největší výhodou automatizovaných testů. Nástroje testují většinu známých bezpečnostních chyb, je u nich však podstatná neustálá aktuálnost. Je jim také nutné vyhradit HW prostředky. Tyto nástroje jsou pouze urychlující pomůckou testerů. Po vyhodnocení výsledků automatizovaného nástroje je ale i nadále nezbytná přítomnost lidského faktoru, který vyhodnotí, jak jsou rizika závažná, nebo odzkouší chyby i manuálně.

3.3 Hodnocení rizik

Hodnocení potencionálních rizik je důležitá součást následného testování. Umožňuje mít celkovější nadhled na problematiku a pomáhá vyhodnotit, které bezpečnostní chyby je nezbytné opravit. V následujícím textu jsou popsány dvě používané techniky.

3.3.1 DREAD

DREAD je systém hodnocení bezpečnostních rizik využívaný firmou Microsoft. Jeho název je odvozen z počátečních písmen kategorií. Hodnotí se 5 kategorií, ze kterých vznikl i název systému.

- Damage (poškození): Jak velká škoda vznikne?
- Reproducibility (reprodukovatelnost): Jak snadné je napodobit útok?
- Exploitability (zneužitelnosti): Jak snadný je útok?
- Affected users (ovlivnění uživatelé): Na kolik uživatelů bude mít útok dopad?
- Discoverability (objevitelnost): Jak snadné je chybu objevit?

Každá kategorie se hodnotí známkou 0, 1, 2 nebo 3, kde znamená 0 - žádné riziko, 1 – nízké riziko, 2 – střední riziko a 3 – vysoké riziko. Pro výsledné hodnocení se vyhodnotí součet ze všech kategorií. [24]

3.3.2 OWASP Risk Rating Methodology

Systém hodnocení od OWASPU je podrobnější než předchozí DREAD. Vychází se z modelu: $RIZIKO = PRAVDĚPODOBNOST \times DOPAD$. Postupuje se v 5 krocích:

3.3.2.1 Identifikace rizika

Zde patří identifikace rizika, které má být hodnoceno. Získávají se informace o zranitelnosti, útoku, útočníkovi a dopadech z testů.

3.3.2.2 Faktory pro odhad pravděpodobnosti

Po identifikaci bezpečnostního rizika se zjišťuje pravděpodobnost zneužití. Hodnotí se na stupnici od 0 po 9, kde 0 představuje žádnou závažnost a 9 závažnost maximální. Hodnocení probíhá ve dvou kategoriích, kde se hodnotí jednotlivé kroky:

Odhad pravděpodobnosti útoků ze strany útočníka:

- Úroveň dovedností
- Motiv
- Příležitost
- Velikost

Odhad pravděpodobnosti ze strany zranitelnosti:

- Jak snadné je zjištění zranitelnosti
- Jak snadné je využití zranitelnosti
- Povědomost
- Detekce průniku

Jednotlivé hodnocení sečteme a vydělíme počtem kroků (8), tím získáme průměrnou hodnotu.

3.3.2.3 Faktory pro odhad dopadu

Po předchozích dvou krocích stanovujeme, jakou technickou a bussines škodu by zanechaly. Opět hodnotíme jako v předchozím kroku (0 – 9).

Technické faktory dopadu:

- Ztráta důvěrných dat
- Ztráta integrity
- Ztráta dostupnosti služeb
- Ztráta odpovědnosti

Bussines faktory dopadu:

- Finanční škody
- Poškození dobrého jména
- Poškození obchodních dohod
- Narušení ochrany soukromí

Stejně jako v předchozím kroku spočítáme sumu a zprůměrujeme.

3.3.2.4 Určení závažnosti rizika

Každý z průměrů dosadíme do tabulky a získáme tím úroveň faktorů a úroveň dopadů.

Tab. 3. Převod průměrů
na úroveň

Průměr	Úroveň
0 až <3	nízká
3 až <6	střední
6 až 9	vysoká

Obě hodnoty dosadíme do následující tabulky podle modelu

$$\text{RIZIKO} = \text{PRAVDĚPODOBNOST} \times \text{DOPAD}$$

Tab. 4. Celkové vyhodnocení závažnosti

		Celkové riziko			
Dopad	vysoký	střední	vysoké	kritické	
	střední	nízké	střední	vysoké	
	nízký	velmi nízké	nízké	střední	
		nízká	střední	vysoká	
		Pravděpodobnost			

Z této tabulky již získáváme jasnou představu, jak závažná jsou rizika.

3.3.2.5 Rozhodování o tom, co opravit

Po dosažení výsledků následuje rozhodování, které chyby je nutno neprodleně opravit. Obecně se postupuje od nejzávažnějších chyb po méně závažné. Občas je však nutné rozhodnout se, jestli se na opravu méně závažných rizik vyplatí investovat finanční prostředky. [7]

II. PRAKTICKÁ ČÁST

4 ÚVOD DO PRAKTICKÉ ČÁSTI

Hlavním cílem praktické části této bakalářské práce bylo naprogramovat webovou aplikaci. Tato aplikace, jež je zaměřena na demonstraci nejčastěji se vyskytujících bezpečnostních rizik. K naprogramování byly použity Open Source technologie PHP, MySQL a značkový jazyk HTML. Aplikace obecně popisuje vybraná rizika. Ke každému popisu jsou připojeny dva příklady. Na prvním z nich je možné vyzkoušet, jak takové útoky probíhají. Nemají víceméně žádnou ochranu a popisují, jak by kód neměl vypadat. Druhý příklad je doplněn o náležité ochranné prvky a ukazuje, jak se účinně proti riziku bránit. V aplikaci lze vyzkoušet SQL injection, XSS, CSRF, nezabezpečené objekty, nekontrolované přesměrování a neomezení přístupu. Tyto rizika byla vybrána ze seznamu OWASP TOP 10. [7]

5 VYUŽITÉ PROGRAMOVACÍ JAZYKY A TECHNOLOGIE

Pro naprogramování aplikace bylo využito jazyků a technologií, které se počítají za základ většiny webových stránek dnešní doby. V následujícím textu jsou jednotlivě stručně popsány.

5.1 HyperText Markup Language

HTML je značkovací programovací jazyk, určený pro tvorbu webových aplikací a webových prezentací v prostředí World Wide Web. Jeho vývoj je zpětně ovlivněn vývojem internetových prohlížečů. Vychází z univerzálního značkovacího jazyka SGML. Vyznačuje se množinou tagů, jejich atributů a předepsanou strukturou, do které spadá deklarace definice typu dokumentu, kořenový element, hlavička a tělo dokumentu.

5.1.1 Historie

Vývoj jazyka započal v roce 1989 v CERNU. Tuto první neoficiální verzi vyvinuli Tim Berners-Lee spolu s Robertem Caillau. Tu následně propojili s protokolem HTTP. Spolu s tímto vznikl i první webový prohlížeč zvaný WorldWideWeb. V roce 1991 byla zprovozněna první webová stránka CERN. Ta byla napsána ve verzi 0.9 a nepodporovala uživatelské grafické rozhraní.

Po dvou letech, v roce 1993, byl uveden návrh na verzi HTML 2.0. Taktéž vyšel prohlížeč Mosaic. Ten se stal prvním internetovým prohlížečem podporující grafické rozhraní. Jeho následovník Netscape, byl uveden o rok později. Standard HTML 2.0 byl však vydán až v roce 1995. Nově obsahoval podporu grafiky a formulářů.

Verze HTML 3.0 nebyla nikdy pro její složitost přijata jako standard. Následující HTML 3.2 bylo obohaceno o tabulky, stylové elementy a zarovnávání textu. Standard byl vydán mezinárodním společenstvím W3C v lednu roku 1997.

Na konci stejného roku byl vydán nový standard HTML 4.0. V této verzi krom jiného byly standardizovány rámy. Nově má být vzhled ovlivňován připojovanými styly.

Tato verze je následována HTML 4.01. Předpokládalo se, že bude jednat již o poslední verzi, která bude nahrazena XHTML. Vydána byla v prosinci roku 1999.

V roce 2007 započal vývoj verze HTML 5. Ukončení vývoje specifikace této verze se odhaduje na rok 2022. [28]

5.2 PHP

PHP je server-side scriptovací jazyk, který byl vyvinut Rasmusem Lerdorfem v roce 1995. Jedná se o Open Source technologii. Je multiplatformní, a proto ji lze provozovat v dnešní době na většině webových serverů a existujících OS. Využívá se pro tvorbu dynamických stránek, jako jsou například e-shopy, diskusní fóra, redakční systémy, informační systémy a dynamické prezentace. Syntaxe tohoto jazyka vychází z jazyků Perl, C a Java, je tedy přívětivá většině vývojářů. Aktuální verze je PHP 5.2.8 a je stále ve vývoji.



Obr. 12. Logo PHP [29]

5.2.1 Historie

První původní verze nesla název PHP/FI. Ta byla vydána v roce 1995. Obsahovala jednoduchou sadu Perl scriptů, pojmenovanou Personal Home Page Tool a implementaci v jazyce C – Form Interpreter, který uměl komunikovat s databázemi. Následující verze PHP/FI 2.0 byla vydána v roce 1997. Ve své době tuto technologii využívalo přes 50 000 domén.

PHP 3 bylo oficiálně vydáno po devítiměsíčním testování v červnu roku 1998. K vývoji projektu se přidali Andi Gutmans a Zeev Suraski, kteří jej kompletně přepsali a přidali obrovské množství rozšíření. Původní název Personal Home Page Tools se přejmenovává na PHP: Hypertext Preprocessor. PHP 3 bylo využíváno přibližně na 10% všech WWW serverů.

V následující verzi PHP 4 byl nejvýraznější počín přepracování engine jádra PHP. Jeho inovací se zvýšil výkon a zlepšila modularita kódové báze PHP. Čtvrtá verze PHP byla vydána v roce 1999. Přidala se podpora HTTP sessions, bufferingů vstupu, bezpečnější

zpracování vstupů a další. Tato verze PHP je provozována přibližně na dvaceti procentech domén, což čítá několik miliónů serverů.

Verze PHP 5 byla vydána v roce 2004. PHP se zde začalo přibližovat jazykům podporujícím objektově orientované programování. Obsahuje taktéž druhou verzi svého jádra Zend Engine 2.0. V podpoře je zahrnuta 32bitová a 64bitová verze. [29]

5.3 Kaskádové styly

CSS je jazyk využívaný pro nadefinování vzhledu webových stránek. Používaný je především ve spojení s HTML a XHTML, může být ale využit i ve spojení s XML dokumenty. Vývoj byl navrhnut a je podporován organizací W3C.

5.3.1 Historie

První verze CCS 1 vyšla v roce 1996. Plnou podporou CSS 1 měl až v roce 2000 prohlížeč Opera. Druhá verze CSS 2 byla uvolněna v květnu 1998. Její využití se ale nedoporučuje. Místo této verze se doporučuje CSS 2.1, která byla standardizována až v roce 2011. Momentálně se může využívat verze CSS 3. Ve vývoji je i CSS 4, které zatím není podporované v žádném z prohlížečů. [28]

5.4 MySQL

Tento databázový nástroj byl vytvořen firmou MySQL AB patřící pod společnost Oracle Corporation. Je to světově nejoblíbenější Open Source databáze, kterou využívají velké organizace jako je například Facebook, Google nebo Adobe. Komunikace s databází je zajištěna pomocí jazyku SQL. Velká výhoda MySQL je vývoj a optimalizace zaměřená na vysokou rychlost.



Obr. 13. Logo MySQL [31]

5.5 phpMyAdmin

System psaný v jazyce PHP je určen pro správu MySQL databází. Disponuje webovým rozhraním. Přeložen byl do světových 57 jazyků. Obdobnou, často využívanou, alternativou ke správě je nástroj Adminer.

5.6 Apache HTTP Server

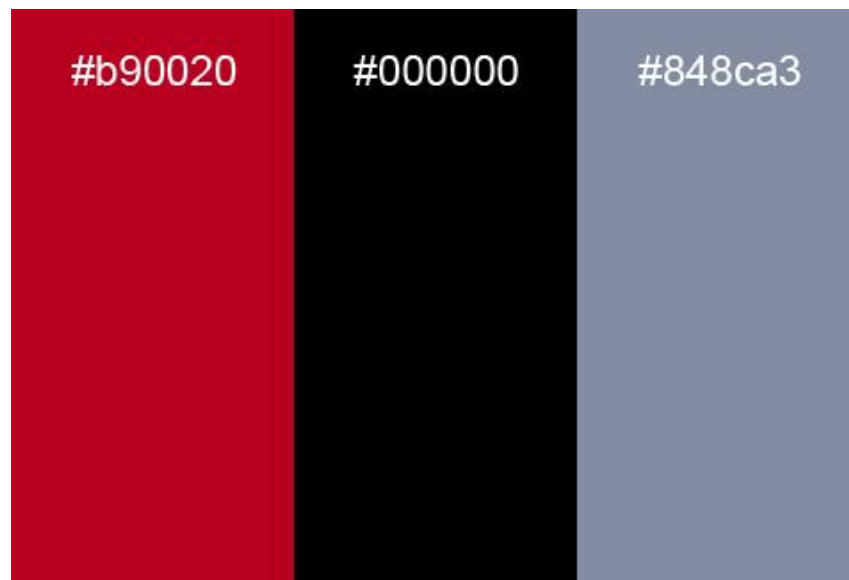
Vývoj tohoto webového SW serveru započal v roce 1993 na Illinoiské univerzitě pod původním názvem NCSA HTTPd. Svůj dnešní název získal od spojení „a patchy server“, kdy správci web serverů přidávali své vlastní úpravy. Dnes používaná verze Apache 2.0 byla kompletně přepsána a je vyvíjena společností Apache Software Foundation. Společně s PHP a MySQL patří k nejčastěji používané triádě programů využívaných pro tvorbu a provozování webových aplikací. Apache je nejpoužívanější webový server na světě. V březnu roku 2012 jej využívalo 65% provozovaných serverů. [30]

6 VLASTNÍ POPIS APLIKACE

Aplikace byla programována tak, aby byla uživatelsky přívětivá a jednoduchá na použití. Na jednotlivých stránkách jsou popsány a vysvětleny vybrané problematiky s ukázkovým příkladem.

6.1 Popis layoutu

Vzhled aplikace je rozdělen do tří základních částí a to, banner s logem a menu, obsahová část a patička s nastavení databáze a odkazem na univerzitní stránky. Celý layout je laděn do červené, černé a šedé barvy.



Obr. 14. Vzorník barev využitých ve webové aplikaci

K nadefinování vzhledu webové aplikace byly využity kaskádové styly. Tento souhrn pravidel je uložen samostatně v souboru `style.css`. Následně je připojován do hlavičky dokumentu.



Obr. 15. Screen úvodní obrazovky

6.2 Struktura webové stránky

Moderní přístup k programování webových stránek vyžaduje, aby stejné části kódu, které jsou využívány na různých místech aplikace, byly uloženy ve funkcích nebo samostatných scriptech. Tyto funkce nebo skripty uloženy v externím souboru se pak následně volají, když mají být využity. Typickým příkladem je banner s menu nebo patička. Tyto dva prvky jsou přítomny na každé stránce aplikace. Využívá se tedy vložení těchto částí kódu do funkce jednak pro lepší přehlednost a pro menší velikost výsledného kódu, ale především při jakémkoliv změně se mění kód pouze na jednom místě.

Tato aplikace využívá hned několik samostatných scriptů. V souboru **utils.php** jsou definovány veškeré potřebné funkce s hlavičkami a patičkami. Konstanty pro připojení k databázi jsou uloženy v souboru **config.php**. Soubory **__logout.php**, **__logoutfail.php**, **_auth.php** a **_heslo.php** jsou skripty potřebné pro přihlášení a odhlášení uživatele.

6.3 Popis jednotlivých stránek aplikace

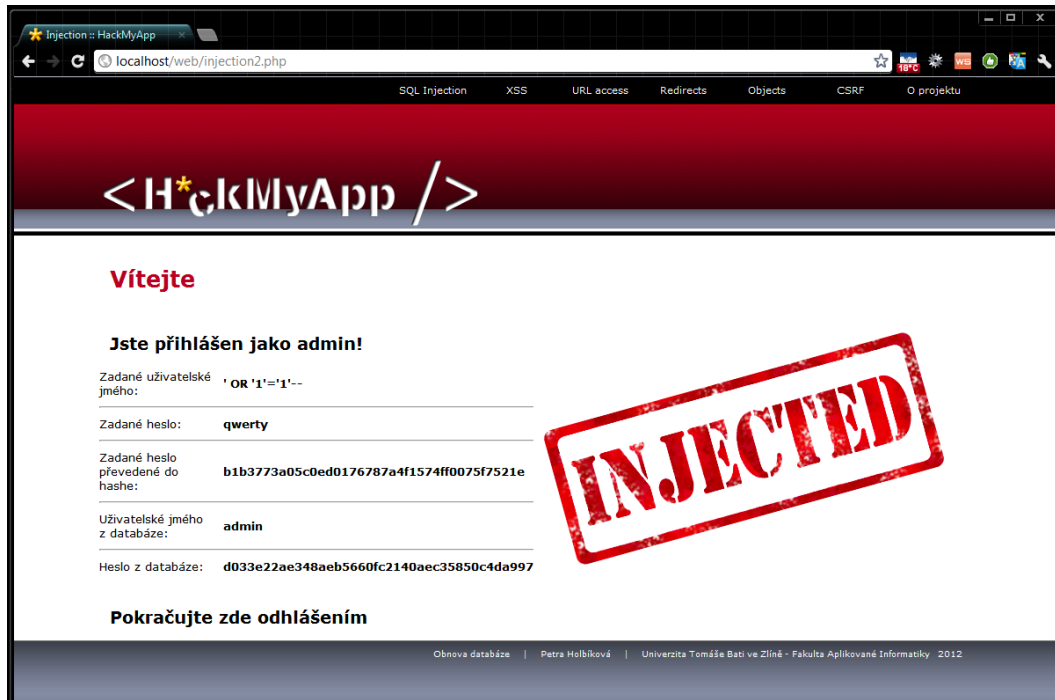
V následující části práce budou popsány jednotlivé stránky aplikace. Všechny jsou jednotné konstrukce. Jako první je uveden obecný popis bezpečnostního rizika nebo útoku. Následuje

vysvětlený příklad bez zabezpečení, tzn. například přihlašovací formulář náchylný na SQL Injection. Nakonec je uveden postup, jak se takovýmto bezpečnostním rizikům vyvarovat nebo jak ošetřit naprogramovaný script. Doplněný je názornou ukázkou, kde jsou tato ošetření implementována.

Jelikož popisu bezpečnostních rizik je v této práci věnována samostatná kapitola, budou zde popsány pouze příklady bez ochrany nebo i s ochranou proti zneužití.

6.3.1 SQL Injection

V případě SQL Injection byl vybrán jako ukázka formulář pro přihlašování uživatele. Pokud není využita žádná funkce, která by zabránila vkládání jednoduchých nebo dvojitých uvozovek, je tento formulář náchylný na SQL Injection. V tomto příkladu je přihlašování uskutečněno pomocí následujících kroků. Po vyplnění formuláře se script odešle metodou POST. Vyplněné hodnoty se uloží do sessions. V následujícím scriptu proběhne SQL dotaz, kde se vyplní hodnoty jména a hesla. Pokud databáze vrátí jakýkoliv záznam, považuje se uživatel za přihlášený. Žádná další kontrola zde už není.



Obr. 16. Screen stránky po úspěšném útoku SQL Injection

Účinná obrana proti tomuto útoku je funkce escapování nepovolených znaků na vstupu. V druhém příkladu je tato funkce již zahrnuta. Escapování se provádí pomocí funkce

addslashes(). Ta vloží před jednoduché nebo dvojité uvozovky zpětné lomítko. Porovnávají se taktéž data z databáze s daty odeslané uživatelem. Při neshodě se vyhodnotí chyba a uživatel je přesměrován zpět na přihlašovací formulář.

```
<?php
@$link = mysql_connect(MySQL_Server, MySQL_User, MySQL_Password);
if (!$link) {die('Could not connect to MySQL: '. mysql_error());}

@$db = mysql_select_db(MySQL_DB);
if (!$db) {die('Could not connect to database: '. mysql_error());}

if (get_magic_quotes_gpc())
{
    $username = $_SESSION["jmeno"];
    $password = $_SESSION["heslo"];
}
else
{
    $username = addslashes($_SESSION["jmeno"]);
    $password = addslashes($_SESSION["heslo"]);
}

$dotaz= mysql_query('SELECT jmeno, heslo FROM uzivatele WHERE
                    jmeno="' . $username .'" and heslo="' . $password .'"');
$radek= mysql_fetch_array($dotaz);
$heslodb = $radek['heslo'];
$jmenodb = $radek['jmeno'];

?>
```

Obr. 17. Využití funkce addslashes() pro escapování uvozovek



Obr. 18. Screen stránky po úspěšném přihlášení uživatele

6.3.2 Cross – Site Scripting

Pro Cross -Site Scripting byl vybrán příklad se zaměřením na typ stored. Tento typ se vyznačuje uložením útočného scriptu do databáze. Script je načten pokaždé, jakmile někdo navštíví stránku a napadne uživatele.

Typickou ukázkou tohoto typu je vkládání komentářů. Jestliže necháme data z formuláře uložit bez jakékoliv validace, může útočník vložit jakýkoliv script. Takto může stránky přesměrovat například na aplikaci obsahující škodlivý malware. Komentáře jsou v tomto případě jen obyčejně vložený záznam do databáze.


```
<?php
$db_dotaz = mysql_query('SELECT * FROM komentare');
while ($radek=mysql_fetch_array($db_dotaz)):
    echo "<span class=\"strong\">";
    echo htmlspecialchars($radek['id'], ENT_QUOTES).": ";
    echo htmlspecialchars($radek['jmeno'], ENT_QUOTES)."</span>";
    echo "<br />";
    echo htmlspecialchars($radek['komentar'], ENT_QUOTES);
    echo "<hr>";
endwhile;
?>
```

Obr. 21. Část kódu s použitím funkce htmlspecialchars()

6.3.3 Failure To Restrict URL Access

Příklad neomezení přístupu je vysvětleno na zobrazení odkazu na stránku o autorovi přihlášeným uživatelům. Stránka samotná již zabezpečení nemá. Odkaz sice nepřihlášeným uživatelům není zobrazen, avšak při znalosti URL je možné se na stránku dostat. Uživatel po přihlášení je odkázán na zabezpečenou stránku, kde je upozorněn na zobrazení nové položky v menu. Pokud se odhlásí a zadá URL na zabezpečenou stránku, je přesměrován zpět na přihlašovací formulář.

6.3.4 Unvalidated Redirect

Riziko spojené s tímto útokem spočívá v neověřeném přesměrování. Aby se tomuto nechtěnému přesměrování zabránilo, využívají se tzv. whitelisty. V demonstrované ukázce je whitelist uložen do databáze. V části kódu, která plní funkci přesměrování se data z databáze porovnávají s proměnnou, ve které je uložena adresa přesměrování. Seznam povolených adres nemusí být uložen pouze v databázi. Může se například načítat z externího textového souboru. Doporučuje se však přesměrování vůbec nepoužívat.

```
<?php
include "inc/config.php";

if (isset($_GET["url"]) && $_GET["url"]=='')
{
?>
    <script>
    history.back();
    </script>
<?php
}
elseif (isset($_GET["url"]))
{
    $url=$_GET["url"];

    @$link = mysql_connect(MySQL_Server, MySQL_User, MySQL_Password);
    if (!$link) {die('Could not connect to MySQL: ' . mysql_error());}

    @$db = mysql_select_db(MySQL_DB);
    if (!$db) {die('Could not connect to database: ' . mysql_error());}

    $query= mysql_query('SELECT url FROM whitelist WHERE url="'.$url.'");

    while ($line=mysql_fetch_array($query)):
        if ($line['url']==$url)
        {
            header("Location:".$url);
        }

    endwhile;
?>
    <script>
    history.back();
    </script>
<?php
}
else
{
?>
    <script>
    history.back();
    </script>
<?php
}
?>
```

Obr. 22. Kód kontroly pomocí whitelistu a následné přesměrování

6.3.5 Insecure Objects

Pro demonstraci nezabezpečených objektů byla vytvořena složka „upload“. V ní jsou umístěné testovací soubory. Pokud nevytvoříme pravidlo v souboru .htaccess, jsou volně dostupné. Pokud by se jednalo o interní citlivá data, mohlo by to znamenat zneužití a mít

finanční dopady. Pro ošetření se definuje pravidlo „Option –Indexes“ v souboru **.htaccess**, které zakazuje indexování složky.

6.3.6 CSRF

Jako ukázkový příklad je zde vybráno hlasování do ankety. Hlasování do ankety je uskutečňováno přes formuláře. V prvním případě, bez obrany proti CSRF útoku, jsou formuláře odeslány pomocí metody GET. Ta odesílá proměnné v URL adrese. V následujícím scriptu je ošetřené pouze, zda je v předávané proměnné uloženo id hlasované možnosti. Pokud není je uživatel přesměrován zpět. V opačném případě se script připojí k databázi a navýší se vybraná možnost o jeden hlas. Metoda GET je snadno zneužitelná, je možno upravit URL, podle toho, do jaké možnosti se má přidat hlas. Při upravené URL musí útočník čekat, jestli oběť klikne na odkaz. Využívá se proto taktéž tagu ``. Do jeho atributů se nastaví nulová výška a nulová šířka, čímž ho zneviditelníme. Takovýto tag, odkazující na anketu s předem vybranou možností, může útočník umístit například na veřejné fórum. Pokaždé, když pak někdo navštíví stránku, se automaticky hlasuje. Tento způsob je taktéž demonstrován. Jestliže uživatel navštíví stránku s XSS, nevědomě odešle hlas. Tento tag je uložen do komentářů.

V druhém příkladu je pro hlasování do ankety využito metody POST. Tato metoda je sice taktéž zneužitelná, ale je podstatně bezpečnější než metoda GET. V tomto případě se taktéž generuje autorizační token, který se ukládá do session. Taktéž je odeslán ve skrytém poli formuláře. V následujícím scriptu se tyto dvě hodnoty porovnávají. Pokud se rovnají, uloží se hlas do databáze.

```

<?php
include "inc/config.php";

session_name("anticsrf");
session_start();

if ((isset($_POST["votes"]) && $_POST["votes"]!='') || (isset($_POST["token"]) && $_POST["token"]!=''))
{
    header("Location:csrf.php");
}
elseif (isset($_POST["votes"]) && isset($_POST["token"]))
{
    if ($_POST["token"]==$_SESSION['token'])
    {
        $id = $_POST["votes"];

        @$link = mysql_connect(MySQL_Server, MySQL_User, MySQL_Password);
        if (!$link) {die('Could not connect to MySQL: ' . mysql_error());}

        @$db = mysql_select_db(MySQL_DB);
        if (!$db) {die('Could not connect to database: ' . mysql_error());}

        $query = mysql_query('SELECT votes FROM poll_token WHERE id_p="'. $id ."'");
        $line = mysql_fetch_array($query);
        $sum = $line['votes'];
        $sum++;
        $query = mysql_query('UPDATE poll_token SET votes = "'. $sum ."' WHERE id_p="'. $id ."'");

        header("Location:csrf.php?ok=1");
    }
    else
    {header("Location:csrf.php");}
}

else {header("Location:csrf.php");}

?>

```

Obr. 23. Kód na uložení dat do databáze po kontrole autorizačního tokenu

6.4 Obnova databáze

V aplikaci je zahrnuta obnova databáze pro případ jejího zničení během testování XSS nebo injection. Tato obnova je prováděna SQL dotazy. Napřed se smaže celá databáze pomocí příkazu DROP DATABASE. Následně se vytvoří databáze nová – CREATE DATABASE. Vloží se do ní tabulky a následně i data. Poslední dvě akce jsou prováděny příkazy CREATE TABLE a INSET INTO. Script následně uživatele přesměruje na **index.php**.

7 PENETRAČNÍ TESTOVÁNÍ WEBOVÉ APLIKACE

Penetrační testování je jednou z nejučinnějších metod jak se dozvědět o stavu zabezpečení vybrané aplikace. Využití testovacích nástrojů a manuálních metod se blíží reálnému útoku. Dříve se pojem penetrační testování vztahoval pouze k testování sítí. V dnešní době se testování rozšířilo na aplikace, servery, osobní počítače atd. Z pohledu testovacích metod se pentest může rozdělit na přístup Black-box a White-box. Black-box nastává v případě, že tester nezná vnitřní strukturu aplikace či jakékoliv kódy a strukturu sítě. Informace tester získává plnohodnotným útokem, který využívají hackeři, a z veřejně dostupných zdrojů. White-box testování je pravým opakem Black-boxu, tzn. tester má přístup ke kódům a vnitřní struktuře testované aplikace. V penetračním testování se využívá jak automatizovaných nástrojů, tak manuálního testování.

7.1 Scénář penetračního testování

Literatura nedefinuje přesné scénáře penetračního testování. Tyto testy se přizpůsobují konkrétním situacím či společností. Existují však doporučené postupy, které následně si jednotlivý tester přizpůsobí vlastním podmínkám a přístupům k testování.

7.1.1 Sběr informací

Tímto prvním krokem se rozumí sbírání dat jako je například IP adresa, DNS servery, vyhledávání základních informací o společnosti na internetu apod. K zjištění informací mohou pomoci také databáze, jako je například WHOIS. Tam můžeme získat již zmiňované IP adresy nebo jména vlastníků, která lze následně využít jako potenciální uživatelská jména. Ve sběru dat je nápomocná i technika Google hacking.

7.1.2 Mapování sítí

K mapování sítí se využívají automatizované nástroje - port scannery. Mezi takovéto nástroje patří například Nmap, který byl popsán v druhé kapitole. Zjišťují se otevřené porty, OS, použitý firewall a jeho pravidla. Cílem tohoto kroku je utvořit si představu o typologii a rozložení mapované sítě.

7.1.3 Automatizované testy slabin

V tomto bodě pentestů se využívají automatizované nástroje, aby tester získal předběžnou představu potencionálních bezpečnostních chyb. Automatizované nástroje šetří testerův čas. Obsahují rozsáhlé databáze známých bezpečnostních chyb. V tomto kroku by mohl být využit například nástroj OWASP ZAP, který obsahuje aktivní a pasivní scannery, útoky hrubou silou a další testovací nástroje, které zjistí náchylnost na bezpečnostní rizika.

7.1.4 Manualní testy

Jakmile automatizované testy odhalí bezpečnostní slabiny, jsou na základě těchto výsledků prováděny manuální testy. V tomto kroku velmi záleží na kreativitě a vědomostech testera.

7.1.5 Vyhodnocení bezpečnostních slabin

Po ukončení testování by měla být k dispozici přehledná zpráva, ve které jsou uvedeny výsledky pentestů - soubor nalezených bezpečnostních slabin, výsledky testů, případně i doporučení způsobu opravy slabin.

ZÁVĚR

Tato bakalářská práce se zabývá tématem bezpečnosti webových aplikací.

V úvodní části obsahuje analýzu současných bezpečnostních rizik týkajících se webových aplikací. Je zde zmíněná také statistika vypracovaná společností WhiteHat Security. Dále následuje rozbor jednotlivých bezpečnostních rizik podle žebříčku OWASP Top 10 z roku 2010. U každého bezpečnostního rizika je uveden popis problematiky, jaké je zneužití a možný dopad.

Druhá kapitola se zabývá rozbohem dostupných testovacích nástrojů. Uveden je základní popis software, jeho stručná historie vývoje a hlavní programátoři, taktéž zahrnuté nástroje, licenční podmínky a podporované platformy. Popsané testovací nástroje byly vybírány podle dostupnosti a doporučení vyhledaných na internetu.

Ve třetí části práce jsou zhodnoceny možnosti automatizovaného a manuálního testování. Nejprve jsou vysvětleny dva základní životní cykly vývoje aplikací v kontextu s testováním bezpečnosti a následně byly popsány klady a zápory manuálního a automatizovaného testování. Závěr této kapitoly vysvětluje dva možné způsoby hodnocení rizik a to, systém DREAD a systém OWASP Risk Rating Methodology.

V praktické části bylo úkolem naprogramovat webovou aplikaci. Její popis je uveden v poslední kapitole této práce. Nejprve jsou popsány programovací jazyky a technologie, které byly při tvorbě aplikace využity. Následuje vlastní popis aplikace a nakonec možný scénář penetračního testování.

Aplikace názorně ukazuje rozdíly mezi zabezpečenými a nezabezpečenými prvky webů, které je možné si vyzkoušet. Může být využita k výukovým účelům v předmětech zabývajících se webovými technologiemi a nastínit studentům problematiku bezpečnostních rizik a ukázat případnou obranu proti nim.

CONCLUSION

This thesis deal with theme of web application security.

An analysis of today's security risks regrading web applications is developed in the opening part. Also statistics formulated by WhiteHat Security is mantioned here, followed by an analysis of each security risks according to OWASP Top 10 of year 2010. For each security risk is given the description of the problem, what is abuse and the potential impact.

Second chapter discusses analysis options of automated and manual testing. Firstly, there are two basic explanation of the application life cycles development in the context of testing the safety and subsequently described the pros and cons of manual and automated testing. Conclusion of this chapter explains two possible methods of risk assessment, the system DREAD and OWASP Risk Rating Methodology.

In the practical part was the mission including programming a web application. Its description is given in the last chapter of this work. Programming languages and technologies that were used to create applications are described. Then there is the description of the application and eventually penetration testing scenario.

Application demonstrates the difference between secured and unsecured elements of web sites which may be tried. It can be used for educational purposes in study programs dealing with web technologies and outline the issues of security risks to students and show possible defense against them.

SEZNAM POUŽITÉ LITERATURY

- [1] HOWARD, Michael a David LEBLANC. Bezpečný kód: [techniky a strategie tvorby bezpečných webových aplikací]. Vyd. 1. Brno: Computer Press, 2008, 895 s. ISBN 978-802-5120-507.
- [2] CROSS, Michael. Developer's guide to web application security. Rockland, MA: Syngress Publishing, c2007, 489 s. ISBN 159749061X.
- [3] STUTTARD, Dafydd a Marcus PINTO. The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws. Indianapolis: John Wiley & Sons, 2011, 489 s. ISBN 1118079612.
- [4] HOFFMAN, Billy a Bryan SULLIVAN. Ajax Security. Boston: Addison-Wesley Professional, 2007. ISBN 0132701928.
- [5] SOLARI, Carlos Curtis. Security in a Web 2.0+ World: A Standards-Based Approach. Chichester: John Wiley & Sons, 2011. ISBN 0470971088.
- [6] SHIFLETT, Chris. Essential PHP security. Sebastopol: O'Reilly, 2006, 109 s. ISBN 05-960-0656-X.
- [7] OWASP FOUNDATION. OWASP Foundation [online]. 13. 1. 2012 [cit. 2012-01-19]. Dostupné z: www.owasp.org
- [8] Anonymous Czech Republic. *Facebook* [online]. 2012 [cit. 2012-05-27]. Dostupné z: <https://www.facebook.com/AnonymousCzech>
- [9] Anonymous (group). *Wikipedia: The Free Encyclopedia* [online]. 2012 [cit. 2012-05-27]. Dostupné z: [http://en.wikipedia.org/wiki/Anonymous_\(group\)](http://en.wikipedia.org/wiki/Anonymous_(group))
- [10] LulzSec. *Wikipedia: The Free Encyclopedia* [online]. 2012 [cit. 2012-05-27]. Dostupné z: <http://en.wikipedia.org/wiki/LulzSec>
- [11] Cross Site Request Forgery. *SOOM.cz* [online]. 2008 [cit. 2012-05-27]. Dostupné z: <http://www.soom.cz/index.php?name=articles/show&aid=484>
- [12] Google Hacking 1.díl. *EMag.cz: technologický magazín* [online]. 2007 [cit. 2012-05-27]. Dostupné z: <http://www.emag.cz/google-hacking-1dil/>

- [13] OWASP Top 10 for .NET developers part 7: Insecure Cryptographic Storage. *Troy Hunt's Blog* [online]. 2011 [cit. 2012-05-27]. Dostupné z: <http://www.troyhunt.com/2011/06/owasp-top-10-for-net-developers-part-7.html>
- [14] OWASP Top 10 for .NET developers part 10: Unvalidated Redirects and Forwards. *Troy Hunt's Blog* [online]. 2011 [cit. 2012-05-27]. Dostupné z: <http://www.troyhunt.com/2011/12/owasp-top-10-for-net-developers-part-10.html>
- [15] *CIRT.net: suspicion breeds confidence* [online]. 2010 [cit. 2012-05-27]. Dostupné z: <http://cirt.net/>
- [16] *SeleniumHQ: Web Application Testing System* [online]. 2012 [cit. 2012-05-27]. Dostupné z: <http://seleniumhq.org>
- [17] Skipfish: Web Application Security Scanner. *Google Code* [online]. 2010 [cit. 2012-05-27]. Dostupné z: <http://code.google.com/p/skipfish/>
- [18] *Nmap.cz: Nmap - Scanner Portů* [online]. 2012 [cit. 2012-05-27]. Dostupné z: <http://nmap.cz/>
- [19] *WireShark* [online]. 2012 [cit. 2012-05-27]. Dostupné z: <http://www.wireshark.org/>
- [20] INTERNET USAGE STATISTICS: World Internet Users and Population Stats. *Internet World Stats: Usage And Polulation Statistics* [online]. 2012 [cit. 2012-05-27]. Dostupné z: <http://www.internetworldstats.com/stats.htm>
- [21] Internet Usage in Europe: World Internet Users and Population Stats. *Internet World Stats: Usage And Polulation Statistics* [online]. 2012 [cit. 2012-05-27]. Dostupné z: <http://www.internetworldstats.com/stats4.htm>
- [22] *Český Statistický Úřad* [online]. 2011 [cit. 2012-05-27]. Dostupné z: <http://www.czso.cz/csu/redakce.nsf/i/home>
- [23] Vodopádový model životního cyklu software (The waterfall Life cycle). *Testování softwaru* [online]. 2012 [cit. 2012-05-30]. Dostupné z: <http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/vodopadovy-model/>

- [24] Chapter 3 □ Threat Modeling. *MSDN Library* [online]. 2003 [cit. 2012-05-30]. Dostupné z: <http://msdn.microsoft.com/en-us/library/ff648644.aspx>
- [25] *Websecurify* [online]. 2012 [cit. 2012-06-01]. Dostupné z: <http://www.websecurify.com/>
- [26] What is a Security Fuzzer?. *CGISecurity.com: Web Application News And More* [online]. 2008 [cit. 2012-06-03]. Dostupné z: <http://www.cgisecurity.com/questions/securityfuzzer.shtml>
- [27] *Zaproxy* [online]. 2012 [cit. 2012-06-03]. Dostupné z: <http://code.google.com/p/zaproxy/>
- [28] *W3C* [online]. 2012 [cit. 2012-06-05]. Dostupné z: www.w3c.org
- [29] History of PHP and related projects. *PHP: Hypertext Preprocessor* [online]. 2009 [cit. 2012-06-06]. Dostupné z: <http://php.tonnikala.org/manual/en/history.php.php>
- [30] Apache HTTP Server. *Wikipedia: The Free Encyclopedia* [online]. 2012 [cit. 2012-06-06]. Dostupné z: http://en.wikipedia.org/wiki/Apache_HTTP_Server
- [31] *MySQL: The world's most popular open source database* [online]. 2012 [cit. 2012-06-06]. Dostupné z: <http://www.mysql.com/>
- [32] *WhiteHat Security* [online]. 2012 [cit. 2012-06-06]. Dostupné z: <https://www.whitehatsec.com/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ACTA	Anti – Countefeiting Trade Agreement
AES	Advanced Encryption Standard
CIA	Central Intelligence Agency
CSFR	Cross – Site Request Forgery
CSS	Cascading Style Sheets
DDoS	Distributed Denial of Service
DOM	Document Object Model
DoS	Denial of Service
FBI	Federal Bureau of Investigation.
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol Secure
HTTPS	Hypertext Transfer Protocol Secure
HW	Hardware
LDAP	Lightweight Directory Access Protocol
MD5	Message – Digest Algorithm
OS	Operační Systém
OSA	Ochranný Svaz Autorský
OWASP	Open Web Application Security Project
PBS	Public Broadcasting Service
PHP	PHP: Hypertext Preprocessor
PHP/FI	Personal Home Page Tool / Form Interpreter
SEQUEL	Structured English Query Language
SGML	Standard Generalized Markup Language
SHA	Secure Hash Algorithm

SQL	Structured Query Language
SSL	Secure Sockets Layer
SW	Software
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
URL	Unique Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XSS	Cross – Site Scripting

SEZNAM OBRÁZKŮ

Obr. 1. Vlajka hackerské skupiny Anonynouse [9]	12
Obr. 2. Logo hackerské skupiny LulSec [10]	13
Obr. 3. Vývoj útoku XSS typu stored	17
Obr. 4. Vývoj útoku XSS typu reflected.....	18
Obr. 5. Vývoj útoku XSS typu DOM based	19
Obr. 6. Logo OWASP ZAP [7].....	27
Obr. 7. Logo Selenium IDE [16].....	29
Obr. 8. Logo Skipfish [17].....	30
Obr. 9. Logo Nikto2 [15]	31
Obr. 10. Logo Websecurify [25]	31
Obr. 11. Vodopádový model vývoje SW	34
Obr. 12. Logo PHP [29]	42
Obr. 13. Logo MySQL [31].....	43
Obr. 14. Vzorník barev využitých ve webové aplikaci	45
Obr. 15. Screen úvodní obrazovky	46
Obr. 16. Screen stránky po úspěšném útoku SQL Injection	47
Obr. 17. Využití funkce addslashes() pro escapování uvozovek.....	48
Obr. 18. Screen stránky po úspěšném přihlášení uživatele.....	49
Obr. 19. Úspěšný Cross – Site scripting útok pomocí alert()	50
Obr. 20. Část kódu s použitím funkce strip_tags().....	50
Obr. 21. Část kódu s použitím funkce htmlspecialchars().....	51
Obr. 22. Kód kontroly pomocí whitelistu a následné přesměrování.....	52
Obr. 23. Kód na uložení dat do databáze po kontrole autorizačního tokenu.....	54

SEZNAM TABULEK

Tab. 1. Žebříčky OWASP Top 10 pro roky 2004, 2007 a 2010	13
Tab. 2. Vybrané pokročilé operátory pro vyhledávání pomocí vyhledávače Google	21
Tab. 3. Převod průměrů na úroveň.....	38
Tab. 4. Celkové vyhodnocení závažnosti.....	38

SEZNAM GRAFŮ

Graf 1 Top 10 zranitelností webových aplikací z roku 2010 (všechna odvětví) [32]	14
Graf 2 Přehled náchylnosti na zranitelnost webových aplikací podle odvětví z roku 2009 [32]	15
Graf 3 Uživatelé internetu rozdělení podle světových regionů [20]	26

SEZNAM PŘÍLOH

PI CD se zdrojovými kódy webové aplikace