Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Bezpečnostní pluginy systému WordPress

Bc. Václav Vodička

© 2018 ČZU v Praze

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Václav Vodička

Informatika

Název práce

Bezpečnostní pluginy systému WordPress

Název anglicky

WordPress Security Plugins

Cíle práce

Cílem této diplomové práce je analyzovat a interpretovat efektivnost vybraných pluginů, které slouží k zabezpečení webových stránek založených na systému WordPress. Součástí hlavního cíle je vzájemná komparace vybraných pluginů.
Dílčím cílem práce je definování základních principů hackingu a programovacích jazyků. Získané znalosti budou podkladem pro analýzu efektivnosti pluginů, interpretaci výsledků a využití nástrojů a technik, které vedou k neoprávněnému průniku do systému.

Metodika

K teoretické části bude využita odborná literatura, publikované odborné a vědecké články a expertní šetření z oblasti hackingu.

Pro analýzu efektivnosti každého pluginu budou vytvořeny webové stránky běžící na systému WordPress. Na stránkách bude pomocí dostupných nástrojů a vybraných technik hackingu zkoumána jejich bezpečnost. Prováděné útoky na webové stránky budou znázorněny ukázkou komentovaného kódu nebo slovně vysvětleny a doplněny o obrázky.

Zjištěné nedostatky každého bezpečnostního pluginu budou interpretovány a použity pro vzájemnou komparaci všech pluginů.

Doporučený rozsah práce

60 – 80 stran

Klíčová slova

hacking, web, bezpečnost, útoky, WordPress, plugin

Doporučené zdroje informací

ERICKSON, Jon. Hacking: The Art of Exloitation. 2nd Edition. San Francisco: No Strach Press, 2008. ISBN 1-59327-144-1.

KIM, Peter. Hacking: praktický průvodce penetračním testováním. Přeložil Jan POKORNÝ. Brno: Zoner Press, 2015. Encyklopedie Zoner Press. ISBN 978-80-7413-313-8.

KIM, Peter. The Hacker Playbook 2: Practical Guide To Penetration Testing. North Charleston: Practical Guide To Penetration Testing, 201n. l. ISBN 978-1512214567.

LONG, Johnny. Google hacking. Brno: Zoner Press, 2005. Encyklopedie Zoner Press. ISBN 80-86815-31-5.

REGALADO, Daniel, Shon HARRIS, Allen HARPER, Chris EAGLE, Jonathan NESS, Branko SPASOJEVIC, Rayn LINN a Stephen SIMS. Gray Hat Hacking: The Ethical Hacker's Handbook. Fourth Edition. United States: McGraw-Hill Education, 2015. ISBN 978-0-07-18328-0.

STUTTARD, Dafydd a Marcus PINTO. The web application Hacker's Handbook: Findindg and Exploiting Security Flaws. 2nd edition. Indianapolis: John Wiley, 2011. ISBN 978-1-118-02647-2.

VRÁNA, Jakub. 1001 tipů a triků pro PHP. Brno: Computer Press, 2010. ISBN 978-80-251-2940-1.

Předběžný termín obhajoby

2017/18 LS – PEF

Vedoucí práce

Ing. Václav Lohr, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 31. 10. 2017

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 1. 11. 2017

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 22. 03. 2018

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Bezpečnostní pluginy systému WordPress" jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 09.03.2018 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Václavu Lohrovi, Ph.D. za jeho cenné rady při psaní této diplomové práce.

Bezpečnostní pluginy systému WordPress

Abstrakt

Tato práce analyzuje efektivnost vybraných bezpečnostních pluginů systému WordPress. Zjištěné výsledky jsou interpretovány a využity k vzájemné komparaci pluginů. Součástí práce je definování základních principů hackingu a programovacích jazyků, které jsou podkladem pro analýzu bezpečnostních pluginů, provedení útoků a interpretaci zjištěných výsledků. Analyzováno bylo deset nejpoužívanějších pluginů z hlediska obrany proti šesti nejběžnějším útokům na systém WordPress.

Pro každý bezpečnostní plugin byla vytvořena webová stránka s WordPressem, kam byl daný plugin nainstalován. Pomocí dostupných nástrojů a vybraných technik hackingu byla zkoumána bezpečnost webových stránek. Zjištěné hodnoty u každého prováděného útoku a pluginu byly bodovací metodou ohodnoceny. Na základě těchto bodů bylo stanoveno pořadí pluginů. Metodou pořadí, tedy součtem pořadí z jednotlivých komparací bylo stanoveno pořadí celkové. Z důvodu zvýšení objektivity komparace byla také vytvořena webová aplikace, která na základě stanovených priorit uživatelem a zjištěných výsledků z analýzy, vyhodnotí celkové pořadí bezpečnostních pluginů.

Až do napsání této diplomové práce neexistoval vědecký výzkum, který by analyzoval efektivnost bezpečnostních pluginů systému WordPress. Zjištěné poznatky z této diplomové práce mohou sloužit vývojářům při vývoji nových bezpečnostních pluginů, správě pluginů již vyvinutých nebo při tvorbě webových aplikací, které nutně nemusí být založené na systému WordPress.

Klíčová slova: hacking, web, bezpečnost, útoky, WordPress, plugin

WordPress Security Plugins

Abstract

This work analyses the effectiveness of selected WordPress security plugins. The results are interpreted and used to compare the plugins. The work includes definition of basic principles of hacking and programming languages, which are the basis for analysing security plugins, performing attacks and interpreting the results. The 10 most commonly used plugins have been analysed for protection against the 6 most common attacks on WordPress.

For each security plugin was created a WordPress site where the plugin was installed. Using the available tools and selected hacking techniques, site security has been investigated. The detected values for each attack and plugin were evaluated by the scoring method. Based on these points, the order of the plugins was determined. The overall order was determined by the ranking method as the sum of the order of the individual comparisons. To increase the objectivity of the comparison, a web-based application has been. The application evaluates the overall order of security plugins based on user-defined priorities and the results obtained from the analysis.

Until the writing of this diploma thesis there was no scientific research that would analyse the effectiveness of WordPress security plugins. The information obtained from this thesis may be useful to developers in developing new security plugins, managing plugins that are already developed, or creating web applications that may not run on WordPress.

Keywords: hacking, website, security, attacks, WordPress, plugin

OBSAH

Seznam obrázků

Seznam tabulek

1 Úvod

V době psaní této práce běží 28 % všech webových stránek na systému WordPress a jeho popularita stále roste. Jedná se tedy o nejpoužívanější Content Management System (CMS) na trhu. Díky široké škále dostupných pluginů, lze pomocí WordPressu vytvořit standardní webovou prezentaci firmy či dokonce e-shop. S větší popularitou roste i větší zájem vývojářů, kteří vytváří nové pluginy a rozšiřují tím možnosti využití WordPressu.

Každá webová aplikace, ať už malá nebo velká obsahuje citlivá data, která je potřeba chránit. Bezpečnost tak hraje klíčovou roli při vývoji webových stránek a především těch, které shromažďují důvěrné informace o uživatelích. Ochrana osobních údajů a jejich bezpečné uložení je navíc aktuálním tématem, protože v květnu 2018 vejde v účinnost Obecné nařízení na ochranu osobních údajů neboli GDPR (General Data Protection Regulation).

WordPress má ve srovnání s ostatními CMS jako Joomla nebo Drupal více nalezených zranitelností. Pokud k počtu reportů odeslaných na CVE Details zahrneme i popularitu WordPressu, a tedy více lidí, kteří jsou schopni zranitelnost odhalit, pak nelze říci, že by byl méně bezpečný než ostatní CMS. Nehledě na tento fakt, je každá nově objevená zranitelnost ihned opravena v nové verzi.

Zásadním bezpečnostním problémem WordPressu jsou ovšem pluginy a šablony, které si do něj uživatelé instalují. Pluginy jsou vytvářeny nezávislými vývojáři, kteří často nevěnují dostatečnou pozornost k ochraně citlivých dat nebo svou neznalostí dovolí, aby v jejich pluginech vznikly bezpečnostní díry. Tým WordPressu sice kontroluje bezpečnost každého pluginu před jeho vložením na své stránky, avšak nedokáže nalézt všechny jeho zranitelnosti. Z těchto důvodů je nejčastější příčinou hacknutých webových stránek existence zranitelnosti v nainstalovaném pluginu. Právě zranitelnost v nainstalovaném pluginu byla využita v rámci této diplomové práce při útoku na webové stránky, a tedy v analýze bezpečnostních pluginů z hlediska obrany proti SQL Injection a Cross-Site Scripting.

Jedinou možností, jak se bránit útokům a tím chránit svá citlivá data ve WordPressu, je instalace bezpečnostního pluginu. Většina takových pluginů obsahuje funkce jako je např. firewall, monitorování neočekávaných událostí, hledání malwaru nebo blokování uživatelů po několika neúspěšných pokusech o přihlášení. WordPress sám o sobě žádnou takovou funkci neobsahuje a je proto důležité mít bezpečnostní plugin nainstalován.

V současné době existuje několik článků v anglickém jazyce, které porovnávají bezpečnostní pluginy z hlediska nabízených funkcí. V žádném z nich není ovšem jejich funkčnost otestována. Právě to bylo podkladem pro výběr tématu této diplomové práce, která analýzou efektivnosti bezpečnostních pluginů a jejich vzájemnou komparací rozšiřuje výzkum v oblasti bezpečnosti webových aplikací. Zjištěné poznatky z této diplomové práce mohou například sloužit vývojářům při vývoji nových bezpečnostních pluginů, správě pluginů již vyvinutých nebo při tvorbě webových aplikací, které nutně nemusí být založené na systému WordPress.

2 Cíle a metodika práce

2.1 Cíl práce

Cílem této diplomové práce je analyzovat a interpretovat efektivnost vybraných pluginů, které slouží k zabezpečení webových stránek založených na systému WordPress. Součástí hlavního cíle je vzájemná komparace vybraných pluginů.

Dílčím cílem práce je definování základních principů hackingu a programovacích jazyků. Získané znalosti budou podkladem pro analýzu efektivnosti pluginů, interpretaci výsledků a využití nástrojů a technik, které vedou k neoprávněnému průniku do systému.

2.2 Metodika práce

K teoretické části bude využita odborná literatura, publikované odborné a vědecké články a expertní šetření z oblasti hackingu.

Pro analýzu efektivnosti každého pluginu budou vytvořeny webové stránky běžící na systému WordPress. Na stránkách bude pomocí dostupných nástrojů a vybraných technik hackingu zkoumána jejich bezpečnost. Prováděné útoky na webové stránky budou znázorněny ukázkou komentovaného kódu nebo slovně vysvětleny a doplněny o obrázky.

Zjištěné nedostatky každého bezpečnostního pluginu budou interpretovány a použity pro vzájemnou komparaci všech pluginů.

3 Teoretická východiska

Webové stránky jsou všeobecným pojmem pro statické či dynamické stránky. Jelikož nemá smysl zkoumat bezpečnost statických stránek, tak v této práci bude používán především výraz webová aplikace. Webové aplikace představují složitější webové stránky, které generují dynamický obsah z libovolného uložiště, obsahují důvěrné informace a často také umožňují přihlašování uživatelů.

V následující části diplomové práce jsou vysvětleny základní principy hackingu a programovacích jazyků.

3.1 WordPress

WordPress je CMS (Content Management System), resp. systém pro správu obsahu webových stránek (adaptic.cz), který je napsán v PHP a využívá MySQL databázi (WordPress.org). Celkem 28 % ze všech webových stránek běží na WordPressu a řadí ho to tak mezi nejpoužívanější CMS (WordPress.com). Zásluhu na tom má open-source licence, velká komunita uživatelů, časté aktualizace a spousta pluginů, které si lze do CMS nainstalovat (whichcmstochoose.com, 2017).

Všechny zmíněné výhody vedou k jedné hlavní nevýhodě WordPressu, kterou je nízká bezpečnost. Vysoká popularita přináší pozornost hackerů, pro které je díky open-source licenci snadné najít ve zdrojovém kódu bezpečnostní díru. K útokům také využívají nedostatečně chráněné pluginy, které jsou vytvořeny uživateli WordPressu. (Moreno, 2014) (McKinnon, 2017)

3.2 Webový server

Pojem webový server se dá vyložit jako počítač, na kterém běží program umožňující přístup k souborům na něm uložených. Současně se právě i tento program nazývá webovým serverem. Soubory, které jsou uloženy na tomto počítači, nejčastěji představují jednotlivé stránky webu a většinou jsou napsány ve značkovacím jazyku HTML (HyperText Markup Language). (Web Developers Notes)

Komunikace mezi uživatelem stránek, resp. webovým prohlížečem a serverem probíhá pomocí HTTP (HyperText Transfer Protocol), bezpečnějšího HTTPS (HyperText Transfer Protocol Secure) nebo nejnovějšího protokolu HTTP/2.0. Uživatel pomocí webového prohlížeče navštíví určitou stránku zadáním cesty k souboru, resp. URL adresy (Uniform

Resource Locator). URL obsahuje připojovací protokol, IP adresu nebo doménové jméno webového serveru, cestu k souboru a popř. číslo portu nebo parametry stránky (adaptic.cz). Webový server, na který je požadavek odeslán ho zpracuje a pošle odpověď v podobě obsahu souboru, tedy nejčastěji HTML. Tento proces je příkladem výpočetního modelu klient/server. (Rouse, 2015) (Web Developers Notes)

3.3 HTTP

HTTP je základním komunikačním protokolem používaným k přístupu na webové stránky. Funguje na principu zpráv, kdy klient pošle požadavek a server odpověď. Každá tato zpráva se skládá z hlavičky a těla. HTTP je zároveň aplikačním protokolem, který využívá transportní protokol TCP (Transmission Control Protocol). Každé síťové spojení aplikačního a transportního protokolu je rozlišeno číslem, tzv. portem. HTTP má nejčastěji port 80 a HTTPS port 443. (Sochor, 2013) (Clark, 2013) (Stuttard, a další, 2011 str. 40)

HTTP definuje osm metod, kterými lze přistupovat přes URL adresu k souborům na serveru. Jedná se o metodu *GET*, *POST*, *PUT*, *HEAD*, *DELETE*, *TRACE*, *OPTIONS* a *CONNECT*. Webové služby, které podporují tyto akce se nazývají službami zdrojově orientovanými a říká se, že odpovídají architektuře REST (Representational State Transfer). (Clark, 2013)

Nejpoužívanější metodou je *GET* a *POST*. Metoda *GET* slouží k přijímání zdrojů ze serveru. Získané zdroje se mohou lišit podle parametrů v URL adrese. Všechny URL adresy se zobrazují v prohlížeči a jsou uchovány v historii prohlížeče a na webovém serveru. To sebou nese bezpečnostní riziko, a proto by parametry neměly obsahovat citlivé informace. (Stuttard, a další, 2011 str. 42)

Při použití metody *POST* jsou parametry odesílány přímo v těle zprávy a nejsou tak viditelné v URL. Díky tomu se parametry neukládají do historie prohlížeče ani na webový server. Právě proto se tato metoda hodí ke spouštění funkcí na serveru, které upravují nebo přidávají data v datovém uložišti. (Yaworski, 2016 str. 11) (Stuttard, a další, 2011 str. 43) (W3schools)

Metoda *HEAD* je identická s metodou *GET*, avšak server v odpovědi neposílá tělo zprávy, nýbrž pouze hlavičku. Metoda *PUT* slouží ke spuštění funkce, která pracuje s existujícím objektem na serveru. Metoda *DELETE* slouží k mazání zdrojů, *TRACE* k testování příchozích požadavků na server, *OPTIONS* informuje o tom, které HTTP metody jsou na

serveru k dispozici a metoda *CONNECT* je vyhrazena pro proxy server. (Yaworski, 2016 stránky 11,12) (Stuttard, a další, 2011 str. 43)

3.3.1 Hlavičky

Každý požadavek nebo odpověď se skládá z těla a hlavičky. Hlavička obsahuje hodnoty, jinak řečeno hlavičky, které specifikují obsah těla zprávy a poskytují další potřebné informace pro server nebo klienta. Podle (W3C) je celkem 47 možných hodnot, z nichž některé jsou podle (Stuttard, a další, 2011 stránky 45,46) pro neobvyklé účely, a tedy méně využívané. Autoři dále dodávají, že dále zmíněné hlavičky jsou důležité z hlediska bezpečnosti webových aplikací. (Jackson, 2016) seznam prodlužuje o některé další.

Hodnoty společné pro požadavek i odpověď:

- **Connection** informuje koncové zařízení o tom, zda má ukončit spojení nebo ho nechat otevřené pro budoucí zprávy.
- **Content-Encoding** určuje v jakém kódování je tělo zprávy, např. gzip.
- **Content-Length** je délka zprávy v bajtech.
- **Content-Type** specifikuje typ obsahu zprávy, např. text/html pro HTML dokument.
- **Transfer-Encoding** určuje všechna kódování, která byla během přenosu přes HTTP aplikována na tělo zprávy.

Hlavičky požadavku:

- **Accept** informuje server o tom, jaký typ obsahu je klient ochotný přijmout.
- **Accept-Encoding** informuje server o tom, jaké kódování obsahu je klient ochotný přijmout.
- **Authorization** obsahuje informaci o použité HTTP autentizaci a přihlašovací údaje.
- **Cookie** předává serveru tzv. cookies, které jsou blíže popsané v kapitole 3.5.
- **Host** specifikuje název hostitele, který je v URL požadavku.
- **If-Modified-Since** určuje, kdy prohlížeč naposledy přijal požadovaný objekt neboli zdroj. Pokud se zdroj od té doby nezměnil, pak server pošle odpověď bez těla zprávy s HTTP kódem 304. Prohlížeč tak může načíst objekt ze své mezipaměti.
- **If-None-Match** specifikuje tzv. entity tag vydaný serverem při posledním požadavku na objekt. Server tento tag používá k určení, zda může prohlížeč načíst objekt ze své mezipaměti.

- **Origin** se používá při ajaxovém požadavku na jinou doménu k indikaci, z které domény původně požadavek pochází.
- **Referer** specifikuje URL, ze které byl odeslán požadavek.
- **User-Agent** poskytuje informaci o verzi prohlížeče či jiného klientského softwaru, který vygeneroval požadavek.

Hlavičky odpovědi:

- **Access-Control-Allow-Origin** indikuje, zda může být zdroj získán z jiné domény či nikoliv.
- **Cache-Control** předává prohlížeči pokyny k ukládání do mezipaměti.
- **ETag** je již zmiňovaný entity tag, který je následně prohlížečem odeslán v požadavku a informuje server o tom, jakou verzi objektu prohlížeč drží ve své mezipaměti.
- **Expires** informuje prohlížeč o tom, po jakou dobu je obsah v těle zprávy validní, a tedy jak dlouhou může prohlížeč načítat objekt ze své mezipaměti.
- **Location** se používá při přesměrování a specifikuje cílovou URL.
- **Server** poskytuje informace o použitém webovém serveru.
- **Set-Cookie** nastavuje cookies v prohlížeči, které jsou následně prohlížečem zasílány zpět v požadavcích.
- **WWW-Authenticate** určuje, které typy autentizací server podporuje.
- **X-Frame-Options** specifikuje, zda a jak může být aktuální odpověď načtena v rámci prohlížeče.
- **Content-Security-Policy** definuje zdroje obsahu, které může prohlížeč načíst. Tato zásada pomáhá předcházet útokům jako je například Cross-Site Scripting (XSS).
- **Strict-Transport-Security** informuje prohlížeč, aby vynutil HTTPS připojení.
- **Public-Key-Pins** informuje prohlížeč o přidružení veřejného klíče s určitým webovým serverem, aby zabránil Man-in-the-Middle (MitM) útokům pomocí padělaných certifikátů.
- **X-Content-Typ-Options** zabraňuje prohlížeči změnit deklarovaný typ obsahu v hlavičce Content-Type.
- **X-XSS-Protection** vynucuje Cross-Site Scripting (XSS) filtr, který zastavuje načítání stránek při detekci XSS útoku.

3.3.2 HTTPS

HyperText Transfer Protocol Secure, na rozdíl od HTTP využívá kromě nezabezpečeného transportního protokolu TCP, také vrstvu SSL (Secure Sockets Layer), která poskytuje protokol pro šifrovanou komunikaci. Vrstva SSL byla dnes již nahrazena bezpečnější vrstvou TLS (Transport Layer Security), avšak pojmenování se často zaměňuje nebo se používá vzájemné spojení TLS/SSL. (Stuttard, a další, 2011 str. 49) (Kangas, 2016)

TLS/SSL využívá kombinaci symetrické a asymetrické šifry. Požadavek, který klient posílá na server je šifrován pomocí klíče symetrické šifry, kterým je pak také dešifrován. Před odesláním zprávy je proto nejdříve nutné odeslat symetrický klíč a provést tedy, tzv. podání ruky, anglicky „*SSL Handshake*“, kdy dochází k ověření identity obou zařízení. V této fázi klient požádá server o certifikát, který obsahuje veřejný klíč asymetrické šifry. Po jeho získání ověří, zda se jedná o důvěryhodný certifikát a pokud ano, tak klíčem zašifruje symetrický klíč a pošle ho na server. Server zprávu dešifruje privátním klíčem a zašle klientovi odpověď o úspěšném získání symetrického klíče. V další fázi pak probíhá samotná komunikace. K novému ověření totožnosti dochází po vygenerování nového symetrického klíče, např. po zavření a otevření prohlížeče. (Walls, 2006 stránky 344,345) (DigiCert) (Pedersen, 2011 stránky 569,570)

3.3.3 HTTP/2

HTTP/2 je nejnovější verze protokolu HTTP, který přinesl řadu vylepšení. Hlavním přínosem je multiplexing, tzn. zpracování více požadavků a odpovědí najednou. Dochází tím ke zlepšení výkonu aplikace, čemuž přispívá také komprese HTTP hlaviček. Jednotlivé zprávy jsou navíc formovány do rámů neboli proudů, kterým lze nastavit priority. (Grigorik, 2017)

Pro kompresi HTTP hlaviček byl zprvu využíván protokol SPDY s kompresí ZLIB, který vyvinul Google. V roce 2012 byl ovšem publikován CRIME (Compression Ration Info-leak Mafe Easy) útok proti SPDY, který vede k Session Hijacking (únos relace). Pro kompresi se proto nyní používá algoritmus HPACK. (Grigorik, 2017)

3.3.4 HTTP autentizace

HTTP protokol poskytuje vlastní mechanizmus pro autentizaci uživatelů, přičemž pro ověřování používá několik schémat. Schémata se liší v bezpečnosti, jejich dostupnosti

v prohlížeči a serverovém softwaru. V následujícím výčtu jsou běžně používaná schémata. (Stuttard, a další, 2011 str. 50) (Hunt, a další, 2017)

- **Basic** – přihlašovací údaje zakódované v base64.
- **Digest** – přihlašovací údaje jsou zašifrovány pomocí md5.
- **Bearer** – využívá tokeny k ověření přístupu zabezpečených zdrojů.
- **HOBA** – HTTP Origin-Bound Autentication, k ověření využívá digitální podpis.
- **NTLM** – využívá Windows NTLM protokol.

V případě, že je nutné provést autentizaci, tak server pošle v odpovědi HTTP kód 401 (Unauthorized) a v hlavičce *WWW-Authenticate* pošle dostupná ověřovací schémata. Klient, resp. prohlížeč nejprve zobrazí formulář pro zadání přihlašovacích údajů a ty společně se zvoleným schématem odešle v hlavičce svého požadavku. (Hunt, a další, 2017)

3.4 HTTP Proxy server

Proxy server zprostředkovává přístup mezi prohlížečem a webovým serverem. Dále mohou poskytovat další služby jako je například ukládání dat do mezipaměti (caching) nebo ověřování a řízení přístupu uživatelů. Proxy server se využívá při útocích na webové aplikace, protože umožňuje prohlížet a měnit všechny požadavky a odpovědi, a to i přesto, že komunikace probíhá přes HTTPS. (Stuttard, a další, 2011 str. 50)

3.5 Cookies

Cookies jsou uživatelská data uložená v souboru klientského počítače (Olsson, 2016). Každý jednotlivý údaj v cookies se nazývá cookie a skládá se z názvu a hodnoty. Cookies jsou odesílány v každém požadavku pomocí HTTP hlavičky *Cookie*. Nový údaj může být přidán serverem skrze hlavičku *Set-Cookie* v odpovědi. Kromě názvu a hodnoty lze nastavit cookie následující atributy:

- **expires** – doba trvání cookie,
- **domain** – doména, pro kterou cookie platí,
- **path** – URL pro kterou cookie platí,
- **secure** – je-li nastaven tento atribut, pak bude cookie odesláno pouze v HTTPS požadavcích,
- **HttpOnly** – zabraňuje přistupovat k hodnotě přes skriptovací jazyk JavaScript.

(Stuttard, a další, 2011 str. 47)

3.6 Client-side technologie

Client-side je souhrnný název pro technologie běžící na straně klienta, tedy v prohlížeči. Často se také používá pojem Front-End technologie. (Wodehouse, 2015)

3.6.1 HTML

HyperText Markup Language neboli HTML je základní technologií používanou pro vytváření webového rozhraní. Struktura dokumentu se vytváří z tagů, resp. elementů, přičemž každý z nich plní určitou funkci. Z hlediska bezpečnosti webových aplikací jsou nejdůležitější elementy, které po akci uživatele, např. kliknutí, vyvolají komunikaci mezi klientem a serverem. Do této skupiny tagů patří odkaz a formulář. (Stuttard, a další, 2011 str. 58)

V roce 2008 vyšla nová verze HTML5, která s sebou přinesla nové tagy, atributy a funkce. Přibyly například nové možnosti ukládání uživatelských dat na straně klienta a tím vznikl nový druh útoku nazývaný Client-side SQL Injection. Dále došlo ke změně způsobu zpracování Ajax požadavků, které nově umožňuje obousměrnou interakci mezi doménami, a tím se otevřela další možnost útoku. (Stuttard, a další, 2011 stránky 64,65) (Lubbers, a další, 2011 stránky 22,25,29-32)

3.6.2 JavaScript

JavaScript je skriptovací programovací jazyk, který se spouští v prohlížeči po stažení HTML dokumentu ze serveru. Používá se k vylepšení uživatelského rozhraní způsoby, které nelze provést pomocí HTML. Jedná se například o dynamickou změnu uživatelského rozhraní vyvolanou akcí uživatele, např. kliknutím. K provedení této změny není potřeba odeslat požadavek na webový server. Díky tomu dochází k úspoře času potřebného k navázání spojení se serverem a počkání na jeho odpověď. Webová aplikace se tak stává rychlejší a interaktivní. (Suehring, 2008 stránky 19,20)

Mezi další časté důvody použití JavaScriptu patří ověření dat zadaných ve formuláři ještě předtím, než jsou odeslána na server. Dále pak k ovládání prohlížeče pomocí DOM (Document Object Model), který je abstraktní reprezentací HTML dokumentu. DOM umožňuje manipulovat s jednotlivými HTML elementy, cookies, URL adresou a umožňuje spouštět akce jako je odeslání formuláře, stisknutí klávesy nebo přesměrování stránky. (Stuttard, a další, 2011 stránky 61,62)

V dnešní době je JavaScript nedílnou součástí webových aplikací a je jednou z nejpoužívanějších technologií vůbec. Tento fakt potvrzuje i existence nespočet knihoven, které usnadňují programování v JavaScriptu. Mezi nejpoužívanější knihovny patří jQuery, React, Angular, Vue aj. (Žára, 2015 str. 12) (JavaScripting.com, 2017)

3.6.3 Ajax

Asynchronous JavaScript and XML neboli Ajax je technologie, pomocí které lze provádět asynchronní HTTP požadavky na server. Požadavky se provádí na pozadí a uživatele o nich nemusí ani vědět. Tuto technologii umožňuje objekt v JavaScriptu nazývaný XMLHttpRequest, který přijímá a odesílá požadavky na server v XML nebo formátu JSON. (Stuttard, a další, 2011 str. 62)(Zakas, a další, 2007 str. 15) (Suehring, 2008 str. 263)

XML (Extensible Markup Language) je značkovací jazyk, který stejně jako HTML používá elementy pro vytvoření struktury dokumentu. Jednotlivé elementy jsou uživatelsky definované, a to je důvodem, proč se XML často používá pro výměnu dat. (Suehring, 2008 str. 263)

JSON (JavaScript Object Notation) je jednodušším a efektivnějším formátem, který vychází ze syntaxe JavaScriptu. Není nutné tedy data převádět na objekty JavaScriptu jako je tomu v případě XML. Pro převedení dat do objektů se v moderních prohlížečích používá funkce *JSON.parse* a ve starších prohlížečích se používá funkce *eval*. Funkce *eval* spustí řetězec v parametru jako příkaz, a proto je nutné z bezpečnostních důvodů nejdříve tento řetězec ošetřit. (Stuttard, a další, 2011 str. 280) (Zakas, a další, 2007 stránky 237,240,241) (Vrána, 2012 str. 335)

3.6.4 Same-Origin Policy

Same Origin Policy je politika implementovaná v prohlížečích, která zamezuje přístup k datům nebo obsahu webových stránek z jiné domény. Politika dále kontroluje subdoménu, protokol a port, na kterém komunikace probíhá. Kromě Same Origin Policy existují také bezpečnostní politiky zaměřené na objekt XMLHttpRequest, technologii Silverlite, Gears nebo na objekty Javy. (Kümmel, 2011 stránky 42, 43)

3.7 Server-side technologie

Technologiím, které běží na serveru se kromě Server-side říká také Back-End technologie. Využívají se pro generování dynamického obsahu, který je součástí většiny dnešních webových aplikací. (Stuttard, a další, 2011 str. 51) (Wodehouse, 2015)

Dynamický obsah je výsledkem proběhnutého kódu, resp. programu. Takový program může přijímat vstupní proměnné, na základě kterých lze vygenerovat specifický obsah. Díky tomu můžeme například každému uživateli zobrazit unikátní obsah. (Stuttard, a další, 2011 stránky 51,52)

3.7.1 PHP

Programovací jazyk PHP vychází z jazyka C a jeho jádro tvoří Zend Engine, který rozebírá a spouští zdrojový kód skriptů. PHP se propojuje s webovým serverem třemi způsoby. Prvním a nejpomalejším je CGI, kdy každý skript spustí jeden interpret PHP. Druhým způsobem je využití modulu webového serveru. Posledním způsobem je vylepšené CGI, tzv. FastCGI, kdy interpret PHP běží na pozadí a předávají se mu skripty pro zpracování. (Vrána, 2012 stránky 47,439)

PHP obsahuje tzv. session (relace), které řeší problém bezstavovosti HTTP a umožňují tak uchovat data skrze více požadavků. Relace je nejčastěji používána právě pro zapamatování přihlášeného uživatele. K zapamatování slouží session proměnné, které jsou na serveru uloženy do souboru. V cookies je pak uložen otisek (hash) session ID, které slouží k identifikaci právě běžící relace. Každá relace má omezenou dobu trvání. (W3Schools, 2017) (Vrána, 2012 stránky 65, 177, 411)

PHP je častou volbou při tvorbě webových aplikací, díky jednoduchosti jeho použití. Způsob, jakým bylo PHP navrženo a jeho základní nastavení, vedlo ke snadnějšímu vytvoření bezpečnostních děr ve zdrojovém kódu webové aplikace. Konfigurace, které způsobovaly problémy byly s novými verzemi PHP odstraňovány. Nejnovější verze PHP 7, dle statistik webu (CVE Details, 2017), obsahuje aktuálně 78 bezpečnostních děr. (Stuttard, a další, 2011 stránky 55,732)

3.7.2 SQL

Structured Query Language (SQL) je jazyk, který se používá k manipulaci dat v relačních databázích Oracle, MS-SQL, MySQL, PostgreSQL aj. V relačních databázích jsou data

uložená v tabulkách, kde každá tabulka má sloupce, které reprezentují určitou hodnotu a řádky reprezentující záznamy. (Stuttard, a další, 2011 str. 55)

Webové aplikace, které pracují s databází využívají SQL dotazy jejichž součástí jsou často hodnoty vložené uživatelem aplikace. Například po odeslání formuláře pro úpravu kontaktních informací uživatele, se tyto údaje pomocí SQL dotazu odešlou na server. Pokud by útočník záměrně odeslal škodlivý údaj, mohl by pozměnit nebo přečíst citlivé údaje v databázi. (Stuttard, a další, 2011 str. 56)

3.8 Penetrační testování

Penetrační testování je proces, jehož cílem je nalézt a následně odstranit slabé stránky zabezpečení webových aplikací. Útočníci využívají penetrační testování k nalezení nejméně jedné bezpečnostní díry, zatímco penetrační testeři se snaží najít všechny zranitelnosti aplikace, které se následně opraví. (Caddy, 2011 stránky 920, 921) (Vrána, 2012 str. 375)

Předpokladem pro penetrační testování je fakt, že vybraný cíl je tvořen z hardwaru, softwaru a lidských zdrojů. Díky tomu je nutné kromě hledání bezpečnostních děr v aplikaci analyzovat zranitelnost lidí, např. pomocí sociálního inženýrství. (Caddy, 2011 stránky 920, 921)

Penetrační testování se podle (Kim, 2015 str. 21) a (Selecký, 2012 stránky 7, 8, 12-14) dělí na externí skenování, interní skenování a skenování webové aplikace. Samotné skenování probíhá ve 4 fázích – určení cíle a rozsahu, sběr dat, skenování a exploatace a nakonec report. (Caddy, 2011 stránky 920, 921) rozděluje penetrační testování na pasivní a aktivní odposlech, který je popsaný dále v této kapitole. (Stuttard, a další, 2011 str. 702) dělí penetrační testování na Black-box (černá skříňka) a White-box (bílá skříňka). Při Black-box testování není k dispozici zdrojový kód a tester neví, jak systém funguje a jak vypadá jeho struktura. Naopak u White-box testování je zdrojový kód k dispozici.

3.8.1 Pasivní odposlech

Při pasivním odposlechu neboli pozorování se snaží tester nebo útočník získat co nejvíce informací o cíli, sítích, klientech a dalších souvisejících věcech, aniž by nepříznivě ovlivnil provoz aplikace. Často se k takovému průzkumu využívají falešné účty na sociálních sítích jako je LinkedIn, Facebook, Twitter atd. (Kim, 2015 str. 26) (Caddy, 2011 stránky 920, 921)

Kromě nástrojů jako Discover Scripts, Recon-ng, Spiderfoot, Wordhound atd. lze potřebné informace o testovaném cíli zjistit pomocí vyhledávačů. Například vyhledávač Google umožnuje ve vyhledávání používat speciální operátory, které pomáhají dostat se lépe k požadovaným informacím. Obdobně lze využít proměnné v URL Googlu. Díky tomu je možné dohledat informace jako jsou názvy domén, které náš cíl využívá, podrobnosti o zařízeních na síti a další užitečné informace. (Kim, 2015 stránky 26-34) (Long, 2005 stránky 39, 58, 159-179)

3.8.2 Aktivní odposlech

Aktivní odposlech je proces, při kterém je snahou identifikovat jednotlivé systémy a služby cílového prostředí a najít v nich bezpečnostní nedostatky (Caddy, 2011 stránky 920, 921). (Kim, 2015 str. 45) uvádí několik vhodných nástrojů pro aktivní odposlech. Jedná se například o Masscan, Sparta, HTTP Screenshot, Nexpose, Nessus, OpenVAS, Nmap atd.

3.8.3 Interní a externí skenování sítě

Interní skenování se snaží najít vnitřní hrozby, které jsou reprezentovány útoky ze samotné napadané sítě. Nejčastějšími aktéry bývají vlastní zaměstnanci. Naopak předmětem externího skenování jsou testy vůči hrozbám z vnější strany sítě. Jedná se především o útoky ze sítě Internet. (Selecký, 2012 str. 39)

3.8.4 Skenování aplikace

Skenování, též mapování aplikace, začíná prozkoumáním obsahu aplikace a pochopením jejího chování. Následně je důležitá podrobná analýza aplikace, kde je nutné porozumět jednotlivým procesům, bezpečnostním mechanismům a zjistit použité technologie na straně klienta i serveru. Dodržením tohoto postupu je snadnější nalézt případné nedostatky v zabezpečení aplikace. (Stuttard, a další, 2011 str. 73)

Získání obsahu a pochopení chování aplikace

Prozkoumání obsahu aplikace a jejího chování je možné provést manuálním procházením stránek přes prohlížeč nebo využitím některého z nástrojů jako je například Burp Suite, WebScarab, Zed Attack Proxy, Nikto, w3af nebo CAT. Nástroje sice nejsou schopny procházet odkazy generované JavaScriptem nebo odesílat složitější formuláře, ovšem značně urychlují skenování aplikace. V některých případech může ale dojít k nepřetržitému běhu programu důsledkem mechanismu, který při každém požadavku přidá do URL náhodně

vygenerovaný parametr. Ideální je tedy použít nástroje společně s manuálním procházením webové aplikace. (Stuttard, a další, 2011 stránky 74-76) (Kim, 2015 str. 59)

Webové aplikace jsou složeny z obsahu, resp. souborů a funkcí, které nejsou přímo dohledatelné. Jedná se například o funkce, které byly implementovány při testování aplikace a nebyly odstraněny. Dále to může být skrytý obsah, který je dostupný pouze přihlášeným uživatelům pod určitými právy. Pro útočníka jsou pak důležité především soubory, které nejsou přímo dosažitelné z webového rozhraní aplikace a obsahují citlivé údaje jako jsou například přihlašovací údaje do databáze, informace o uživatelských účtech, záznamy o chybách v aplikaci atd. (Stuttard, a další, 2011 stránky 80, 81)

V některých případech se stává, že je obsah skrytý, avšak v minulosti tomu tak být nemuselo. Vyhledávací stroje jako je Google, Yahoo nebo MSN a webový archiv WayBack Machine ukládají kopie obsahu do své paměti. Pomocí rozšířených dotazů o speciální operátory ve vyhledávačích nebo kopie aplikace ve webovém archivu, lze dohledat užitečný obsah, který byl dostupný v dřívější verzi webové aplikace. Dále je také důležitá historie webových stránek, kde vývojáři například popisují použité technologie při tvorbě aplikace, na kterou cílíme útok. (Stuttard, a další, 2011 stránky 89-91)

Analýza bezpečnosti aplikace

Prozkoumání obsahu je první fází skenování aplikace. Druhou fází je analýza aplikace z hlediska její funkčnosti, chování a použitých technologií. Tato fáze nakonec vede k identifikaci všech proveditelných útoků proti cílové aplikaci. Následující výčet obsahuje základní oblasti, které jsou během této fáze analyzovány. (Stuttard, a další, 2011 str. 97)

- Základní i okrajová funkčnost aplikace, např. zpracování chyb nebo přesměrování.
- Bezpečnostní mechanismus, tj. řízení přístupu, autentizace nebo práce se session.
- Všechna místa, kde aplikace zpracovává uživatelské vstupy – parametry v URL, data odeslaná přes metodu POST, cookie a hlavičky.
- Technologie použité na straně klienta, tj. formuláře a komponenty jako Java applety, ovládací prvky Active-X nebo Flash.
- Technologie použité na straně serveru, tj. použití SSL, webový server, použitý programovací jazyk, komunikace s databází, e-mailový systém a využité back-endové komponenty třetích stran.

(Stuttard, a další, 2011 stránky 97, 98)

Pro analýzu bezpečnosti aplikace lze využít nástroje jako je Burb Suite, AppScan, Acunetix, w3af, Paros, Hailstorm nebo NetSpark. Prohlížeč Internet Explorer, Firefox a Chrome obsahují nástroj, pomocí kterého lze procházet zdrojový kód stránky, zobrazit a upravit hlavičky požadavků i odpovědí, zobrazit běžící skripty na pozadí a mnoho dalších funkcí, které jsou užitečné při hledání zranitelných míst v aplikaci. (Stuttard, a další, 2011 stránky 748-750, 781, 782) (Kim, 2015 str. 36)

3.9 Obcházení ovládacích prvků na straně klienta

Předávání dat z klienta na server lze provést pomocí prvků na straně klienta nebo přes session. Použití session pro všechna data je nežádoucí, protože zpracování těchto dat by zabralo moc času a aplikace by nebyla dostatečně výkonná. Zároveň by byl vývoj aplikace složitější, protože by bylo obtížné řešit případy, kdy webová aplikace komunikuje s více něž jedním serverem. Ze zmíněných důvodů jsou data na server přenášena přes klientské prvky jako jsou formuláře, cookies, URL parametry, HTTP hlavičky nebo Client-side komponenty. (Stuttard, a další, 2011 stránky 9, 118)

Odesílání dat přes klienta není zcela bezpečné, protože uživatelé mohou na server odeslat libovolný vstup, který může vést k získání neoprávněného přístupu k datům a funkcím aplikace. Kvůli tomu jsou na straně klienta implementována opatření, která kontrolují odesílaná data na server. Stejná opatření jsou nutná implementovat také na straně serveru, protože jakoukoliv kontrolu na straně klienta lze snadno obejít a zároveň může během přenosu dat dojít k jejich modifikaci. (Stuttard, a další, 2011 stránky 9, 117) (Long, 2005 str. 429)

3.9.1 Formuláře

HTML formuláře jsou nejjednodušším způsobem, jak uživatelský vstup odeslat na server. Zároveň umožňují nastavit omezení a provádět validaci dat. Tuto ochranu je snadné obejít například odstraněním daného omezení nebo změnou hodnoty v HTML. Možným omezením je například atribut *max*, který udává maximální číslo ve vstupním poli. V případě, že by vstupní pole, tzv. input sloužil k zadání množství produktů ke koupi, jeho odstranění by mohlo vést k prodeji většího množství produktů, než je ve skutečnosti na skladě. (Stuttard, a další, 2011 stránky 127, 128)

Vestavěný validační mechanismus ve formulářích obsahuje pouze základní a nelze ho použít k validaci všech druhů vstupů. Například u registrace uživatelů jsou vstupy jako e-mail,

telefonní číslo, jméno atd. Pro takové formuláře se používá validace pomocí skriptu napsaném v JavaScriptu. Skript se spouští při odeslání formuláře a při úspěšné validaci jsou data odeslána na server. V opačném případě je uživatel vyzván k opravě zadaného údaje. JavaScript lze ovšem upravit nebo v prohlížeči vypnout a vyhnout se tak validačnímu procesu. (Stuttard, a další, 2011 stránky 129, 130) (Suehring, 2008 str. 25)

HTTP formuláře mají několik typů vstupních polí. Jedním z nich je například typ *hidden*, který prohlížeč informuje o tom, aby input na obrazovce nezobrazoval. Zadaná hodnota je přesto odeslána na server. Tento input se využívá k odesílání dat, která nemají být uživatelem modifikována, ale přesto jsou pro následující zpracování důležitá. V některých případech chceme mít naopak input viditelný na obrazovce, ale neměnitelný. K tomuto účelu slouží atribut *disabled*, který odstraňuje možnost editace a zároveň zamezuje odeslání dané hodnoty na server. Změna typu, resp. atributu inputu, by při absenci ošetření na straně serveru vedla ke zpracování nežádoucí hodnoty. (Stuttard, a další, 2011 stránky 118, 119, 131, 132) (Long, 2005 str. 429)

3.9.2 HTTP Cookies

Dalším běžným způsobem, jak dostat uživatelská data na server je použít cookies. Obdobně jako u inputů typu hidden není ani cookies zobrazeno na obrazovce a uživatelé nemohou hodnoty v něm přímo upravovat. Cookies se často používá k uložení informací o uživateli, jako například jeho jméno nebo e-mailová adresa. Citlivé údaje však není vhodné do cookies ukládat, protože uživatel může obsah cookies zobrazit s využitím proxy serveru nebo prostřednictvím rozšíření v prohlížeči. Zároveň není vhodné do cookies ukládat data potřebná pro správný chod aplikace, protože uživatel si může podporu cookies vypnout. (Stuttard, a další, 2011 str. 121) (Vrána, 2012 str. 174)

3.9.3 URL parametry

Častým způsobem je přenos dat přes URL parametry, které mohou být uživatelem velice snadno upraveny bez použití speciálního nástroje. Aplikace obsahují URL s parametry v HTML kódu, u kterých se ale očekává, že je běžní uživatelé nemohou zobrazit a měnit. Jedná se například o parametry v URL, které specifikují, jaký obrázek nebo obsah webu se má načíst. V tomto případě se dají parametry modifikovat stejně jako u formulářů nebo cookies. (Stuttard, a další, 2011 stránky 121, 122) (gotowebs, 2017)

3.9.4 Hlavička Referer

Stejně jako obsah dat je pro aplikaci důležité i pořadí v jakém jsou na server data odesílána. Pokud by například nebyl pomocí hlavičky *referer* ověřen zdroj požadavku, mohl by útočník přeskočit některý krok autorizace ke konkrétní akci nebo provést požadavek ze svého serveru. Tuto kontrolu lze snadno obejít podstrčením požadované hodnoty do referer hlavičky pomocí proxy serveru. Prohlížeč hlavičku referer nemusí ani přenášet, takže se nedá na její přítomnost spolehnout. (Stuttard, a další, 2011 stránky 9, 122) (Vrána, 2012 str. 363)

3.10 Útoky na autentizaci

Autentizace je základním bezpečnostním opatřením proti neoprávněnému přístupu. Útočník, který prolomí toto zabezpečení získá plnou kontrolu nad funkcemi aplikace a neomezený přístup k jejím datům. Bez robustní autentizace, na kterou se lze spoléhat, nemůže být žádný z ostatních bezpečnostních mechanismů účinný. (Stuttard, a další, 2011 str. 159)

Způsobů, jak ověřit identitu uživatele je několik. Nejběžnějším mechanismem je HTML formulář, do kterého se zapisuje uživatelské jméno a heslo. Ve webových aplikacích, pro které je bezpečnost důležitá, jako například internetové bankovnictví, je nutné do formuláře zadat další údaje jako je PIN nebo jednorázové heslo zaslané na e-mail nebo SMS. Některé aplikace mají v sobě implementovaný mechanismus na ověřování uživatelů na základě klientských SSL certifikátů. Dalším běžným způsobem je pak autentizace pomocí HTTP hlavičky a službou třetích stran. (Stuttard, a další, 2011 stránky 160, 161) (Vrána, 2012 str. 377) (Scambray, a další, 2002 str. 143)

3.10.1 Hrubá síla

Spousta webových aplikaci má minimální požadavky na složitost hesla a umožňuje neomezený počet pokusů pro jeho zadání. S dnešním výkonem počítačů a internetového připojení lze provést několik tisíc pokusů o přihlášení během jedné minuty. S využitím vhodného nástroje, jako je například Burb Intruder, by tak v tomto případě uhodnutí hesla, tzv. hrubou silou, trvalo jen pár minut. Aplikace, které kontrolují počet pokusů používají cookies nebo session k uchování aktuální hodnoty. Odstraněním session ID nebo samotné hodnoty z cookies dojde k získání dalších pokusů o přihlášení. Ideální variantou je tedy ukládat počet pokusů do databáze. (Stuttard, a další, 2011 stránky 161-164) (Vrána, 2012 str. 383)

Většina aplikací účet zamkne při několika neúspěšných pokusech. V některých případech se ale stává, že aplikace stále vrací chybovou hlášku, která indikuje špatné heslo. Útočník je potom schopný heslo dál hádat. (Scambray, a další, 2001 stránky 125, 126) dále dodávají, že chybové hlášky, které jsou navíc odlišné pro špatně zadané heslo a uživatelské jméno, usnadňují útočníkovi práci. (Vrána, 2012 str. 383) popisuje, že jednotná zpráva pro ověření totožnosti není uživatelsky přívětivá a útočník se o existenci uživatelského jména v systému může dozvědět při registraci.

3.10.2 Přenos přihlašovacích údajů

Kromě použití hrubé síly je možné heslo a uživatelské jméno získat během přenosu přístupových údajů na server. Nejběžnějším útokem je tzv. Man-in-the-Middle (MitM), kdy komunikace mezi klientem a serverem probíhá přes útočníka. Útočník je navíc schopný udržovat dva samostatné klíče a odposlouchávat šifrovanou komunikaci jako je tomu v případě HTTPS. Kromě toho dokáže data na trase měnit. (Stuttard, a další, 2011 str. 171) (Erikson, 2008 stránky 406, 407)

V první fázi MitM útoku se provádí tzv. otrávení protokolu ARP (Address Resolution Protocol), který slouží k získání MAC adresy na základě IP adresy. Důsledkem otrávení veškerá komunikace klientského zařízení s routerem probíhá přes počítač útočníka. V další fázi může útočník získat přímo přihlašovací údaje nebo token relace přihlášeného uživatele. Token následně importuje do svého prohlížeče a získá veškerá jeho oprávnění. Pro MitM útok existuje řada nástrojů jako je např. mitm-ssh, Ettercap, BDFProxy nebo Cain and Abel. (Kim, 2015 stránky 144-151)

3.10.3 Uložení přihlašovacích údajů

V databázi se místo hesla ukládá jen jeho otisk, aby útočník nebo i zaměstnanec heslo nezneužil. K vytvoření otisku se nejčastěji používá hashovací funkce *MD5* a *SHA-1*. Útočník, který získá otisk může jeho hodnotu porovnat s databází ostatních otisků a zjistit tak čistý text hesla. Kvůli tomu se k heslům přidává ještě tzv. salt (sůl), což je řetězec náhodných znaků generovaných aplikací. Dále je lepší používat bezpečnější hashovací funkci jako je *SHA-256* nebo *SHA-3*. (Stuttard, a další, 2011 str. 190) (Vrána, 2012 stránky 380, 382)

3.10.4 Funkce pro změnu hesla

Dobře navržený autentizační mechanismus aplikace obsahuje funkci, která periodicky nutí uživatele změnit heslo, a tím zmírňuje hrozbu jeho uhodnutí a zneužití. Současně funkce slouží jako jednodušší a rychlejší možnost změnit heslo uživatelem v případě podezření na jeho vyzrazení. Přestože aplikace tuto funkci obsahují, mohou být špatně implementované. V některých případech se lze setkat s tím, že funkce pro změnu hesla je přístupná bez předešlé autentizace a poskytuje chybové hlášky indikující, zda je uživatelské jméno nebo současné heslo správné či nikoliv. Společně s neomezeným počtem pokusů na zadání správného současného hesla může útočník tuto funkci využít k uhodnutí hesla hrubou silou. (Stuttard, a další, 2011 stránky 171, 172) (Vrána, 2012 str. 385)

3.10.5 Funkce pro zapomenutí hesla

Funkce pro zapomenuté heslo je řešena například skrze nápovědu, kterou uživatel zadává během registrace. Pro útočníka je pak snadnější heslo uhodnout. Dalším způsobem je kontrolní otázka po jejímž správném zodpovězení je uživatel přihlášen. Často se jedná o otázky, u kterých jsou odpovědi veřejně známé nebo dohledatelné. U složitějších otázek je množství potenciálních odpovědí pořád menší než soubor možných hesel. Nejčastějším a nejbezpečnějším způsobem je odeslání unikátní, neuhodnutelné a časově omezené URL na e-mailovou adresu uživatele, na které si může vytvořit nové heslo. Vývojáři se při implementaci tohoto řešení dopouští často závažné chyby, kterou je možnost zadat e-mail, na který se má odkaz s URL odeslat. Ověření, zda e-mail patří danému uživateli není přitom provedeno. Útočník tak může zadat svůj vlastní e-mail. (Stuttard, a další, 2011 stránky 173-175) (Vrána, 2012 str. 384)

3.10.6 Funkce pro trvalé přihlášení

S aktivovaným trvalým přihlášením nemusí uživatelé zadávat přihlašovací údaje pokaždé, když používají aplikaci ze stejného počítače. Nejčastějším způsobem implementace je využití cookies, kdy aplikace na základě jedinečného identifikátoru pozná, o jakého uživatele se jedná a přeskočí přihlášení. Přestože je hodnota v cookies šifrovaná a těžko uhodnutelná, může být stále dosažitelná pomocí techniky Cross-Site Scripting nebo útočníkem, který má přístup k počítači uživatele. (Stuttard, a další, 2011 stránky 176, 177)

3.10.7 Zabezpečení autentizace

Vzhledem k velkému počtu možných zranitelností v autentizaci a potenciálně složité obraně proti útokům vývojáři přijímají určité hrozby a soustředí se především na ty vážnější. Při rozhodování hraje roli možnost zabezpečení vzhledem k funkčnosti aplikace, stupeň tolerance uživatelů vůči odlišným způsobům autentizace, náklady na podporu méně uživatelsky přívětivého systému a vztah finančních nákladů na zabezpečení vůči příjmům, které aplikace generuje nebo hodnotě dat, které aplikace chrání. (Stuttard, a další, 2011 str. 191)

V následující části bude výčet doporučení a způsobů, jak se bránit útokům na autentizační mechanismus.

Silné přihlašovací údaje

- Heslo by mělo splňovat minimální délku (nejčastěji 8 znaků) a mělo by se skládat z velkých i malých písmen, číslic a typografických znaků.
- Heslo by nemělo odpovídat žádném z běžných hesel nebo odpovídat heslu dříve zvolenému či již existujícímu.
- Uživatelské jméno by mělo být unikátní.
- Automaticky generované přihlašovací údaje by neměly být předvídatelné.
- U formulářového pole pro zadání hesla by mělo být vypnuto automatické doplňování dříve zadaných údajů prohlížečem.

(Stuttard, a další, 2011 str. 192) (Vrána, 2012 str. 386)

Zpracování přihlašovacích údajů

- Přihlašovací údaje by měly být vytvořeny, uchovány a přenášeny způsobem, který nevede k jejich neoprávněnému odhalení.
- Komunikace by měla být chráněna kryptografickou technologií jako je SSL, která je součástí protokolu HTTPS.
- Údaje by měly být odesílány pouze přes HTTP metodu *POST*.
- V databázi by měl být uložen pouze otisk hesla, který byl získán použitím silné hashovací funkce, např. *SHA-256* nebo *SHA-3*.
- Heslo by mělo obsahovat, tzv. sůl.

(Vrána, 2012 stránky 380, 382) (Stuttard, a další, 2011 stránky 192, 193)

Funkce pracující s přihlašovacími údaji

- U funkce pro trvalé přihlášení by neměly být přihlašovací údaje nikdy uloženy v cookies prohlížeče v nešifrované formě a měl by být kladen důraz na eliminaci Cross-Site Scripting útoků.
- Funkce pro změnu hesla by měla být vždy implementována, kvůli pravidelné vynucené, resp. vyžádané změně hesla aplikací, resp. uživatelem. Zároveň by měla být přístupná pouze přihlášenému uživateli, který by měl být vždy vyzván k zadání hesla současného.
- Uživatelé by při změně hesla měli nové heslo vždy potvrdit opětovným zadáním.
- V případě zadání chybných přihlašovacích údajů by se uživateli měla zobrazit jednotná hláška.
- Uživatelé by měli být informováni, že došlo ke změně hesla.
- Při zapomenutém hesle se uživateli pošle na předem domluvený komunikační kanál (obvykle e-mail) URL s náhodně vygenerovaným řetězcem, který má omezenou dobu trvání. Tento způsob je vhodné kombinovat s kontrolními otázkami.

(Stuttard, a další, 2011 stránky 193, 199, 200) (Vrána, 2012 stránky 383, 384, 385)

Ochrana proti hrubé síle

- Uživatelský účet by měl být zablokován po několika neúspěšných pokusech o přihlášení a obnoven až po několika minutách, např. 30 minut. Informace o počtu zbývajících pokusů nebo minut by neměla být zobrazena.
- Chybové zprávy by měly být, co nejméně podrobné, aby podle nich útočník nemohl optimalizovat proces hádání hesla.
- Každá stránka, resp. formulář, který může být cílem útoku hrubou silou, by měl obsahovat CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), která slouží k identifikaci robota či nástroje, který útok provádí. Jedná se o obrázek s různě deformovaným textem, který musí uživatel rozluštit. Deformace musí být navíc pokročilá, aby nebyl text programově rozluštěn pomocí OCR (Optical Character Recognition).

(Stuttard, a další, 2011) (Vrána, 2012 stránky 373, 383) (Scambray, a další, 2001 stránky 126, 132)

Monitorování

- Aplikace by měla uchovávat veškeré informace týkající se autentizace, jako je přihlášení, odhlášení, změna hesla nebo jeho obnovení do protokolů na serveru. Protokoly by měly být silně chráněny proti neoprávněnému přístupu.
- Administrátor aplikace by měl být upozorněn v případě zvláštních událostí, které by nastaly během autentizace některého uživatele.
- Uživatel by měl být e-mailem upozorněn o jakýchkoliv bezpečnostních událostech na jeho účtu a měl by mít k dispozici údaje o jeho posledním přihlášení a počtech neplatných pokusů o přihlášení.

(Stuttard, a další, 2011 str. 201)

3.11 Útoky na session

HTTP protokol je bezstavový a jednotlivé požadavky nejsou nijak provázány. Webové aplikace používají session tokeny, resp. identifikátory relace, které identifikují a sledují uživatele během procházení jednotlivých stránek webu. Session tokeny jsou obvykle uložené v cookies, ale mohou být předávány přes parametr v URL nebo ve skrytých polích uvnitř HTML formuláře. (Schema, 2012 str. 142) (Long, 2005 stránky 442, 443)

Standardní mechanismus pro řízení relace je náchylný na různé útoky. Útočník se primárně snaží unést relaci (anglicky Session Hijacking) a vydávat se tak za jiného uživatele, čímž získá přístup k jeho datům a funkcím. Dalším útokem je Session Fixation, který spočívá v podvrhnutí oběti útočníkova tokenu. Útočník následně počká, až se oběť přihlásí a začne se za uživatele vydávat. Posledním útokem v návaznosti na ukládání identifikátoru do cookies je Cross-Site Scripting útok, který je blíže popsán v kapitole 3.14. Větší aplikace, převážně komerční, proto implementují vlastní funkce pro správu relací, které jim umožňují větší kontrolu nad chováním session. U bezpečnostně kritických aplikací jako je internetové bankovnictví se lze často setkat se správou relace, která není založena na HTTP cookies. (Stuttard, a další, 2011 stránky 207, 243) (Vrána, 2012 stránky 365, 367) (Scambray, a další, 2002 str. 155)

3.11.1 Slabiny v generování tokenu

Podle (Kim, 2015 str. 69) je při generování tokenu nutné zohlednit dvě hlavní věci – nesmí být uhodnutelný a musí řádně sledovat uživatele. (Stuttard, a další, 2011 str. 210) poukazují

na fakt, že v aplikaci existuje řada funkcí, kde bezpečnost aplikace závisí na nepředvídatelnosti tokenů, které sama generuje. Jedná se například o token, který je zaslán uživateli při zapomenutí hesla nebo je využíván funkcí pro trvalé přihlášení. Dále se tokeny využívají ve skrytých formulářových polích jako ochrana proti útoku Cross-Site Request Forgery nebo k udělení jednorázového přístupu k chráněnému obsahu.

Tokeny mohou být generovány několika způsoby. Jedním z nich je transformace komponent, resp. uživatelských údajů jako je uživatelské jméno, křestní jméno, příjmení, e-mail, uživatelská role atd. do řetězce, který je zakódován pomocí XOR, Base64 nebo hexadecimálním kódováním. Každý jednotlivý údaj může být přitom zakódován jiným způsobem. Aplikace, která přijme požadavek s tokenem zpracovává všechny nebo pouze některé komponenty. Útočník analýzou tokenu může kromě zvoleného kódování zjistit, která část tokenu je aplikací validována. Množství potenciálně validních tokenů je tak nižší. (Stuttard, a další, 2011 stránky 210 - 212)

Dalším způsobem vygenerování tokenu je pomocí funkce, která generuje sekvenci náhodných znaků. Faktem je, že náhodné číslo nebo řetězec je generován určitým algoritmem a sekvence se jako náhodná pouze jeví. Útočník, který získá malý vzorek vygenerovaných znaků je schopen odhadnout tokeny, které budou validní. K těmto účelům lze využít například program Burp Intruder. Kvalitu náhodnosti sekvence lze testovat pomocí programu Burp Sequencer. (Stuttard, a další, 2011 stránky 213, 218) (Scambray, a další, 2001 str. 151) (Vrána, 2012 stránky 367, 368)

Některé aplikace šifrují tokeny pomocí tajného klíče ještě předtím, než jsou vydány uživatelům. Tento token se nejčastěji skládá z uživatelských údajů. V závislosti na použitém šifrovacím algoritmu a způsobu jakým jsou tokeny v aplikaci zpracovány, lze v některých situacích manipulovat s obsahem tokenu bez předešlého dešifrování. Registrací několika uživatelů je útočník schopný zjistit, která část tokenu odpovídá uživatelskému jménu nebo roli a odhadnout použitou blokovou šifru. Dosazením části vlastního tokenu nebo manipulací bloků šifrovaného tokenu může útočník vystupovat pod jiným uživatelem nebo získat jiná oprávnění. Úspěch útoku závisí na způsobů, jakým aplikace zpracovává a ověřuje token. (Stuttard, a další, 2011 stránky 223 -232)

3.11.2 Slabiny v manipulaci s tokeny

Nezáleží na tom, jakým způsobem a jak efektivně aplikace generuje tokeny, když s nimi pečlivě nezachází. Příkladem nesprávné manipulace s tokeny je jejich odesílání přes nešifrované HTTP spojení. Přestože aplikace používá HTTPS pro každou stránku, útočník může přimět uživatele provést požadavek přes HTTP, který odchytí a získá tak token uživatele. (Stuttard, a další, 2011 stránky 233-236)

Systémové logy, resp. soubory, do kterých se zaznamenávají činnosti programů, jsou místem kam se tokeny zapisují. Mnoho aplikací administrátorům a personálu podpory umožňuje monitorovat a kontrolovat běh aplikace a nahlížet do logů. Rozsah potenciálních útočníků je tak daleko širší. (Stuttard, a další, 2011 str. 237)

Další slabinou v aplikaci je několik validních tokenů pro jednoho uživatele nebo statický token, který se uživateli přiřadí při každém přihlášení. Každý token by měl mít omezenou dobu trvání a uživatel by měl mít možnost relaci kdykoliv ukončit prostřednictvím funkce pro odhlášení. V opačném případě může útočník použít token, který uživatel použil dříve nebo získat aktuální a nebýt při tom časově omezen. (Stuttard, a další, 2011 stránky 240 - 243)

Poslední slabinou v manipulaci s tokeny je defaultní sdílení cookies v prohlížeči s více aplikacemi, které běží pod stejnou doménou. Každá taková aplikace má většinou vlastní subdoménu a může generovat tokeny stejným algoritmem. Zároveň může mít rozdílné bezpečnostní opatření jehož slabinu může útočník využít a získat identifikátor sezení, který bude validní pro ostatní aplikace běžící na stejné doméně. Kromě subdomény mohou běžet další aplikace také v jiném adresáři. Nevhodně nastavený adresář, pro který má cookies platit opět vede ke zranitelnosti aplikace. (Stuttard, a další, 2011 stránky 245 - 247)

3.11.3 Zabezpečení řízení session

Pro ochranu řízení relací proti útokům musí webové aplikace přijmout dvě bezpečnostní opatření, mezi která spadá generování silných tokenů a jejich ochrana po celou dobu životního cyklu, tj. od vytvoření do jejich likvidace. (Stuttard, a další, 2011 str. 248)

Generování silných tokenů

- Token by měl obsahovat pouze identifikátor, který server využije k dohledání relevantního objektu (uživatele) a neměl by tedy obsahovat žádné smysluplné informace o uživateli.
- K vygenerování náhodného řetězce by měla být využita funkce, která poskytuje nejlepší pseudonáhodnou sekvenci znaků.
- Vhodné je k řetězci připojit tajné slovo, které zná pouze server a informaci jako je IP adresa, čas v milisekundách nebo User-Agent hlavičku, která zmírní nedostatky v použitém generátoru pseudonáhodných čísel.
- Celý řetězec je vhodné hashnout například pomocí funkce *SHA-256*.

(Stuttard, a další, 2011 stránky 248, 249) (Vrána, 2012 str. 368) (Scambray, a další, 2002 str. 155)

Ochrana tokenů

- Token by měl být přenášen pouze přes HTTPS a cookies hodnota by měla být označená jako *secure*.
- Cookies by mělo být nastaveno pro takovou doménu či adresář, který odpovídá přesně dané aplikaci a nedošlo tak k získání tokenu skrze nezabezpečenou aplikaci běžící na subdoméně či v podadresáři.
- Tokeny by nikdy neměly být přenášeny v URL, protože je pak snadné provést Session Fixation útok. Pokud má prohlížeč vypnutou podporu cookies, je lepší použít skryté formulářové pole a odesílat token skrze *POST* požadavek.
- Aplikace by měla poskytovat funkci pro odhlášení uživatele, která zlikviduje všechny relace držené uživatelem na serveru.
- Uživatel by měl být automaticky odhlášen po určité době nečinnosti, např. 10 minut.
- Při každém přihlášení by měl být uživateli vygenerován jiný token a všechny ostatní jemu přiřazené odstraněny.
- Pokud aplikace obsahuje administrativní a diagnostické funkce, které obsahují identifikátory sezení, měly by být robustně chráněny proti neoprávněnému přístupu. Ideálně by místo tokenu měly obsahovat pouze základní informace o uživateli.

- Ještě lepší kontrolu nad session lze dosáhnout generováním tokenu každým požadavkem, přičemž nový token je porovnán s hodnotou předešlého. V případě nesouladu je relace ukončena.

(Stuttard, a další, 2011 stránky 250 - 252) (Vrána, 2012 str. 367)

Monitorování

- Aplikace by měla monitorovat všechny požadavky, které obsahovaly neplatný token. IP adresa s velkým množstvím takových požadavků by měla být zablokována.
- Administrátor aplikace a uživatel by měli být informováni v případě zvláštních událostí týkajících se relace.

(Stuttard, a další, 2011 str. 253)

3.12 Útoky na autorizaci

Řízení přístupu, resp. autorizace je bezpečnostní technika, která reguluje, kdo může zobrazit nebo použít určité zdroje webové aplikace. Autorizace, díky předešlé autentizaci pomocí přihlašovacích údajů, ověřuje a schvaluje přístup odpovědným subjektům. Udělování přístupu ke zdrojům se provádí například na základě rolí jednotlivých uživatelů. Špatné řízení přístupu se řadí mezi nejčastěji se vyskytující zranitelnosti webových aplikací. (Stuttard, a další, 2011 str. 257) (Rouse, 2014)

3.12.1 Zranitelnosti

Existují tři hlavní bezpečnostní nedostatky, díky kterým je útočník schopný přistupovat k funkcím a zdrojům, ke kterým nemá oprávnění. Prvním nedostatkem je možnost uživatele spouštět funkce, které nepřísluší jeho roli. Druhým nedostatkem je možnost zobrazit a upravit data, která patří jinému uživateli. Posledním nedostatkem je možnost uživatele přeskočit některou fázi, resp. stav aplikace a získat tak přístup k cílovému zdroji nebo funkci. Jedná se například o přeskočení platby během objednávky na e-shopu nebo přeskočení některé ověřovací fáze. (Stuttard, a další, 2011 stránky 258, 259) (gotowebs, 2017)

V některých aplikacích mohou být důvěrné funkce a zdroje přístupné každému, kdo zná relevantní URL. Příkladem může být adresa *https://example.com/admin/*, kde uživatel může využívat administrátorské funkce. Tato URL většinou není pro uživatele s jinou rolí viditelná a nemusí být vždy, tak snadno uhodnutelná. Přesto může útočník zmíněné URL najít ve zdrojovém kódu aplikace, a to konkrétně ve skrytých formulářových polích či JavaScriptu.

Obdobným způsobem lze přistupovat ke zdrojům, které patří jinému uživateli. Příkladem je adresa *https://example.com/ViewDocument.php?docid=1280149120*, kde parametr *docid* značí identifikační číslo dokumentu. Uhodnutím čísla dokumentu jiného uživatele a nedostatečným zabezpečením, se tak může útočník neoprávněně dostat k cizím datům. Stejný problém nastává u získání URL pro přímé stažení daného souboru, např. *https://example.com/download/9780636628104.pdf*. Webový server statický soubor vrátí bez spuštění kódu aplikace a nelze tedy stažení zabránit. (Stuttard, a další, 2011 stránky 259 - 264)

Kromě URL může být úroveň přístupu přenášena skrze skrytá pole ve formuláři, hlavičky nebo cookies. Během každého požadavku aplikace tyto hodnoty přečte a rozhodne se, zda uživateli přístup povolí. Pro útočníka, který nezíská administrátorská práva, je těžké zjistit jaká a zda vůbec jsou data tímto způsobem přenášena. Příkladem může být udělení oprávnění na základě IP adresy či hlavičky *Referer*, kdy aplikace kontroluje, zda byl požadavek proveden z administrátorské stránky. (Stuttard, a další, 2011 stránky 265, 266) (Scambray, a další, 2002 str. 168)

Spousta aplikací spoléhá na řízení přístupu na základě uživatelských práv v souborovém systému webového serveru. Ovšem tato práva mohou být u některých důležitých souborů špatně nastavená a útočník se k nim může dostat. Dalším problémem může být ukládání webových stránek do mezipaměti, kdy útočník může využít sdílený počítač v knihovně, ve škole či dalších veřejných místech a dostat se neoprávněně k důvěrným informacím. (OWASP, 2010)

3.12.2 Útoky

Nejjednodušší a nejefektivnější metodou testování řízení přístupu aplikace je použití dvou uživatelských účtů ideálně s rozdílnými právy. Následně testovat, které zdroje a funkce jsou legálně dostupné jedním účtem a zároveň nelegálně druhým. Tam kde není potřeba lidská inteligence, lze využít automatické testování pomocí vhodného nástroje. V případech vícestupňového procesu, kdy uživatel musí provést několik požadavků ve správném pořadí je toto testování neefektivní. Řešením je testování každého jednotlivého procesu zvlášť. (gotowebs, 2017)

Pokud nelze v aplikaci vytvořit více uživatelů a útočník má k dispozici pouze jeden účet, pak je potřeba vynaložit více úsilí k nalezení bezpečnostní díry. Především je potřeba

otestovat veškeré parametry v URL, hodnoty v cookies a hlavičky. Zároveň je důležité prozkoumat zdrojový kód aplikace a zjistit, jestli se v něm nevyskytují odkazy na funkce či zdroje, které jsou v uživatelském rozhraní skryté. Může se jednat například o staré funkce, které nebyly smazány nebo nové funkce, které nebyly zatím publikované. Dále se útočník soustředí na hledání odkazů na přímé stažení dokumentů a snaží se pochopit, jakým způsobem je tvořen jejich název a získat tak přístup k souboru jiného uživatele. (Stuttard, a další, 2011 stránky 273, 274, 277)

3.12.3 Zabezpečení řízení přístupu

Autorizace je nejjednodušší oblastí bezpečnosti webových aplikací, které je potřeba porozumět a při jejíž implementaci je potřeba důkladně aplikovat danou metodologii. Součástí této metodologie je nejčastěji matice řízení přístupu, která definuje typy uživatelů a jejich povolené funkce. Bezpečnostní díra v autorizaci je většinou způsobena chybnými předpoklady o druzích požadavků, které uživatelé mohou vytvořit a zároveň nedostatečným testováním. (OWASP, 2010)

(Stuttard, a další, 2011 stránky 278 - 280) a (gotowebs, 2017) dále poskytují následující rady, které snižují riziko úspěšného útoku:

- Aplikace by měla uživateli udělovat právo ke zdroji nebo funkci pouze na základě hodnot v session a nikoliv např. v URL parametrech.
- Administrátorská práva by měla být přístupná pouze na základě správné IP adresy.
- Přímý přístup k souborům by měl být chráněn pomocí HTTP autentizace nebo jiné funkce aplikačního serveru.
- Autorizace by měla být prováděna na každé stránce, a tedy při každém požadavku.
- Aplikace by měla uchovávat záznamy o každém přístupu k důvěrným informacím.

3.13 Útoky na datové uložiště

Data jsou nejčastěji ukládána v SQL databázích, repositářích založených na XML a LDAP adresářích. Do těchto uložišť se také ukládají nastavení, která řídí logiku aplikace. Jedná se například o uživatelské účty, jejich práva nebo jiná nastavení aplikace. Kromě získání důvěrných informací může útočník modifikací správných dat získat kontrolu nad celou aplikací. (Stuttard, a další, 2011 str. 287)

Webové aplikace pro ukládání dat využívají databáze a pomocí skriptů se dotazují k datům, která zrovna potřebují. Programovací jazyk a způsob jakým jsou dotazy do databáze prováděny určuje, zda je aplikace náchylná na tzv. Code Injection (injektáž kódu). Code Injection je obecný termín pro typy útoků, které díky nedostatečné validaci vstupních dat modifikují kód aplikace a umožňují útočníkovi přístup k datovému uložišti. (OWASP, 2013) (Clarke, 2012 str. 17)

Code Injection se často používá u přihlašovacích formulářů, kde lze zadáním vhodného kódu do formulářového pole docílit toho, že aplikace udělí přístup do zabezpečené sekce neoprávněnému uživateli. Podle použité technologie se injektáž kódu dělí na SQL Injection, LDAP Injection, XPath Injection. (Scambray, a další, 2001 stránky 141 - 143)

WordPress, stejně jako většina aplikací, používá MySQL databázi, a proto bude v práci popsáno pouze SQL Injection. LDAP i XPath Injection však funguje na stejném principu a rozdíl je akorát v kódu, který útočník do neošetřeného vstupního pole vkládá. (Stuttard, a další, 2011 stránky 344 - 354)

3.13.1 SQL Injection

SQL Injection je útok, který umožňuje útočníkovi vložit SQL výraz do kódu aplikace a získat tak přístup do databáze. Útočník je injektáží schopný provádět všechny nebo některé CRUD akce. CRUD je zkratka pro vytváření (Creating), čtení (Reading), úpravu (Updating) a mazaní (Deleting) dat v databázi. (Yaworski, 2016 str. 64)

SQL jazyk pro CRUD akce obsahuje funkci *SELECT*, *INSERT*, *UPDATE* a *DELETE*. Funkce jsou dále rozšířitelné o klauzule. Nejčastěji se používá klauzule *WHERE*, *ORDER BY* a *GROUP BY*. Každá ze zmíněných funkcí může být náchylná na SQL Injection. (Stuttard, a další, 2011 stránky 294 - 297)

3.13.2 Zranitelnosti

Všechny uživatelské vstupy jako formulářová pole, parametry v URL, cookies a HTTP hlavičky jsou zpracovávány na serveru. Následně dochází k jejich uložení do databáze nebo je na základě nich vybrán odpovídající záznam. V obou případech je na serveru spuštěn SQL dotaz, jehož součástí jsou zmíněné hodnoty. Aplikace, která nedostatečně ošetří vstup od uživatele umožňuje útočníkovi změnit konstrukci těchto SQL dotazů. (Stuttard, a další, 2011 str. 298) (Clarke, 2012 str. 22)

Všechny větší webové aplikace obsahují vyhledávací formulář na webu, který na základě vloženého řetězce vyhledá odpovídající záznamy. Formulář může být navíc odeslán metodou *GET* namísto metody *POST*. URL v takovém případě vypadá např. následovně - *http://www.victim.com/products.php?search=pharse*. Hodnota v parametru *search* je následně vložena do SQL dotazu, který aplikace provede. (Clarke, 2012 stránky 18, 23) (Vrána, 2012 str. 266)

Parametry v URL nemusí být zapsané ve formátu patrném z příkladu dříve, nýbrž ve formátu čitelnějším, který se používá především pro URL obsahových stránek. Výše uvedená URL adresa, tak může vypadat např. následovně - *http://www.victim.com/products/search/pharse*. Zpracování hodnot však aplikace provádí bez změny a jakákoliv URL obsahové stránky, tak může být zdrojem útoku do databáze. (Sirovich, a další, 2008 stránky 59 - 61)

Zranitelnost aplikace vzniká ve zmíněných místech kvůli nedostatečnému ošetření vstupních dat. (Clarke, 2012 stránky 32 - 37) dále udává tři základní nedostatky při validaci vstupních hodnot:

- Povolení apostrofu, který odděluje data od SQL kódu a útočníkovi umožňuje spustit vlastní SQL dotaz.
- Absence ošetření datového typu hodnot, což útočníkovi umožňuje vložit řetězec do dotazů, kde je očekávaná číselná hodnota, která není ohraničena apostrofy.
- Dynamické generování SQL dotazů na základě zvolených hodnot v uživatelském rozhraní, přičemž hodnoty představují názvy sloupců či tabulek, vede ke vzniku další zranitelnosti aplikace.

Kromě nedostatečného ošetření vstupních dat vzniká zranitelnost aplikace také nevhodným zpracováním chyb z databáze. Pokud jsou chyby z databáze zobrazeny uživateli, není pro útočníka problém tyto chyby využít k dohledání správných názvů tabulek či sloupců a složení korektního SQL dotazu. (Clarke, 2012 str. 38)

3.13.3 Způsoby injektáže

Způsobů injektáže je hned několik. Tím nejzákladnějším je vložení apostrofu do vstupní hodnoty společně s částí dotazu, která se má provést s celým příkazem. Při útoku se využívá toho, že apostrof v SQL slouží k ohraničení řetězce, který tímto předčasně ukončíme. Vše za apostrofem se tak stane součástí samotného dotazu. Neošetření apostrofu ve vstupu

nevede pouze k náchylnosti na SQL Injection, ale také k vyvoláni chyby v případě, že uživatel zadá apostrof bez postranních úmyslů. Příkladem může být příkaz:

SELECT patient_records FROM tblPatients WHERE user_search = 'McSorley's ',

kde *McSorley's* je jméno pacienta, pro kterého se mají v databázi vyhledat záznamy. Dotaz by skončil chybou, protože „*s '*“ není platným SQL výrazem. Zadáním platného výrazu by ale útočník mohl získat například data z jiné tabulky či obejít mechanismy zabudované do aplikace. (Long, 2005 str. 447)

Komentář

Komentáře se v SQL značí dvěma pomlčkami (--). Jejich vložením za apostrof lze docílit toho, že část příkazu bude ignorována. V následujícím příkladu jsou komentáře například použity k ignorování ověření uživatelského hesla.

*SELECT * FROM users WHERE username = 'admin'-- ' AND password = 'foo*'

Část skriptu, která se nakonec provede pouze ověří uživatelské jméno.

*SELECT * FROM users WHERE username = 'admin'*

(Stuttard, a další, 2011 str. 289)

OR 1=1

Výraz, resp. část podmínky *OR 1=1* je užitečná v případě, že útočník neví, jaké uživatelské jméno tabulka v databázi obsahuje. Podmínka *username = 'admin' OR 1=1* bude vždy pravdivá a útočník bude přihlášen. SQL dotaz obohacený o injektáž vypadá následovně:

*SELECT * FROM users WHERE username = ' ' OR 1=1--' AND password = 'foo'*,

kde *' OR 1=1--* je útočníkem vložený kód.

(Stuttard, a další, 2011 str. 290) (Scambray, a další, 2001 str. 142)

INSERT, UPDATE a DELETE

SQL Injection lze provést i u ostatních SQL funkcí. Například pří registraci, kde se používá funkce *INSERT* je možné docílit toho, že aplikace útočníkovi udělí například jiná práva. Jedinou podmínkou u *INSERT* funkce, je stejný počet a typ sloupců v první závorce a klauzuli *VALUES*. Příkladem je následující dotaz:

INSERT INTO users (username, password, ID, prvs) VALUES ('foo', 'bar',999,0)--daf', 'secret', 2248, 1)

Obdobně lze provést injektáž u funkce *UPDATE* a *DELETE*. Tyto funkce mohou být stejně jako funkce *SELECT* rozšířeny o klauzuli *WHERE*. (Stuttard, a další, 2011 stránky 295 - 297)

UNION

UNION je operátor, který slouží ke kombinaci výsledků příkazů *SELECT* do jediné sady. Každý *SELECT* může přitom vybírat data z jiné tabulky, čímž umožňuje útočníkovi vybrat data, která potřebuje. Předpokladem pro úspěšné provedení příkazu je rovnost počtu a datových typů sloupců v obou tabulkách. Útočník, tak kromě názvu tabulky, z které chce data získat, musí znát i sloupce, které jsou vybírány. Pokud aplikace zobrazuje chybové zprávy z databáze, tak je jejich uhodnutí jednoduché. Příklad SQL Injection s *UNION* operátorem je následující:

SELECT name, description, price FROM products WHERE category = 1 UNION SELECT 'A', 'B', 3 FROM all_tables'

(Stuttard, a další, 2011 stránky 304, 305) (sqlinjection.net, 2017)

Filtrování znaků

Některé aplikace mají implementované filtry, které odstraňují znaky ze vstupu často používané k SQL Injection. (Stuttard, a další, 2011) udávají způsoby, jak předejít blokování znaků:

- V případě blokování znaku pro komentář, lze místo *' OR 1=1--'* napsat *' OR 'a'='a* nebo použít komentáře v daném programovacím jazyku, ve kterém je SQL dotaz spuštěn, např. /*.
- V případě blokování více znaků, lze znaky nahradit za kódy v ASCI tabulce, např. *CHR(109).*
- V případě blokování klíčových slov jako je např. *SELECT* může fungovat jeden z následujících zápisů:
    - *SeLeCt*
    - *%00SELECT*
    - *SELSECTECT*
    - *%53%45%4c%45%43%54*
    - *%2553%2545%254c%2545%2543%2554*

Žádný výstup

Ve spoustě případů aplikace nevrací žádný výsledek po aplikování SQL Injection. Útočník tak nemůže zjistit, zda byl jeho útok úspěšný či nikoliv. Podle typu databázového serveru lze ovšem použít funkce, která otevře síťové spojení na útočníkův počítač. Přes toto spojení pak lze přenést libovolná data z databáze, včetně informace o úspěšnosti injektáže. (Stuttard, a další, 2011 stránky 316, 317)

3.13.4 Obrana proti SQL Injection

Při programování aplikace by data neměla být vkládána přímo do dotazů, ale nejprve by se měl definovat celý SQL kód a až později přidána data. Takovému dotazu se říká parametrizovaný a v PHP lze k těmto účelům využít knihovnu PDO, Dibi či některou ORM (Object-Relational Mapping) knihovnu. Připravené dotazy zajišťují, že útočník není schopen změnit záměr dotazu, přestože je kód napaden útočníkem. (OWASP, 2017) (Vrána, 2012 stránky 227-230, 251)

Aplikace, která nezbytně potřebuje provádět SQL dotazy tvořené z uživatelsky definovaných názvů tabulek a sloupců, by měla tyto hodnoty dostatečně ošetřit. Pokud nelze dotazy předělat tak, aby neobsahovaly uživatelské hodnoty, lze alespoň provést mapování těchto hodnot na očekávané názvy tabulek a sloupců. (OWASP, 2017)

Poslední možností je ošetření uživatelských vstupů pomocí, tzv. *escaping*. Jedná se o odstranění znaků, které by mohly ovlivnit funkčnost dotazů. Například v PHP se pro tyto účely používá funkce *mysql_real_escape_string*, která zohledňuje všechna specifika MySQL a respektuje nastavené kódování. (OWASP, 2017) (Vrána, 2012 str. 250)

Doplňujícím opatřením proti SQL Injection je minimalizování škod, při úspěšném útoku. Uživatel databáze, pod kterým je dotaz, a tedy i útok prováděn, by měl mít proto práva k těm tabulkám a SQL funkcím, které nezbytně potřebuje. Dalším opatřením je nevypisování chybových hlášek z databáze, využití firewallu pro webový server, např. *ModSecurity*, a monitorování všech provedených dotazů na databázi. (OWASP, 2017) (Weiss, 2016)

3.14 Útoky na uživatele

Všechny doposud zmíněné útoky byly sice provázány s uživatelem, ale útok byl prováděn na server, a nikoliv na uživatele. Využívalo se přitom nedostatečného zabezpečení aplikace na straně serveru. Stejně je tomu u útoků na uživatele, u kterých se navíc využívají některé

aspekty chování aplikace. Některé útoky, např. Session Hijacking a Session Fixation, již byly zmíněny, a proto se bude tato kapitola soustředit na ostatní útoky vedené na uživatele aplikace. Vynechány budou také útoky, které využívají zranitelnosti prohlížečů nebo Active-X prvků napsaných nejčastěji v nativních jazycích jako je např. C/C++. (Stuttard, a další, 2011 stránky 431, 432, 555)

3.14.1 Cross-Site Scripting

Cross-Site Scripting, zkratka XSS, česky skriptování napříč sítí, je útok založený na injektáži kódu do webové stránky aplikace. Škodlivý kód, kterým je většinou JavaScript či jiný skriptovací jazyk běžící v prohlížeči, je odeslán prostřednictvím webové aplikace jinému uživateli. Prohlížeč nemá žádnou možnost, jak ověřit důvěryhodnost zdroje, a proto škodlivý skript spustí. Útočník tak pomocí XSS může získat token relace, hodnoty v cookies, přepsat obsah HTML stránky či získat jiné citlivé informace uložené v prohlížeči. K útoku se nejčastěji využívají neošetřené uživatelské vstupy, na základě kterých je generován výstup. (OWASP, 2016)

XSS lze dále dělit podle způsobu zpracování uživatelských vstupů na Stored XSS, Reflected XSS a DOM Based XSS. Stored (uložené/perzistentní) XSS vzniká, když je uživatelský vstup ukládán na server, např. do databáze, a poté aplikací načten. Reflected (reflexivní) XSS nastane, když je uživatelský vstup ihned vrácen webovou aplikací v chybové zprávě, výsledku vyhledávání nebo jakékoliv jiné odpovědi. Oba typy XSS útoku mají stejný vzorec chování, ve kterém aplikace přijímá uživatelská data a zobrazuje je uživatelům nezabezpečeným způsobem v HTML. DOM Based XSS neboli XSS útok založený na DOM (Document Object Model), využívá zranitelnosti aplikace, která skrze JavaScript generuje chybové hlášky či jiný obsah na základě parametru v URL, hodnot v cookies či HTTP hlaviček. (OWASP, 2017) (Kümmel, 2011 str. 76) (Klein, 2005)

(Yaworski, 2016) popisuje další typ útoku, tzv. Self XSS, kdy útočník pod určitou záminkou nabádá uživatele ke spuštění škodlivého kódu. Uživatel přitom netuší, že kód slouží k útoku proti němu samotnému.

Zranitelnosti

XSS útoky mohou začínat od prosté změny obsahu webové stránky a končit přidáním nové funkce do aplikace. Záměrem je podvést uživatele, který provede nežádoucí akci, jako je zadání citlivých údajů, které jsou pak předány útočníkovi. Příkladem může být podvrhnutý

přihlašovací formulář, do kterého oběť zadá své přihlašovací údaje. Další nežádoucí akcí může být například SQL Injection. Útočník, tak využije jiného uživatele k získání přístupu do databáze a zůstane přitom neodhalen. (Stuttard, a další, 2011 stránky 444, 446)

K útokům proti uživateli se často využívají také funkce samotného prohlížeče. Jednou z funkcí je automatické doplňování údajů ve formuláři. Prohlížeč si údaje ukládá do cache paměti, ke které lze přes JavaScript přistupovat. Útočník tak může skrze XSS útok data ukrást a odeslat na svůj server. (Stuttard, a další, 2011 str. 446)

Stored XSS, resp. perzistentní XSS je „*nejčastěji aplikováno v diskusních fórech, komentářích pod články a všude tam, kde mají uživatelé možnost uložit trvale na webový server svůj příspěvek*“ (Kümmel, 2011 str. 56). Dále může být aplikováno ve formulářích, kam uživatelé vkládají své osobní údaje, dokumenty, hodnocení, otázky či jiné údaje. Všechny zmíněné uživatelské vstupy mohou být následně aplikací zobrazeny ostatním uživatelům. Některé aplikace navíc do databáze ukládají vstupy z ostatních aplikací nebo komunikačních kanálů, např. SMTP server, které mohou obsahovat škodlivý kód. (Stuttard, a další, 2011 stránky 449, 450)

Perzistentnímu XSS se v případě vkládání pouze HTML kódu říká HTML Injection (Grossman, a další, 2007 str. 75). Využívá se především u zpravodajských serverů, kde útočník skrze tuto zranitelnost provádí tzv. Content Spoofing neboli změnu zobrazeného obsahu. Jedná se především o změnu obsahu článků (Kümmel, 2011 str. 93). Kromě toho existuje také Template Injection, které využívá zranitelnosti šablonovacích systémů. Šablonovací systémy umožňují vývojářům oddělit programovou logiku od prezentované části aplikace a obsahují funkce pro generování dynamického obsahu. Bez dostatečného ošetření uživatelského vstupu mohou být útočníkem tyto funkce zneužity. (Yaworski, 2016 str. 57) (Kettle, 2015)

U reflexivního a DOM Based XSS je útok prováděn pomocí phishingu, tedy hromadného odesílání e-mailů více uživatelům aplikace se škodlivou URL. Útočníci také vytváří webové stránky se zajímavým obsahem a škodlivými URL, na které přilákají uživatele nebo je vkládají do diskuzních fór. (Stuttard, a další, 2011 stránky 448, 449) (Kümmel, 2011 str. 69)

Příklady útoků

Základním Stored XSS útokem je vložení jednoduchého řetězce *<script>alert(/XSS/);</script>* do formulářového pole či jiného uživatelského vstupu, který

se následně ukládá do databáze. Při načtení stránky, kde se uživatelské vstupy generují pro ostatní uživatele, se v případě úspěšného útoku zobrazí výstražné okno s textem */XSS/*. (Kümmel, 2011 stránky 59, 60)

U reflexivního XSS se může skript na server dostat přes HTTP metodu *GET* nebo *POST*. U vyhledávacího formuláře, který hledanou frázi zobrazuje v HTML dokumentu a je odesílán metodou *GET* stačí v URL změnit hledaný výraz za libovolný skript. Skript se pak zobrazí na stránce a spustí. Výsledná URL v takovém případě vypadá následovně: *htttp//www.xssvpraxi.cz/search.php?dotaz=<script>alert(/XSS/);</script>*. (Kümmel, 2011 stránky 64 - 72)

Většina formulářů je ale odesílána metodou *POST*, kde jsou hodnoty parametrů odesílány v těle požadavku. Útočník tak musí přimět uživatele, aby formulář odeslal se škodlivým kódem. Nejčastěji útočník podstrčí uživateli odkaz na webovou stránku, která se postará o odeslání POST požadavku s předem připravenými daty a škodlivým kódem. (Kümmel, 2011 stránky 64 - 72)

DOM Based XSS je velice podobné reflexivnímu XSS a jeho příklad byl již uveden. Jediný rozdíl oproti reflexivnímu XSS je způsob vkládání hodnot do HTML. U reflexivního XSS jsou vkládány hodnoty na straně serveru, kdežto u DOM Based XSS jsou hodnoty parametrů vkládány na straně klienta, a to pomocí JavaScriptu. (Kümmel, 2011 str. 76)

Prolomení obrany

Aplikace se často brání filtrováním výrazu *<script>* či jiných HTML tagů. Vhodnou úpravou výrazu lze docílit toho, že bude skript prohlížečem přesto spuštěn. (Stuttard, a další, 2011 stránky 451, 452, 459) dále udávají následující příklady úpravy výrazu *<script>*:

- *<script >*
- *<ScRiPt>*
- *%3cscript%3e*
- *<scr<script>ipt>*

Kromě filtrování HTML tagů mohou aplikace blokovat celý uživatelský vstup v případě detekce XSS. Aplikace nejčastěji blokuje vstup, který obsahuje výraz *<script>*. Existují ovšem HTML tagy a jejich atributy, do kterých lze javascriptový kód schovat a aplikace ho přijme. (Stuttard, a další, 2011 stránky 456 - 458) dále uvádějí následující příklady:

- *<object data="data:text/html, <script>alert(1)</script>">*
- *Click here*
- *<xml onreadystatechange=alert(1)>*
- *<img type=image src=valid.gif onreadystatechange=alert(1)>*
- *<body onactivate=alert(1)>*
- *<input autofocus onfocus=alert(1)>*
- *<video src=1 onerror=alert(1)*
- *<iframe src=javascript:alert(1)>*

Aplikace dále mohou blokovat samotné HTML komponenty jako jsou tagy, atributy nebo závorky (< a >). Mnoho z těchto filtrů lze obejít přidáním neobvyklých znaků do komponent, takovým způsobem, který prohlížeč toleruje. Na libovolnou pozici lze například vložit ASCII kód *[%00]* reprezentující prázdný bajt nebo nahradit mezeru za znak / či ASCIII kód *[%09]*. Znaky v hodnotě atributu lze také převést do HTML kódování, přičemž tyto znaky jsou později prohlížečem dekódovány. V některých prohlížečích je u HTML tagů dokonce tolerována závorka navíc, jejíž duplikací nemusí dojít k blokaci uživatelského vstupu. (Stuttard, a další, 2011 stránky 460-463)

Nakonec můžou být blokována samotná klíčová slova, závorky, uvozovky a tečky ve skriptu. Stejně jako u HTML komponent i v JavaScriptu existují techniky, kterými lze upravit skript a zachovat jeho funkčnost. Využívá se například Unicode, hexadecimální nebo osmičkové kódování, kterým se nahrazují blokované znaky. (Stuttard, a další, 2011 stránky 465, 466)

Další obranou aplikací je omezení délky uživatelského vstupu, kdy útočník nemůže zadat rozsáhlý škodlivý kód. Řešením může být spuštění vzdáleného skriptu, např. *http://www.acme.com/path/to/search.asp?query="><script src="http://evil.com/s.js"/>*. Nevýhodou je odhalení totožnosti útočníka a následné zablokování serveru obsahujícího škodlivý skript. Dalším řešením je převedení, v tomto případě reflexivního XSS, na DOM Based XSS. Využívá se k tomu funkce *eval* a javascriptová funkce *location.hash.substr(1)*, která vrátí hodnotu za znakem #. Výsledný skript pak vypadá následovně: *http://www.acme.com/path/to/search.asp?query="><script>eval(location.hash.subst r(1))</script>#alert('xss').* (Grossman, a další, 2007 stránky 131, 132)

Prevence proti XSS

Ochranu proti XSS obsahují samotné prohlížeče. Jedná se především o filtry jako je například NoScript od Firefoxu, IE XSS Filter od Microsoftu nebo XSS filter od Google Chrome. Zároveň lze prohlížeči přímo říci, zda se jedná o legitimní obsah pomocí HTTP hlavičky *X-Content-Security-Policy*, dnes pouze *Content-Security-Policy*. Prohlížeči je tak předána informace o tom, že aplikace implementuje bezpečnostní politiku Content Security Policy (CSP). Prohlížeč na základě definovaných direktiv webovou stránku zpracuje. (Kümmel, 2011 stránky 252, 253, 261) (Mozilla, 2018)

Na straně webové aplikace je potřeba vždy validovat vstupní hodnoty a zároveň generovaný výstup. Vstupní hodnoty od uživatele by nikdy neměly být příliš dlouhé, měly by obsahovat pouze určité znaky a odpovídat určitému regulárnímu výrazu. Znaky se speciálním významem ve výstupu, který je generován ze vstupních hodnot, by měly být ošetřeny. Pro reflexivní nebo persistentní XSS lze k tomuto účelu využít PHP funkci *htmlSpecialChars*, která v základním chování převede znaky *<"&>* na tzv. HTML entity (např. *<* pro znak <), které se po interpretaci HTML zobrazí jako daný znak. (Stuttard, a další, 2011 stránky 492, 493) (Vrána, 2012 str. 356)

Vstupní a výstupní filtry mohou selhat v případě, že je útočníkův vstup v neobvyklém kódování, a proto by aplikace měla explicitně specifikovat typ kódování v HTTP hlavičce odpovědi. Například: *Content-Type: text/html; charset=ISO-8859-1*. V některých případech, např. u textového editoru ve webové aplikaci, se očekává HTML vstup od uživatele, který nelze zcela na vstupu filtrovat. V takové situaci se kontroluje vstup s hodnotami v tzv. whitelistu nebo blacklistu, tedy s povolenými nebo zakázanými řetězci. (Stuttard, a další, 2011 stránky 495, 496) (Kümmel, 2011 stránky 249, 250)

U DOM-Based XSS, kde jsou data zpracovávána bez přímé kontroly na serveru, je nutné opět validovat vstup. Především je nutné kontrolovat, zda je v URL odpovídající počet parametrů a zda je jejich název správný. Hodnota parametrů by zároveň měla obsahovat pouze alfanumerické znaky. Před vložením uživatelských dat do dokumentu je nutné provést jejich validaci. K ošetření znaků se speciálním významem lze využít některou z dostupných javascriptových knihoven nebo si vytvořit vlastní funkci. (Stuttard, a další, 2011 stránky 496, 497) (OWASP, 2017)

3.14.2 Request Forgery

Request Forgery, známá také jako Session Riding, je kategorie útoků jejíž cílem je vytvoření požadavků, které napadený uživatel nemá v úmyslu udělat. Útok může být proveden i přesto, že je aplikace chráněna proti XSS. Request Forgery se dá rozdělit na On-Site a Cross-Site. (Stuttard, a další, 2011 stránky 501, 502)

On-Site Request Forgery

On-Site Request Forgery (OSRF) využívá zranitelnosti aplikace, která umožňuje vložit škodlivý kód do obsahu stránky. Spousta aplikací umožňuje přidávat do obsahu obrázky či upravovat jejich URL. Na stránce, kde je obrázek zobrazen, je na danou URL proveden HTTP požadavek *GET*. Útočník může zaměnit URL obrázku za metodu aplikace, která například přidává nového uživatele. Úspěšný útok je v tomto případě vykonán, když administrátor aplikace načte stránku s obrázkem. Ošetření znaků se speciálním významem je v tomto případě neefektivní, protože prohlížeč URL dekóduje ještě předtím, než je proveden požadavek. Ochranou je tak striktní validace uživatelského vstupu či stejný způsob popsaný dále u Cross-Site Request Forgery.(Stuttard, a další, 2011 stránky 502, 503)

Cross-Site Request Forgery

Cross-Site Request Forgery (CSRF nebo také XSRF) útok je na rozdíl od OSRF prováděn z jiné webové stránky (Kümmel, 2011 str. 78). Útočník může například vytvořit webovou stránku s formulářem, který se při načtení automaticky odešle. Akce, která je při odeslání vykonána je ovšem na jiné webové stránce. Pokud takovou stránku navštíví administrátor, který je současně přihlášený do zranitelné aplikace, může útočník docílit toho, že bude vytvořen nový účet. Úryvek kódu takového formuláře je následující:

<form action="https:://mdsec.net/auth/390/NewUserStep2.ashx" method="POST">
<input type="hidden" name="username" value="admin">
...
<script> document.forms[0].submit();</script>
(Stuttard, a další, 2011 stránky 504, 505)

Kromě formuláře, lze využít k útoku HTML značku *iframe, img, link, bgsound* či jinou značku, která načítá obsah z externího souboru metodou *GET*. Útočník přitom nemusí vytvářet nový web, ale může využít některou z již existujících aplikací, které umožňují tyto HTML tagy přidávat (Kümmel, 2011 str. 83). (Grossman, a další, 2007 str. 94) uvádějí

následující příklad pro zaslání milionu dolarů v internetovém bankovnictví na útočníkův účet:

<iframe src=https://somebank.com/transferfunds.asp?amnt=1000000&acct=
123456></iframe>

Obranou proti CSRF může být využití hlavičky *referer*, která obsahuje údaj, z jaké stránky byl požadavek odeslán. Zároveň ale obsahuje všechny hodnoty proměnných v URL, které mohou být důvěrné. Hlavička *referer* může být z tohoto důvodu prohlížečem či firewallem blokována a nedá se na ni spolehnout. Nejúčinnější způsob ochrany představuje použití tokenů. Token je vygenerován a odeslán společně s formulářem. Současně je uložen do session nebo databáze. Po odeslání formuláře je před spuštěním operace ověřena jeho existence a zda odpovídá dané akci a uživateli. (Kümmel, 2011 stránky 84 - 86) (Vrána, 2012 str. 363)

3.14.3 Clickjacking

Clickjacking je útok, který využívá rámy (nejčastěji *iframe*) k vyvolání operací, které jsou chráněné proti CSRF. Princip je takový, že útočník na své webové stránky umístí rám, který obsahuje cílový formulář. Rám překryje jiným obsahem, např. hrou. Uživatel, který hru hraje přitom netuší, že klikáním provádí operace v cílové aplikaci. Proti Clickjacking se lze bránit pomocí JavaScriptu, který kontroluje, zda je stránka na vrcholu hierarchie DOM. Další možností je použití HTTP hlavičky *X-Frame-Options: deny*, která je dostupná v nejnovějších prohlížečích a slouží k zakázání načítání stránky v rámu. Nakonec lze ještě použít hlavičku *frame-ancestors*, která je součástí bezpečnostní politiky CSP a definuje zdroje, které mohou danou stránku do rámu načíst. (Kümmel, 2011 stránky 87, 91, 92) (Vrána, 2012 str. 365)

3.14.4 JavaScript Hijacking

JavaScript Hijacking umožňuje útočníkovi obejít Same Origin Policy v případě, že aplikace používá JavaScript ke zpracování důvěrných informací způsobem, který nebyl při vytváření politiky předpokládán. Same Origin Policy totiž umožňuje na jedné doméně vložit skript z druhé domény a ten spustit. Toho využívají útočníci, kteří na svůj web vloží skript z jiné domény a čekají, až jejich stránku navštíví uživatel, který je přihlášen do cílové aplikace. Důvěrná data dostupná pouze přihlášenému uživateli, která jsou vloženým skriptem vrácena, jsou útočníkem následně odchycena. Zranitelné jsou tedy aplikace, které používají

JavaScript, jako mechanismus pro přenos dat ze serveru do klientského prohlížeče. Používá se především Ajax, tedy asynchronní požadavek na server, který v odpovědi vrací data, nejčastěji ve formátu JSON, která mají být v aplikaci zobrazena. Následující příklad ukazuje, jak lze jednoduše vypsat proměnnou *nonce*, která je definována ve skriptu cílené aplikace:

<script src="https://wahh-network.com/status"></script>
<script> alert(nonce); </script>

(Stuttard, a další, 2011 stránky 519 - 522) (Chess, a další, 2007)

Proti JavaScript Hijacking je možné se bránit přidáním nevalidního javascriptového kódu, který je před spuštěním skriptu v aplikaci odstraněn. Vložený skript pomocí značky *<script>* na útočníkově straně lze ovšem pouze spustit a nikoliv upravit. Současně lze nastavit, aby aplikace přijímala pouze POST požadavky na potenciálně napadnutelný skript. (Stuttard, a další, 2011 str. 524)

4 Praktická část

Praktická část je rozdělena do tří částí. V první části je stručný popis vybraných pluginů se souhrnem nabízených funkcí. V další části je samotná analýza efektivnosti pluginů z hlediska obrany proti vybraným útokům. Nakonec je zhodnocení výsledků a vzájemná komparace pluginů.

4.1 Vybrané pluginy

Bezpečnostní pluginy systému WordPress byly vybrány na základě počtu aktivních instalací a funkcí, které nabízejí. Bylo vybráno 10 nejpoužívanějších pluginů, které obsahují ochranu proti prováděným útokům. Z toho důvodu byly vynechány pluginy, které jsou sice hojně využívané, ale specializují se pouze na konkrétní útoky nebo neobsahují dostatečný počet funkcí. Vynechán byl například plugin Anti-Malware Security and Brute-Force Firewall, SiteGuard WP Plugin a Cerber Security & Antispam.

Vybrané pluginy nejčastěji obsahují ochranu proti útoku hrubou silou, monitorování běhu aplikace, firewall a skenování aplikace, tzn. hledání malwarů a škodlivých kódů. Pluginy jsou dohledatelné na oficiálních stránkách WordPressu (WordPress).

Vybrány byly následující pluginy:

- Jetpack,
- Wordfence Security,
- iThemes Security,
- All In One WP Security & Firewall,
- Sucuri Security,
- Acunetix WP Security,
- BulletProof Security,
- Shield Security for WordPress,
- NinjaFirewall a NinjaScanner,
- SecuPress.

4.1.1 Jetpack

Plugin Jetpack má více jak 4 mil. aktivních instalací a 66 mil. stažení. Patří tedy mezi nejpoužívanější bezpečnostní pluginy. Důvodem je fakt, že kromě zabezpečení webových

stránek nabízí i řadu dalších užitečných funkcí. Plugin vyvinula společnost Automattic Inc. ve spolupráci s Adamem Hecklerem. V bezplatné verzi nabízí plugin nedostatečné množství funkcí pro ochranu webových stránek, a proto byla zakoupena verze Premium, která stojí 216 CZK/měsíc (Jetpack). Analyzována byla verze 5.7, která nabízí následující bezpečnostní funkce:

- ochrana proti útoku hrubou silou,
- monitorování běhu aplikace,
- zabezpečení přihlašování a dvoufázová autentizace,
- skenování aplikace – hledání malwaru, hledání škodlivého kódu,
- zálohování webových stránek.

(Automatic)

4.1.2 Wordfence Security

Wordfence Security je dalším velice používaným bezpečnostním pluginem s více jak 2 mil. aktivními instalacemi a 60 mil. staženími (Wordfence, 2017). Nabízí se bezplatně nebo ve verzi Premium za 100 USD/rok, která je obohacena o další funkce. Analyzována bude pouze bezplatná verze 6.3.22, která nabízí následující zabezpečení:

- firewall, který slouží k identifikaci škodlivého zacházení s aplikací a zablokování útoků,
- manuální blokování uživatelů nebo robotů na základě IP adresy,
- ochrana proti útoku hrubou silou,
- skenování aplikace – hledání malwarů, špatných URL adres, škodlivých kódů a dalších zranitelných míst,
- monitorování běhu aplikace a zaznamenávání neočekávaných událostí.

(Wordfence, 2017)

4.1.3 iThemes Security

Plugin iThemes Security má více jak 800 tis. aktivních instalací a 12 mil. stažení. Nabízí se v bezplatné verzi a ve verzi Pro, která stojí 80–197 USD/rok, podle počtu zabezpečených webových stránek. Analyzována bude bezplatná verze 6.8.0, která nabízí následující ochranu:

- ochrana proti útoku hrubou silou,
- blokování uživatelů, kteří vygenerují příliš mnoho chyb, při kterých server vrací HTTP kód 404,
- vynucení silného hesla pro uživatelské účty a možnost změny tzv. soli u hesla,
- detekce změn v souborech,
- zálohování databáze,
- e-mailové upozornění o neočekávaných událostech,
- kontrola práv souborů a složek.

(Jean, 2017)

4.1.4 All In One WP Security & Firewall

All In One WP Security & Firewall je plugin vyvinut skupinou lidí, která vystupuje pod názvem "Tips and Tricks HQ". Plugin má vice jak 600 tis. aktivních instalací, 5 mil. stažení, je zdarma a analyzovaná verze 4.3.1 nabízí následující funkce:

- ochrana proti útoku hrubou silou,
- automatické zálohování databáze,
- skenování aplikace – identifikace nesprávných práv u souborů a složek, hledání změn v souborech,
- zamezení nalezení uživatelského jména a detekce stejného uživatelského jména pro přihlašování a zobrazovaného jména na webu,
- monitorování běhu aplikace a zaznamenávání neočekávaných událostí,
- firewall – ochrana proti nejběžnějším útokům.

(Amin, a další)

4.1.5 Sucuri Security

Sucuri Security, celým názvem Sucuri Security – Auditing, Malware Scanner and Hardening, je plugin vyvinutý společností Sucuri Inc. s vice jak 300 tis. aktivními instalacemi a 2,5 mil. staženími. Bezplatná verze neobsahuje firewall, a proto byla zakoupena placená verze nazývaná CloudProxy-Basic za 10 USD/měsíc (Sucuri Inc., 2018). K plnému zprovoznění placené verze a zprovoznění firewallu bylo potřeba nastavit několik API klíčů a změnit „A" záznam v DNS na IP adresu definovanou v administraci na webu Sucuri. Plugin ve zvolené verzi 1.8.11 nabízí následující ochranu:

- ochrana proti útoku hrubou silou,
- firewall – ochrana proti DOS/DDOS útoku, SQL Injection, XSS a malwaru,
- skenování aplikace – detekce změn v souborech, hledání malwaru,
- monitorování běhu aplikace a zaznamenávání neočekávaných událostí,
- upozornění e-mailem o prováděném útoku hrubou silou či jiném útoku,
- možnost zablokování uživatelů dle jejich IP adresy.

(Sucuri Inc.)

4.1.6 Acunetix WP Security

Acunetix WP Security je bezpečnostní plugin od společnosti Acunetix Ltd., který má více jak 80 tis. aktivních instalací a 1,5 mil. stažení. Z neznámého důvodu je k dispozici plugin od stejného autora a se stejným popisem funkcí, který se jmenuje Acunetix Secure WordPress a má 60 tis. aktivních instalací. Sečteme-li instalace, tak se dostáváme na 140 tis. aktivních instalací. V práci bude analyzován plugin Acunetix WP Security, ale lze očekávat, že zjištěné hodnoty budou platit i pro druhý plugin. Verze 4.0.5 nabízí následující funkce:

- skenování aplikace a hledání základních bezpečnostních zranitelností,
- zamezení procházení adresáře z webového rozhraní,
- zálohování databáze a zamezení vypisování chyb z databáze či PHP,
- monitorování běhu aplikace a zaznamenávání neočekávaných událostí.

(Acunetix Ltd.)

4.1.7 BulletProof Security

Plugin BulletProof Security má 90 tis. aktivních instalací, 2,5 mil. stažení a je nabízen v bezplatné verzi a ve verzi Pro. Verze Pro stojí jednorázově 70 USD. Analyzována bude bezplatná verze 2.8, která nabízí následující ochranu:

- skenování aplikace – změny v souborech a hledání malwaru,
- ochrana proti útoku hrubou silou,
- zabezpečení adresářů pomocí souboru htaccess,
- zálohování databáze,
- monitorování běhu aplikace a zaznamenávání neočekávaných událostí.

(AITpro Website Security)

4.1.8 Shield Security for WordPress

Shield Security for WordPress, zkráceně Shield Security, je plugin, který má 70 tis. aktivních instalací, 3 mil. stažení a nabízí se v placené a bezplatné verzi. Placená verze stojí 14,5 USD/rok. Bezplatná verze 6.2.2, která je dále analyzována, nabízí tyto funkce:

- ochrana proti útoku hrubou silou,
- firewall – ochrana proti nejběžnějším útokům,
- redukování spamu v komentářích u článku,
- nastavení bezpečnostních hlaviček,
- monitorování neočekávaných událostí.

(iControlWP)

4.1.9 NinjaFirewall a NinjaScanner

NinjaFirewall a NinjaScanner jsou dva pluginy od stejného vývojáře a v této práci byly analyzovány jako jeden celek. Oba pluginy jsou zdarma a lze je ve WordPressu přímo propojit a přistupovat k nim z jednoho uživatelského rozhraní. Kvůli přehlednosti textu bude v práci v některých případech zmiňován pouze plugin NinjaFirewall, ovšem bude se jednat o spojení pluginu NinjaFirewall a NinjaScanner.

Plugin NinjaFirewall má vice než 20 tis. aktivních instalací a 250 tis. stažení. Ve verzi 3.6 nabízí funkce jako je ochrana proti útoku hrubou silou, ochrana proti nejběžnějším útokům, rozpoznání změn v souborech, monitorování aplikace v reálném čase a e-mailové upozornění o neočekávaných událostech. Při instalaci lze zvolit WAF mód nebo Full WAF mód, který dokáže aplikovat firewall napříč celou adresářovou strukturou a zamezit spuštění škodlivých skriptů (The Ninja Technologies Network, 2017). Pro analýzu tohoto pluginu byl zvolen Full WAF mód.

NinjaScanner má vice jak 600 aktivních instalací a 4 tis. stažení. Verze 1.1 obsahuje vylepšenou funkci pro hledání změn v souborech a funkci pro hledání malwaru ve webové aplikaci. (The Ninja Technologies Network)

4.1.10 SecuPress

SecuPress je plugin jehož první verze byla vydána teprve 23.08.2016. Do dnešního dne (02.02.2018) má vice jak 8 tis. aktivních instalací a přes 45 tis. stažení. Plugin v bezplatné verzi nabízí spoustu funkcí, avšak skenování aplikace je dostupné pouze v placené verzi, která stojí 60 EUR/rok. Zmíněná funkcionalita je nezbytná pro analýzu efektivnosti pluginu při obraně proti vloženému malwaru. Z tohoto důvodu byla placená verze zakoupena. Analyzovaná verze 1.3.3 nabízí následující funkce:

- ochranu proti útoku hrubou silou,
- firewall – blokování škodlivých požadavků,
- skenování aplikace – hledání malwaru a změn v souborech,
- zálohování souborů a databáze,
- zaznamenávání neočekávaných událostí a zasílání e-mailových upozornění.

(SecuPress)

4.2 Analýza efektivnosti bezpečnostních pluginů

Podle (Wright, 2017) má systém WordPress nedostatečnou ochranu proti útoku hrubou silou, SQL Injection, Cross-Site Scripting, malwaru a File Inclusion, kdy je snahou získat obsah souborů s citlivými informacemi. Bezpečnostní pluginy proto obsahují funkce, které tyto nedostatky redukují. Na základě těchto poznatků byla efektivnost bezpečnostních pluginů analyzována z hlediska ochrany proti:

- zjištění uživatelského jména,
- útoku hrubou silou,
- malwaru,
- přístupu k souborům s citlivými informacemi,
- SQL Injection,
- Cross-Site Scripting.

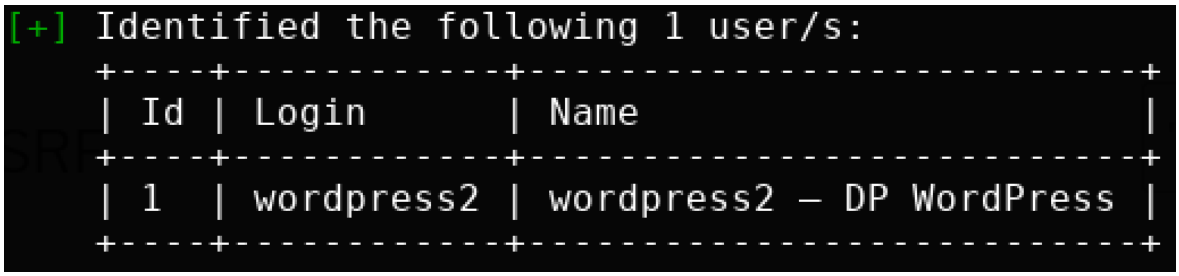
4.2.1 Zjištění uživatelského jména

K získání uživatelského jména lze využít nástroj WPScan, který je například dostupný v operačním systému Kali Linux. Tento nástroj složí k nalezení zranitelností na webových stránkách běžících na systému WordPress (WPScan). Jednou z jeho funkcí je nalezení uživatelských jmen, která jsou v systému registrovaná. S nástrojem se pracuje v terminálu a příkaz pro nalezení uživatelů je například následující:

wpscan --url domena.cz --enumerate u

V případě nalezení uživatelů v systému, je na terminálu zobrazen jejich výčet. Ukázka výstupu je na následujícím obrázku (Obrázek 1).

Obrázek 1 - Nalezení uživatelé pomocí nástroje WPScan



Zdroj: vlastní zpracování

Během analýzy bezpečnostních pluginů bylo dále zjištěno, že následující pluginy blokují hledání uživatelských jmen pomocí nástroje WPScan:

- Wordfence Security,
- All In One WP Security,
- Shield Security,
- NinjaFirewall a NinjaScanner.

Uživatelské jméno lze získat také zadáním parametru *author* s hodnotou *1*, která značí ID uživatele v databázi. Stránka je následně přesměrována na URL, jejíž součástí je uživatelské jméno patřící danému uživateli. Dále lze získat uživatelská jména přes REST API WordPressu. Stačí zadat URL adresu */wp-json/wp/v2/users*. V některých pluginech je možné podporu REST API vypnout. V následující tabulce (Tabulka 1) je přehled pluginů a k nim informace, zda zabezpečují webové stránky proti získání uživatelského jména prostřednictvím parametru *author* nebo přes REST API.

Tabulka 1 - Zabezpečení jednotlivých pluginů proti zjištění uživatelského jména

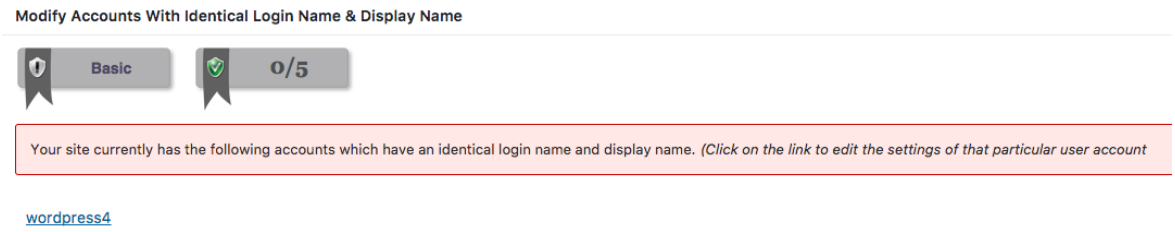
Název pluginu	**Zabezpečení parametru *author***	**Zabezpečení */wp-json/wp/v2/users***
Jetpack	Ne	Ne
Wordfence Security	Ano	Ano
iThemes Security	Ne	Ano
All In One WP Security & Firewall	Ano	Ano
Sucuri Security	Ne	Ne
Acunetix WP Security	Ne	Ne
BulletProof Security	Ne	Ne
Shield Security for WordPress	Ano	Ano
NinjaFirewall a NinjaScanner	Ano	Ano
SecuPress	Ne	Ano

Zdroj: vlastní zpracování

Spousta šablon zobrazuje u článku autora s odkazem na všechny články, které kdy přidal. Defaultně se zobrazuje uživatelské jméno. Bezpečnostní plugin by měl docílit toho, že se u článku zobrazí pouze jméno a příjmení autora. Ze stejných údajů by pak měla být složena i URL webové stánky, kde jsou všechny články daného autora. Zároveň by měl plugin ošetřit, aby uživatelské jméno a jméno autora nebylo stejné.

Analýzou bezpečnostních pluginů bylo zjištěno, že pouze plugin All In One WP Security se snaží získání uživatelského jména ještě více eliminovat. Plugin přiděluje body za každé aktivované zabezpečení. V případě jiného uživatelského jména a jména zobrazeného vedle článku, je do celkového součtu přičteno 5 bezpečnostních bodů. V URL však uživatelské jméno stále existuje. Upozornění na nedostatečné zabezpečení uživatelského jména v pluginu All In One WP Security je na následujícím obrázku (Obrázek 2).

Obrázek 2 - Upozornění v pluginu All In One WP Security na stejné uživatelské jméno se jménem zobrazovaným na webu



Zdroj: vlastní zpracování

Ze všech testovaných pluginů je na tom nejlépe plugin All In One WP Security, Wordfence, Shield Security a NinjaFirewall. Přestože zmíněné pluginy nabízejí dobrou ochranu, tak lze uživatelské jméno stále získat z URL autora u článku. Uživatel s neomezenými právy ve WordPressu, by proto nikdy neměl přidávat žádný příspěvek. Útočník, který uživatelské jméno získá, může snadněji provést útok hrubou silou.

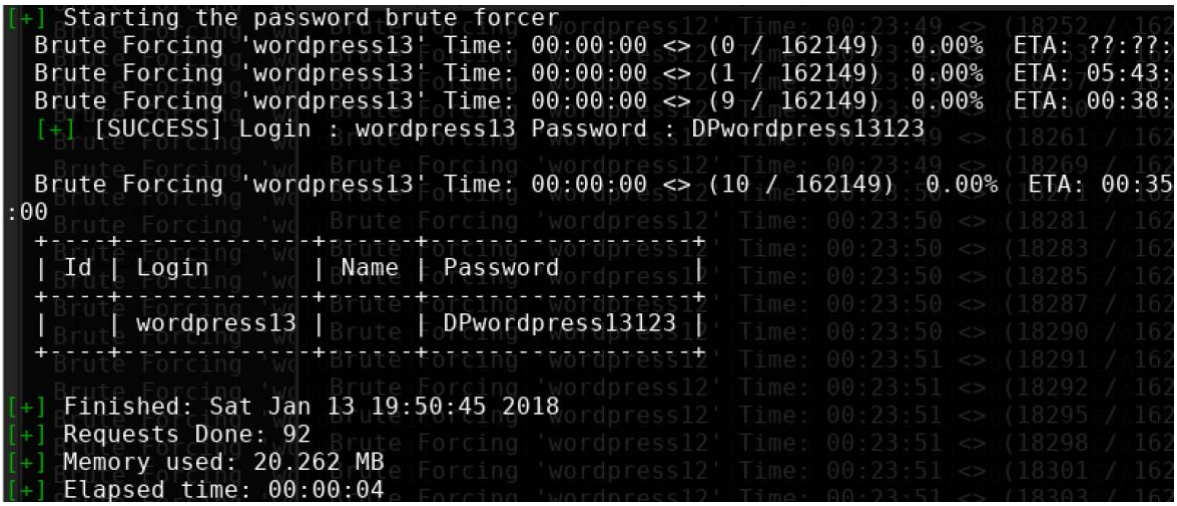
4.2.2 Útok hrubou silou

Pro útok hrubou silou, tedy uhodnutí hesla, bude použit nástroj WPScan popsaný v předchozí kapitole. Zároveň bude použit program Tor, který umožňuje anonymizaci uživatele (The Tor Project). Díky němu bude útok prováděn z jiné IP adresy, než je adresa skutečná. V opačném případě by pluginy útok hrubou silou nemusely blokovat, protože současná IP adresa 192.168.0.1, jak bylo zjištěno, je ve většině případech uložena v tzv. whitelistu. Po nainstalování a konfiguraci programu Tor do Kali Linux, dle dokumentace (The Tor Project), stačí spustit útok nástrojem WPScan, např. tímto příkazem:

wpscan --url domena.cz --proxy socks5://127.0.0.1:9050 --wordlist wordlist.txt --username wordpress2

WPScan projde všechna hesla ve slovníku a pokusí se s nimi přihlásit. Slovníky s nejběžnějšími hesly jsou dohledatelné na internetu nebo lze vygenerovat vlastní prostřednictvím nástroje Crunch, který je také dostupný v Kali Linux. Výstup po úspěšném uhodnutí hesla nástrojem WPScan je na následujícím obrázku (Obrázek 3).

Obrázek 3 - Úspěšný útok hrubou silou pomocí nástroje WPScan



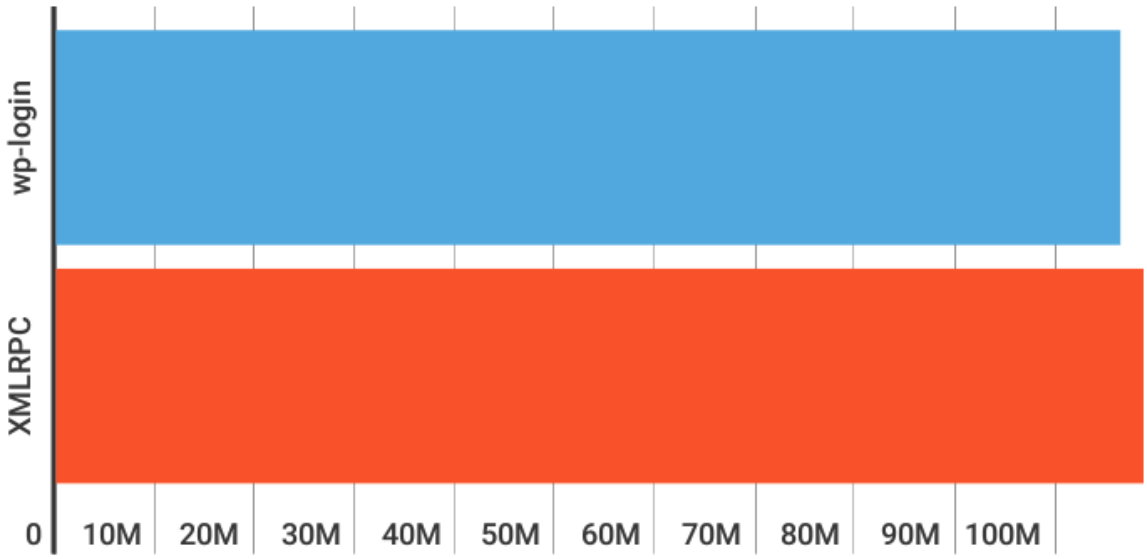
Zdroj: vlastní zpracování

Na základě teoretických poznatků by bezpečnostní plugin systému WordPress měl monitorovat pokusy o přihlášení, zablokovat uživatele po několika pokusech a přihlášení znovu povolit až po několika minutách. Zároveň by měl umožnit ochranu formuláře, např. pomocí captchy či dvoufázového přihlášení, a vyžadovat nastavení složitého hesla.

Ve WordPressu se autentizace provádí standardně přes přihlašovací stránku umístěnou na *wp-login.php* nebo před spuštěním metody v XML-RPC. XML-RPC je webová služba, která se využívá při tvorbě mobilních aplikací, pluginů nebo šablon a vyžaduje přihlášení. Ve WordPressu běží na adrese *xmlrpc.php*. (Zhivilo, 2017)

Bezpečnostní plugin by kromě standardního způsobu přihlašování na *wp-login.php* měl proti útoku hrubou silou chránit také XML-RPC. Na následujícím obrázku (Obrázek 4) je porovnání počtu útoků provedených od 16.01.2017 do 29.01.2017 na oba zmíněné způsoby autentizace. Je patrné, že zabezpečení obou způsobů autentizace je stejně důležité.

Obrázek 4 - Celkový počet útoků hrubou silou na standardní přihlašovací stránku a XML-RPC



Zdroj: (Maunder, 2017)

Pro útok na XML-RCP bude použit skript dostupný na GitHubu (Zhivilo, 2017). Konkrétně se jedná o skript *wp_find_password.rb*. K provedení útoku pod jinou IP adresou byl opět použit program Tor společně s nástrojem ProxyChains, který je již v Kali Linux nainstalován (Hoi, 2017). Útok se provádí následujícím příkazem:

proxychains ruby wp_find_password.rb --url http://domena.cz/xmlrcp.php --wordlist wordlist.txt --username wordpress1

V další části této kapitoly budou analyzovány jednotlivé bezpečnostní pluginy a zjištěné výsledky interpretovány. Zkoumána bude existence a funkčnost následujících funkcí v pluginu:

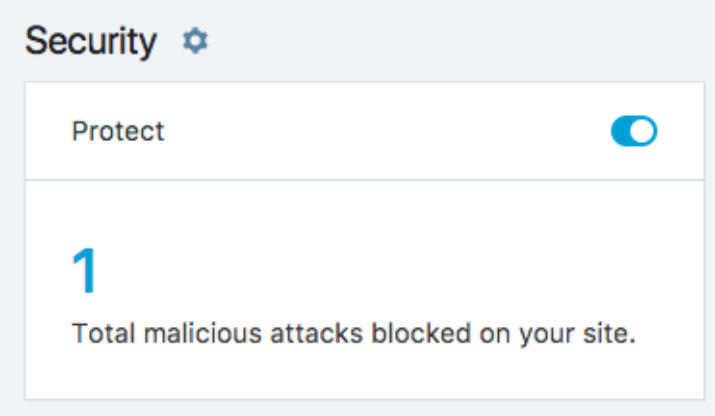
- Blokování uživatele či IP adresy, z které je útok na přihlašovací formulář prováděn.
- Zaznamenání a přehledné zobrazení chybných přihlášení do administrace a blokovaných uživatelů, resp. IP adres.
- Možnost nastavení počtu pokusů, po kterých je IP adresa zablokována, délka její blokace a změna URL přihlašovacího formuláře.
- Vynucení silného uživatelského hesla.
- Možnost nastavení captchy a dvoufázového přihlášení u přihlašovacího formuláře.
- Zabránění útoku hrubou silou na XML-RPC.

Jetpack

Plugin Jetpack nevynucuje nastavení silného hesla, neumožňuje nastavit počet povolených pokusů o přihlášení a ani nastavit délku blokace. V rámci ochrany proti útoku hrubou silou pouze blokuje IP adresy na dobu, která je podle (Jetpack) dána několika blíže nespecifikovanými faktory. Zároveň umožnuje formulář rozšířit o ověření přes mobil nebo zcela nahradit přihlašováním přes Google. Útok hrubou silou blokuje plugin ovšem pouze na přihlašovací formulář. XML-RPC potřebuje totiž ke komunikaci s webovou aplikací, která běží na wordpress.com.

Během provádění útoku bylo zjištěno, že plugin nedostatečně chrání webové stránky proti útoku hrubou silou. Přestože se nepodařilo získat heslo pomocí nástroje WPScan, tak skript *wp_find_password.rb* využívající XML-RPC byl úspěšný. Plugin by měl blokovat i IP adresy, z kterých se provede několik neúspěšných pokusů o přihlášení přes XML-RPC a zároveň lépe informovat administrátora o neočekávaných událostech na webu. Momentálně plugin na nástěnce pouze zobrazuje počet útoků, které se podařilo zablokovat (Obrázek 5).

Obrázek 5 - Zablokované útoky na webové stránky chráněné pluginem Jetpack



Zdroj: vlastní zpracování

Wordfence Security

Plugin Wordfence Security umožňuje nastavit blokaci IP adresy od 1 do 500 pokusů po dobu 5 minut až 1 dne. Zároveň umožňuje nastavit blokování IP adresy ihned po zadání špatného uživatelského jména. Vynucení silného uživatelského hesla lze nastavit pro všechny uživatele nebo jen pro administrátory a vydavatele článků (uživatelská role publishers).

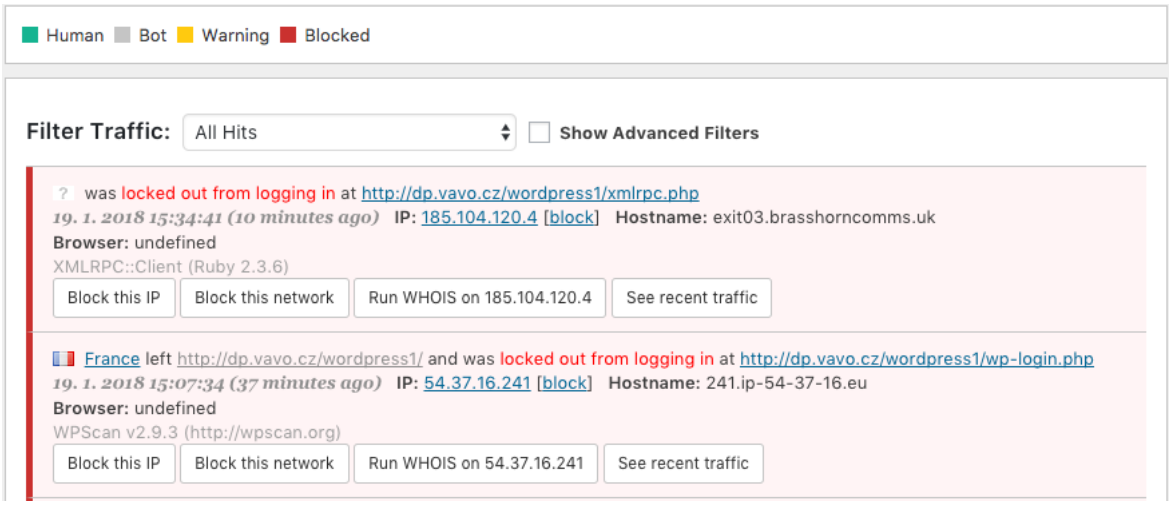
Oproti defaultní kontrole síly hesla ve WordPressu hlídá tento plugin, zda heslo obsahuje kombinaci velkých a malých písmen a zároveň kombinaci písmen a číslic.

Plugin neobsahuje možnost přidání captchy do přihlašovacího formuláře. Jediné, co umožňuje je dvoufázové přihlášení prostřednictvím telefonu, které je ovšem dostupné pouze v placené verzi pluginu.

Wordfence na svém blogu (Maunder, 2015) uvádí, že plugin kromě útoku na přihlašovací formulář chrání také útok na XML-RPC. Při provádění útoku pomocí nástroje WPScan a skriptu *wp_find_password.rb* bylo zjištěno, že kromě zablokování IP adresy kvůli překročení počtu pokusů o přihlášení, plugin ihned blokuje i IP adresy, z kterých už byl někdy nějaký útok prováděn na jiné webové stránky.

Všechna chybná přihlášení jsou viditelná na nástěnce pluginu. Plugin také obsahuje funkci Live Traffic, kde jsou záznamy o přístupu k jednotlivým souborům z prohlížeče. Zaznamenán byl opakovaný přístup na soubor *wp-login.php* a *xmlrpc.php*. Na dalším obrázku (Obrázek 6) je ukázka záznamů funkce Live Traffic.

Obrázek 6 - Funkce Live Traffic v pluginu Wordfence



Zdroj: vlastní zpracování

Plugin nabízí velice přehledné monitorování neočekávaných událostí v aplikaci. Především sledování požadavků v reálném čase je velice užitečné, ovšem je nutné počítat se zpomalením aplikace a zaplněním databáze, a proto je nutné tuto funkci zapínat pouze, když je potřeba. Jediná nevýhoda pluginu je absence možnosti zcela vypnout podporu XML-RPC.

iThemes Security

Plugin iThemes umožňuje nastavit povolený počet pokusů k přihlášení a délku blokace IP adresy na libovolnou hodnotu. Defaultně je nastaveno pět pokusů na jednu IP adresu, deset pokusů pro jednoho uživatele a délka blokace 5 minut. Nastavení captchy a dvoufázové autentizace lze pouze v placené verzi. Plugin zároveň vynucuje silné heslo pro vybranou uživatelskou roli. Oproti defaultní kontrole síly hesla vynucuje iThemes Security zadání takového hesla, které je při nastavení ve WordPressu zobrazeno jako „Bezpečné“. Určitou kombinaci znaků plugin nevynucuje. Monitorování neočekávaných událostí v podobě neplatného pokusu o přihlášení je dostačující.

Plugin iThemes v rámci ochrany proti útoku hrubou silou umožňuje vypnout zcela podporu pro XML-RPC. V případě, že je na webu nainstalován plugin Jetpack nebo je XML-RPC využíváno mobilní aplikací, tak plugin umožňuje blokovat požadavky, které obsahují více pokusů o přihlášení.

Při analýze efektivnosti pluginu iThemes Security byla podpora XML-RPC ponechána zapnuta a testováno bylo blokování více pokusů o přihlášení. Plugin zaznamenal útok na *wp-login.php* i *xmlrpc.php*. IP adresu i uživatele na 5 minut zablokoval. Záznam neočekávaných událostí je na následujícím obrázku (Obrázek 7). Výhodou tohoto pluginu je fakt, že stejně jako Wordfence automaticky blokuje IP adresy, z kterých byl prováděn útok na jiné webové stránky. Plugin dále umožňuje změnit URL standardního přihlašovacího formuláře. WPScan v takovém případě není schopen útok provést.

Obrázek 7 - Záznam neočekávaných událostí na webových stránkách v pluginu iThemes Security

Function	Priority	Time	Host	User	URL	Referrer	Data
Invalid Login Attempt	5	2018-01-19 15:17:25	145.239.91.37	wordpress2			Details
Host or User Lockout	10	2018-01-19 15:17:25	145.239.91.37				Details
Invalid Login Attempt	5	2018-01-19 15:17:23	145.239.91.37	wordpress2			Details

Zdroj: vlastní zpracování

All In One WP Security & Firewall

All In One WP Security umožňuje nastavit počet pokusů před zablokováním IP adresy, časovou délku blokace, okamžité zablokování při zadání špatného uživatelského jména

a nastavit obecnou chybovou hlášku, která útočníkovi stíží hádání. Dále plugin umožňuje přejmenovat přihlašovací stránku, nastavit captchu nebo tzv. Honeypot, což je skryté formulářové pole, které běžný uživatel nevyplní, ovšem robot ano. Vynucení silného uživatelského hesla však v pluginu All In One WP Security nastavit nelze.

Oproti ostatním analyzovaným pluginům má All In One WP Security funkci, která útok hrubou silou zastavuje ještě před spuštěním PHP kódu, a tedy zpomalením aplikace. Do souboru *.htaccess* je přidána kontrola cookies. Zadáním správné URL se cookies nastaví na požadovanou hodnotu, v htaccessu se hodnota ověří a uživatel je přesměrován na přihlašovací stránku. V opačném případě se uživateli přihlašovací stránka nezobrazí.

XML-RPC lze kompletně v pluginu vypnout nebo ponechat zapnuté a spoléhat se na zablokování IP adresy při několika neúspěšných pokusech o přihlášení. Plugin zároveň umožňuje vypnout pouze funkcionalitu Pingback v XML-RPC, která se používá k provádění DDoS útoku.

Při provádění útoku hrubou silou byla IP adresa zablokována u přihlašovacího formuláře i XML-RPC. Blokováním IP adres, z kterých byl již dříve prováděn nějaký útok na jiné webové stránky, plugin nedisponuje. Při zapnutí captchy nebo ochrany založené na cookies se útok pomocí nástroje WPScan nepodařilo provést.

All In One WP Security na nástěnce zobrazuje zablokované IP adresy, jejichž podrobnější přehled je na samostatné stránce. Neúspěšné pokusy o přihlášení jsou také na samostatné stránce.

Sucuri Security

Bezplatná verze pluginu Sucuri Security neobsahuje ochranu proti útoku hrubou silou v podobě blokování IP adres. Plugin pouze informuje uživatele o tom, že je útok prováděn prostřednictvím e-mailu. Defaultně je nastaveno 30 neúspěšných pokusů o přihlášení, na základě, kterých je pluginem určeno, že se jedná o útok hrubou silou. Počet pokusů lze nastavit od 30 do 480. Záznamy o neočekávaných událostech v aplikaci jsou na nástěnce. Neúspěšné pokusy o přihlášení nebo blokovaní uživatelé jsou detailněji popsány na samostatné stránce. Během útoku na webové stránky s bezplatnou verzí pluginu byl odeslán e-mail pouze při útoku na přihlašovací formulář pomocí nástroje WPScan. Při útoku na XML-RPC e-mail odeslán nebyl, avšak neúspěšné přihlášení zaznamenáno bylo.

Během analýzy efektivnosti pluginu Sucuri Security z hlediska obrany proti útoku hrubou silou, byla zakoupena placená verze, která umožňuje nastavit firewall. Díky tomu, že veškerá komunikace probíhá přes firewall od Sucuri a až následně je požadavek poslán na server, kde jsou webové stránky hostovány, tak dochází k okamžitému zablokování útoku pomocí nástroje WPScan. U útoku na XML-RPC se podaří provést pouze dva pokusy o přihlášení a následně dojde k zablokování IP adresy. Placená verze navíc umožňuje nastavit captchu nebo dvoufázovou autentizaci na libovolný zdroj. Monitorování aplikace je přímo v administraci na webových stránkách Sucuri. Na následujícím obrázku (Obrázek 8) jsou červeným kruhem označeny zablokované IP adresy. Na webu Sucuri je také přehled požadavků v reálném čase a další velice přehledné reporty o událostech na webových stránkách.

Obrázek 8 - Přehled požadavků v reálném čase na webové stránky chráněné firewallem od Sucuri

IP Address	Resource Path	HTTP User-Agent	Date/Time
64.78.149.164	GET /.well-known/acme-challenge/tVj7H40R-0M-huowmg4ZiGjDeeJoOFdom4-ZwPlw4No	Mozilla/5.0 (compatible; Let's Encrypt validation server; +https://www.letsencrypt.org)	22/Jan/2018:08:02:39 -0500
185.220.101.41	GET /	WPScan v2.9.3 (http://wpscan.org)	22/Jan/2018:08:37:09 -0500
185.220.101.41	GET /	WPScan v2.9.3 (http://wpscan.org)	22/Jan/2018:08:37:49 -0500
185.220.101.41	GET /	WPScan v2.9.3 (http://wpscan.org)	22/Jan/2018:08:37:50 -0500
176.10.104.240	POST /xmlrpc.php	XMLRPC::Client (Ruby 2.3.6)	22/Jan/2018:09:09:41 -0500

Zdroj: vlastní zpracování

Na webových stránkách Sucuri, lze provádět více nastavení než v samotném pluginu. Počet povolených neúspěšných pokusů o přihlášení, délku blokace IP adresy nebo vynucení silného uživatelského hesla nejde ovšem nastavit ani v pluginu ani v administraci na webu Sucuri. Při vzájemné komparaci bude zohledněn fakt, že samotný firewall dostupný v placené verzi vyžaduje složitou konfiguraci a zároveň není zcela součástí samotného pluginu. Firewall od Sucuri je totiž službou, která nevyžaduje existenci pluginu nebo samotného WordPressu.

Acunetix WP Security

Plugin Acunetix WP Security nemá funkci, která by blokovala IP adresy při velkém počtu neúspěšných pokusů o přihlášení a ani jinou funkci, která by webovou aplikaci chránila proti útoku hrubou silou. Jediné, co se dá považovat za prevenci proti útoku hrubou silou, avšak nedostatečnou, je možnost odstranit chybové zprávy u přihlašovacího formuláře. Podobně

jako plugin Wordfence obsahuje Acunetix WP Security funkci Live Traffic, kde lze dohledat několikanásobný přístup k souboru *wp-login.php* a *xmlrpc.php*. Administrátor tak může alespoň zjistit, že byl nějaký útok prováděn a provést příslušná opatření.

BulletProof Security

Plugin BulletProof Security kromě počtu neúspěšných pokusů a časovou délku zablokování uživatelského účtu umožňuje nastavit obsah chybové zprávy a zda se má či nemá zobrazovat počet zbývajících pokusů. Dále umožňuje nastavit captchu v přihlašovacím formuláři. Vynucení silného hesla nebo jiné funkce sloužící k ochraně proti útoku hrubou silou plugin neobsahuje.

Při útoku pomocí nástroje WPScan musel být k příkazu přidán parametr *--random-agent*, protože plugin útok jinak zcela zablokoval díky zapnutému Root Folder BulletProof módu. Útok pomocí nástroje WPScan ani útok na XML-RPC plugin nezaznamenal. Jedinou možností, jak se úspěšně bránit proti útoku na XML-RPC je využít možnosti editace souboru *.htaccess* a přidat do něj kód, který ochranu obstará. V komparaci pluginů však tato možnost nebude zohledněna, protože takové zabezpečení lze provést i bez použití daného pluginu.

Na závěr bylo otestováno, zda zablokování uživatelského účtu funguje alespoň v případě manuálního hádání uživatelského hesla v přihlašovacím formuláři. Při třech neúspěšných pokusech o přihlášení byl účet úspěšně zamknut a celá událost zaznamenána, jak je patrné na následujícím obrázku (Obrázek 9).

Obrázek 9 - Zamknutý uživatelský účet v pluginu BulletProof Security

Login Status	Lock	Unlock	Delete	User ID	Username	Display Name	Email	Role	Login Time	Lockout Expires	IP Address	Hostname	Request URI
Locked	Lock	Unlock	Delete	1	wordpress11	wordpress11	vaclav.vodicka@vavo.cz	administrator	22.1.2018 18:33	22.1.2018 18:48	192.168.0.1	192.168.0.1	/wordpress11/wp-login.php

Zdroj: vlastní zpracování

Shield Security for WordPress

Plugin Shield Security blokuje uživatelský účet ihned po prvním neúspěšném pokusu na dobu, která je specifikována v nastavení. Nastavit lze hodnoty od 0 až do neomezeného počtu sekund, přičemž 0 znamená vypnuté blokování. Plugin dále umožňuje přejmenovat přihlašovací stránku, nastavit captchu nebo dvoufázovou autentizaci přes Google či e-mail. Jako captchu lze nastavit reCAPTCHU od Googlu nebo tzv. G.A.S.P., což je dynamicky generované zaškrtávací pole JavaScriptem, které roboti nemohou zaškrtnout. Plugin

zabezpečení přihlašování posouvá ještě dál, protože umožňuje autentizaci pomocí zařízení Yubikey. Vynucení silného hesla nepodporuje.

XML-RPC je defaultně v pluginu povoleno, avšak lze ho zcela vypnout. V případě kompletního vypnutí podpory XML-RPC se sice nepodařilo heslo uhodnout, protože soubor *xmlrpc.php* vrátil stavový kód -32601, ale skript mohl dále provádět útok a zpomalovat tak webovou aplikaci. Plugin tedy nesprávně uvádí, že dojde k vypnutí celého XML-RPC systému.

Bez zapnuté captchy nebo dvoufázové autentizace je jedinou ochranou proti útoku hrubou silou blokování přihlašování po specifikovanou dobu. Problém však je, že nástroj WPScan zablokován není a může tedy hádání stále provádět. Existuje určitá pravděpodobnost, kdy se při hádání hesla může WPScan trefit do doby, kdy je účet zrovna odblokován. Zároveň je možné vytvořit skript, který bude heslo zkoušet po uplynutí určitého časového intervalu. Tato ochrana je tudíž nedostatečná a je nutné využít např. zmiňovaný G.A.S.P., který přihlašovací formulář, jak bylo zjištěno, chrání proti útoku pomocí nástroje WPScan dostatečně.

Nevýhodou pluginu je monitorování aplikace a informování administrátora o neočekávaných událostech, které není zdaleka tak dobré jako u některých konkurenčních pluginů. Jediným způsobem, jak zjistit, že byl útok hrubou silou prováděn, je navštívení stránky s auditovacím logem, kde jsou všechny záznamy o neúspěšných pokusech o přihlášení a dalších událostech ve webové aplikaci. E-mailové upozornění o útoku či základní přehled na nástěnce plugin nepodporuje.

NinjaFirewall a NinjaScanner

Ochranu proti útoku hrubou silou je nutné v pluginu NinjaFirewall zapnout. Nastavit lze buď ochranu pomocí další autentizace, nebo captchy. Přidání další autentizace k ochraně přihlašovacího formuláře je nesmyslné, protože je nový formulář náchylný na útok hrubou silou stejně jako chráněný formulář. Vhodnou ochranou proti útoku hrubou silou je využití možnosti přidat do přihlašovacího formuláře captchu.

Plugin dále obsahuje funkci, která ihned rozpozná a zablokuje roboty nebo skripty, které provádí útok hrubou silou. Všechna zmíněná opatření lze nastavit i pro XML-RPC. V zásadách firewallu lze také podporu XML-RPC zcela vypnout. Vynucení silného hesla či jiná preventivní opatření proti prolomení autentizace plugin neobsahuje.

Snahou bylo otestovat, zda plugin blokuje útok hrubou silou bez použití captchy nebo dalšího přihlašovacího formuláře. V nastavení je ovšem nutné vybrat jedno ze zmíněných zabezpečení. Útok pomocí nástroje WPScan se proto nepodařilo provést. Při zvolené možnosti, která aplikuje veškerou ochranu na XML-RPC se nepodařil ani útok pomocí skriptu *wp_find_password.rb*. Plugin NinjaFirewall jako jediný umožňuje přidat captchu nebo další autentizaci na soubor *xmlrpc.php*. V případě, že administrátor potřebuje XML-RPC využít ke komunikaci s mobilní aplikací či jiným pluginem, tak musí celou ochranu XML-RPC vypnout a jak radí (NinTechNet, 2018), tak zapnout blokaci metody *system.multicall*. Jak se však ukázalo, tak toto nastavení nepomohlo a útok se podařilo provést.

Plugin do svého logu, který je dostupný v administraci WordPressu, zaznamenává všechny blokované požadavky. Dále plugin obsahuje tzv. Live Log, kde jsou požadavky na webovou aplikaci v reálném čase. O neočekávaných událostech je zároveň administrátor informován e-mailem.

SecuPress

V pluginu SecuPress lze nastavit libovolnou délku blokace IP adresy po překročení nastaveného počtu pokusů o přihlášení. Dále lze zapnout, aby byla IP adresa zablokována ihned po odeslání neplatného uživatelského jména. Přihlašovací stránku je možné v pluginu přejmenovat. Plugin dále umožňuje nastavit captchu, vypnout podporu XML-RPC nebo pouze zapnout blokaci IP adres, které se několikrát neúspěšně pokusí přes XML-RPC přihlásit. V placené verzi pluginu lze navíc nastavit dvoufázovou autentizaci, vynutit silné uživatelské heslo a nastavit dobu, po které si uživatel musí heslo změnit. Při zapnutém vynucení silného hesla však nebyla vypozorována žádná změna.

Útok pomocí nástroje WPScan plugin zaznamenal a zablokoval IP adresu po nastaveném počtu pokusů. Při testování zabezpečení XML-RPC byla zapnuta funkce, která zabraňuje několikanásobnému pokusu o přihlášení. Bylo provedeno více než 50 pokusů o přihlášení a IP adresa zablokována nebyla. Útok proběhl úspěšně a heslo se podařilo získat i přes zapnutou ochranu XML-RPC. Pouze při zcela vypnuté podpoře XML-RPC se útok nepodařil.

O nefunkčním zabezpečení XML-RPC byla informována podpora, která vysvětlila, že skript *wp_find_password.rb* neprovádí několikanásobný pokus o přihlášení, ale pouze jeden pokus

několikrát za sebou. Dále doporučila zapnout ochranu „Anti Brute-Force“, která blokuje IP adresu uživatele v případě, že odešle 10 požadavků na server během jedné sekundy. Toto nastavení je z neznámého důvodu v jiné části pluginu než ostatní zabezpečení proti útoku hrubou silou, a často nemusí být administrátorem nastavené. Navíc je funkce dostupná pouze v placené verzi pluginu. I přes zapnutí této ochrany se útok hrubou silou skriptem *wp_find_password.rb* podařilo provést.

Neočekávané události jsou zaznamenávány do logu, který je dostupný z administrace WordPressu, je přehledný a poskytuje dostatečné informace. V placené verzi lze nastavit také e-mailová upozornění.

4.2.3 Malware

Do webových aplikací bude vložen malware, tedy soubor, který umožní vzdáleně ovládat server. V případě, že je tento soubor nahrán na webový server a spustitelný jako např. PHP skript, tak je podle (US-CERT, 2015) označován jako web shell. Pro tento útok byl vybrán web shell dostupný na GitHubu (GitHub, 2017). Jedná se o jeden PHP soubor a uživatelské rozhraní je na následujícím obrázku (Obrázek 10).

Obrázek 10 - Uživatelské rozhraní použitého web shellu

Zdroj: vlastní zpracování

Do systému WordPress lze vložit malware při instalaci pluginu jako jeho součást pomocí nahrávání medií či úpravy stávajícího souboru v editoru šablon a pluginů. Velice častým způsobem je také využití bezpečnostní díry některého z již nainstalovaných pluginů. V případě nahrávání medií je nutné vložit skript jako obrázek se speciálně upraveným obsahem. Jednodušší variantou je úprava existujícího souboru v editoru. Vložení škodlivého

souboru, který je součástí pluginu, je výhodnější v tom ohledu, že stačí uživatele WordPressu přimět k jeho instalaci a nemuset přitom prolomit autentizaci či jiné zabezpečení.

Útočník nebo uživatel reprezentující zaměstnance firmy, který se dostane k editoru šablon může tímto způsobem vytvořit zadní vrátka do systému a později je využít k narušení chodu aplikace nebo získání citlivých dat uložených na serveru. Vhodnou úpravou soboru *functions.php* lze také například vytvořit administrátorský účet (Abela, 2014).

Některé pluginy umožňují vypnout podporu editoru šablon a pluginů. Získá-li útočník administrátorská práva nebo prolomí-li autentizaci pomocí útoku hrubou silou, může podporu editoru znovu povolit. Nejlepším řešením je vypnout editor v konfiguračním souboru WordPressu a nastavit všem souborům na serveru taková práva, která nedovolí jejich změnu z webového rozhraní. Analýza bezpečnostních pluginů z hlediska nastavení práv k souborům a adresářům je v následující kapitole.

V této kapitole budou analyzovány jednotlivé pluginy z hlediska obrany proti malwaru. Bezpečnostní plugin systému WordPress by měl umožňovat malware na webovém serveru nalézt, odstranit a informovat o něm administrátora aplikace. K provedení zmíněných kroků obsahují pluginy funkce jako je skenování změn v souborech, vrácení souboru do původního stavu či jeho smazání, zamezení spuštění škodlivého souboru a e-mailové či jiné upozornění na existenci škodlivého souboru v aplikaci.

Pro analýzu zmíněných funkcí byl malware vložen do existujícího souboru *404.php* u šablony, která není momentálně webovými stránkami využívána. Aktuální šablona je „Twenty seventeen“, a proto byla vybrána šablona „Twenty sixteen“. Soubor *404.php* je dostupný na adrese *www.domena.cz/wp-content/themes/twentysixteen/404.php*. Dále byl web shell vložen do adresáře */wp-admin/* a */wp-content/themes/twentysixteen/*, jako nový soubor.

Plugin, který obsahuje funkci pro skenování aplikace jejíž součástí je hledání škodlivých souborů, by měl zaznamenat dva nově přidané soubory a změnu v jednom existujícím souboru. Zároveň by měl o této skutečnosti upozornit administrátora a umožnit mu nápravu daného problému.

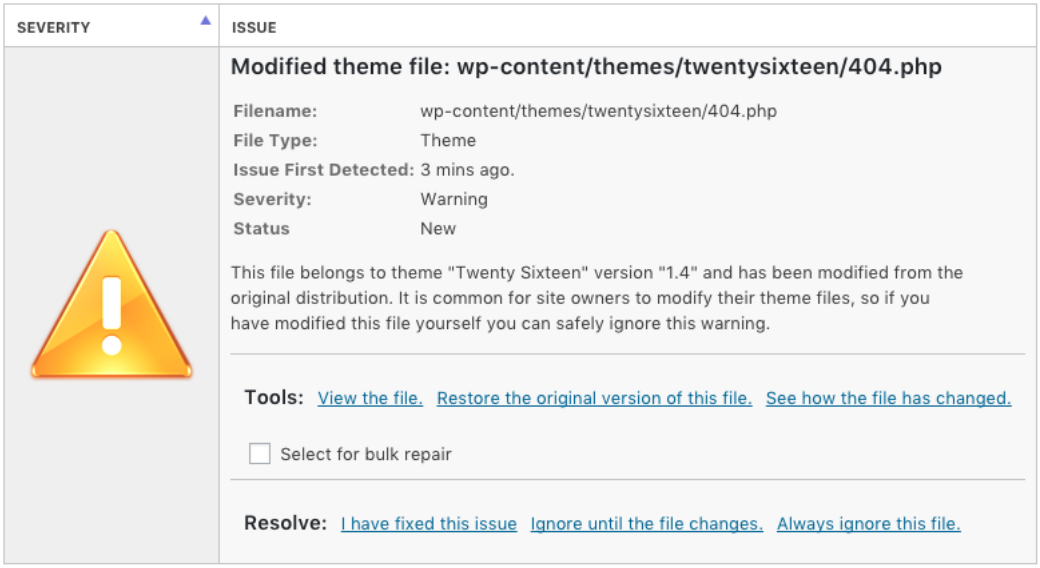
Jetpack

Plugin Jetpack neobsahuje žádnou funkci k nalezení vloženého web shellu, resp. detekci změn v souborech, a tudíž je webová aplikace proti malwaru nechráněna.

Wordfence Security

Funkce pro skenování aplikace v pluginu Wordfence Security rozeznala vložený škodlivý soubor do adresáře */wp-admin/* a umožnila jeho smazání. Změněný soubor *404.php* a nový soubor ve složce */wp-content/themes/twentysixteen/webshell.php* ovšem nezaznamenala. Důvodem je fakt, že funkce defaultně neprochází soubory v šablonách. Po dodatečném nastavení kontroly souborů v šablonách rozeznal plugin navíc pouze změněný obsah v souboru *404.php*. Skener porovnává aktuální soubor s originálním souborem šablony, avšak nově přidané soubory opomíjí. Vložení zadních vrátek v podobě web shellu proběhlo u Wordfence Security pluginu úspěšně. Kladně hodnocena je možnost vyřešení daného problému a zpracování chyby v uživatelském rozhraní, které je k nahlédnutí na následujícím obrázku (Obrázek 11).

Obrázek 11 - Rozpoznaný škodlivý soubor v pluginu Wordfence Security



Zdroj: vlastní zpracování

iThemes Security

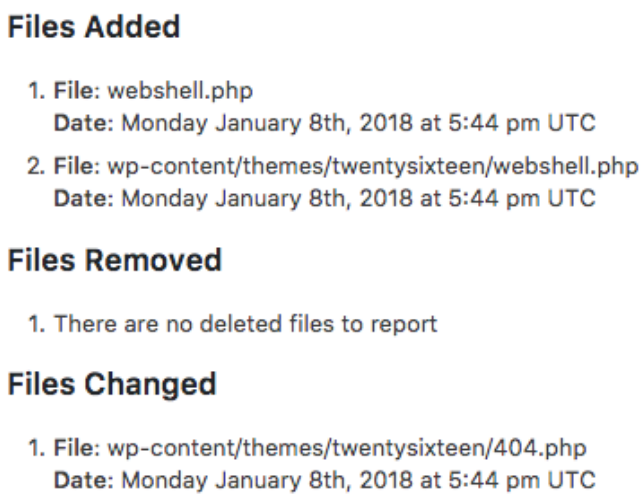
Plugin iThemes Security defaultně umožňuje skenování pouze domovské stránky prostřednictvím online nástroje Sucuri SiteCheck (Sucuri, 2016), který žádný malware na hlavní stránce webu nenašel. V nastavení lze ovšem zapnout detekci změn v souborech

a skenování aplikace. Funkce porovnává všechny aktuální soubory s těmi, které existovaly při posledním provedení skenu. Po instalaci pluginu je tedy nutné funkci povolit a provést sken, který uloží informace o aktuálních souborech do databáze.

Plugin úspěšně rozpoznal dva přidané soubory a jeden upravený, jak je vidět na dalším obrázku (Obrázek 12). Nevýhodou je skutečnost, že v případě smazání nebo přidání pluginu obsahuje report spoustu souborů a stává se nepřehledný. Administrátor, který zapomene vyloučit složku s pluginy ze skenování může škodlivý soubor snadno přehlédnout. Další nevýhodou je absence jakékoliv nápravy problému.

Dále bylo zjištěno, že plugin u kontroly změn v souboru porovnává jeho velikost a datum poslední změny. Pokud se útočníkovi podaří vložit do WordPressu web shell, jako nový soubor, tak může kontrolu snadno obejít. Stačí do web shellu zadat linuxový příkaz, který zjistí velikost a čas změny souboru, který chceme změnit. Pomocí příkazu *touch* se následně souboru nastaví původní datum změny. Podaří-li se škodlivý skript vložit do souboru tak, aby měl stejnou velikost, pak infikovaný soubor nebude pluginem rozeznán. Původně vložený web shell se nakonec smaže.

Obrázek 12 - Detekované změny v souborech u pluginu iThemes Security



Zdroj: vlastní zpracování

All In One WP Security & Firewall

Skenování aplikace v pluginu All In One WP Security & Firewall funguje na stejném principu jako u pluginu iThemes Security. Jediným rozdílem je skutečnost, že není potřeba skenování povolit. Stejně jako v předešlém případě i tento plugin rozeznal dva přidané

a jeden upravený soubor. Při analýze tohoto pluginu byl zároveň vložen škodlivý kód do souboru *404.php* tak, aby velikost souboru byla stejná jako původní a pomocí web shellu bylo nastaveno původní datum změny. Plugin v takovém případě změnu v souboru nerozeznal.

Sucuri Security

V pluginu Sucuri Security je skenování změn v souborech možné provést pouze prostřednictvím funkce WordPress Integrity Diff Utility, která je defaultně vypnutá a kontroluje pouze změnu souborů oproti čisté instalaci WordPressu. Složku s pluginy a šablonami ignoruje a nelze ji do kontroly zařadit. Plugin rozeznal pouze vložený web shell do hlavního adresáře webových stránek. V adresáři *wp-content* lze blokovat spuštění PHP souboru v prohlížeči a zamezit tak spuštění web shellu útočníkem. Při zapnutí této možnosti se však web shell podařilo stejně spustit.

Acunetix WP Security

Skenování souborů pluginem Acunetix WP Security se nepodařilo provést. Plugin nebyl schopen ze serveru stáhnout JSON soubor nutný pro provedení skenu. Pravděpodobně nebyla doposud do pluginu implementována podpora pro nejnovější verzi WordPressu 4.9.1.

BulletProof Security

Plugin BulletProof Security obsahuje funkci pro skenování aplikace, která hledá škodlivý kód uložený v databázi či souborech. Web shell, který byl do stránek vložen, ovšem nezaznamenala.

Shield Security

Plugin Shield Security obsahuje funkci pro skenování souborů ve složce se zdrojovými soubory WordPressu. Plugin nehledá škodlivé soubory ve složce s pluginy a šablonami. V případě nalezení škodlivého souboru pošle administrátorovi e-mail. Vložený web shell nerozeznal plugin ani v jednom adresáři. Zároveň plugin v uživatelském rozhraní neumožňuje sken přímo spustit. Místo toho je nutné zadat speciální parametr do URL nebo počkat na automatické spuštění.

NinjaFirewall a NinjaScanner

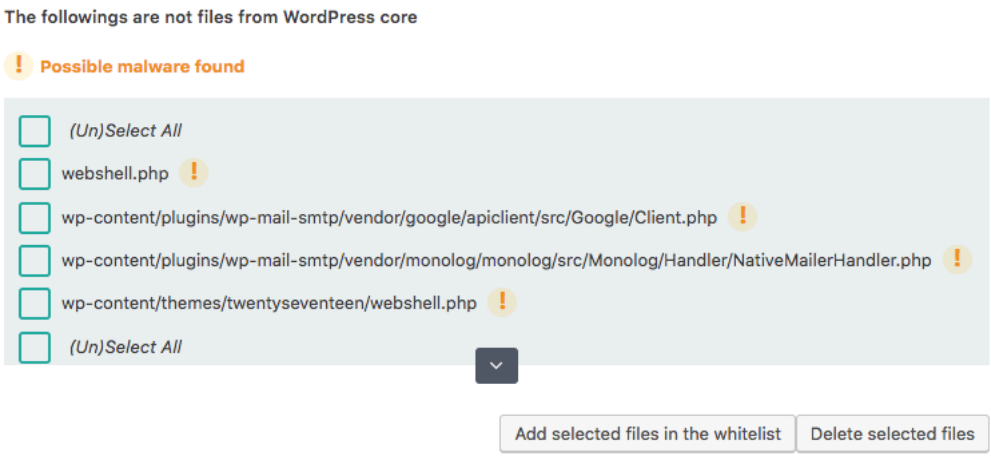
Plugin NinjaFirewall a NinjaScanner obsahuje funkci pro skenování změn v souborech. Před prvním skenováním je nutné provést inicializační sken, aby se uložily informace

o aktuálních adresářích a souborech. Plugin rozeznal oba vložené web shelly a upravený soubor *404.php*. Nevýhoda této funkce je dána tím, že se informace o souborech a adresářích ukládají do PHP souboru na serveru a pomocí web shellu je možné tyto informace upravit. Další nevýhodou je absence jakékoliv nápravy zjištěné změny v souborech. Při zvoleném Full WAF módu by se dalo očekávat, že web shell nepůjde spustit, ovšem firewall nerozeznal skript jako škodlivý, a proto se ho podařilo bez problému spustit.

SecuPress

Pomocí skenování aplikace v pluginu SecuPress byly nalezeny oba přidané soubory a změněný soubor *404.php*. Plugin umožnuje nově přidané soubory smazat a upravený soubor lze vrátit do původního stavu. Zároveň umožnují změny v souboru zobrazit. Informace o aktuálních souborech a adresářích ukládá do databáze. Obejít toto zabezpečení jako u ostatních pluginů pomocí příkazu *touch* se nepodařilo. Na následujícím obrázku (Obrázek 13) je k nahlédnutí výsledek skenování, kde je patrné, že plugin jako možný malware označil i soubory jiného pluginu.

Obrázek 13 - Rozpoznané škodlivé soubory v pluginu SecuPress



Zdroj: vlastní zpracování

4.2.4 Přístup k souborům s citlivými informacemi

Schopnost vložit malware na webové stránky je dána především bezpečnostními dírami nainstalovaných pluginů a nedostatečným nastavením práv složek a souborů. Jak již bylo zmíněno v předchozí kapitole, tak pomocí editoru šablon a pluginů je možné upravit PHP soubor tak, aby spustil libovolný škodlivý kód. Se správným nastavením práv lze docílit toho, že editování souborů z webového prohlížeče nepůjde.

Škodlivý soubor, resp. malware může být vložen skrze nedostatečně zabezpečenou webovou aplikaci, která běží společně s dalšími aplikacemi na jednom webovém serveru. Nesprávně nastavená práva k souborům a složkám vedou k tomu, že k nim malware může přistupovat i přesto, že byl vložen skrze jinou aplikaci. (Wordfence, 2016)

Při útoku na webovou aplikaci je snahou útočníka získat přístup k souborům, které obsahují citlivé údaje. Jedná se o údaje jako jsou přístupy do databáze, záznamy o chybách, konfigurační soubory a soubory s informacemi o verzi WordPressu a použitých pluginech. Díky znalosti verze WordPressu, použitých pluginech a šablonách může útočník využít zranitelnosti, které jsou o nich veřejně známé. Soubory ve WordPressu obsahující citlivé informace jsou následující (Weiss, 2012):

- wp-config.php,
- readme.html,
- license.txt,
- install.php,
- error.log nebo jiný soubor, kam se zaznamenávají chyby,
- .htaccess.

Dalším problémem souvisejícím s právy souborů a složek je tzv. Directory Browsing, tedy prohlížení souborů ve složce prostřednictvím webového prohlížeče. Útočník tak může získat lepší přehled o funkčnosti aplikace a využít získané informace k provedení útoku. Uživatelé mohou ve WordPressu také nahrát soubory s citlivými údaji a z uživatelského rozhraní na ně záměrně neodkazovat. Ve WordPressu se soubory ukládají defaultně do složky *wp-content/uploads*. Díky povolenému Directory Browsing může útočník citlivé údaje snadno získat.

K zjištění složek, u kterých je možné procházet soubory z prohlížeče, byl kromě nástroje WPScan použit také nástroj OWASP ZAP, který kromě zmíněného Directory Browsing v rámci skenování aplikace dokáže nalézt i další zranitelnosti v aplikaci. OWASP ZAP zjistil, že v čisté instalaci WordPressu, tzn. bez žádného pluginu a nainstalované šablony, existuje devět složek (Obrázek 14), ve kterých je Directory Browsing povolené. Nástroj WPScan zjistil Directory Browsing u *wp-includes* a *wp-content/uploads*. Průnikem je celkem 10 složek, ve kterých je procházení souborů ve složce z prohlížeče povolené.

Obrázek 14 - Directory Browsing v čisté instalaci WordPressu

Zdroj: vlastní zpracování

Dále budou nastavena neomezená práva, tj. 777, u souboru *wp-config.php* a *.htaccess*. Stejná práva budou nastavena také u složky *wp-content*, *wp-admin*, *wp-includes*, *wp-content/themes* a *wp-content/plugins*. Správná práva pro složky jsou 755 nebo 750 a pro soubory 644 nebo 640. Soubor *wp-config.php* by měl mít práva 440 nebo 400. (WordPress)

Schopnost pluginu zabránit zjištění verze WordPressu nebo nainstalovaných pluginů a šablonách, bude analyzována pomocí nástroje WPScan. Tento nástroj hledá verzi systému v souboru *readme.html*, v meta tagu a na jiných místech ve webové aplikaci. Zároveň dokáže nalézt nainstalované pluginy a šablony, u kterých vypíše jejich zranitelnosti, jsou-li tyto zranitelnosti uloženy v databázi WPScanu.

Bezpečnostní plugin systému WordPress by měl rozeznal nesprávně nastavená práva u souborů či složek a umožnit jejich nápravu. Dále by měl umožnit vypnout podporu editoru šablon a pluginů a zamezit přístup k souborům s citlivými údaji. Nakonec by měl zamezit zjištění podrobnějších informací o WordPressu, pluginech či šablonách a zamezit procházení souborů ve všech složkách ve webovém prohlížeči. Během analýzy však bylo zjištěno, že žádný z vybraných pluginů nemá funkci, která by zabránila zjištění informací o nainstalovaných pluginech nebo šablonách.

V následujícím výčtu jsou jednotlivá zabezpečení seřazena podle důležitosti a jejichž existence a efektivnost byla analyzována ve vybraných pluginech:

1. Práva souborů a složek – zjištění nedostatečných práv u souborů a složek a jejich náprava.
2. Directory Browsing – zamezení procházení souborů ve složkách skrze prohlížeč.
3. Vypnutí editoru – vypnutí podpory editoru šablon a pluginů.

4. Verze WordPressu – ochrana proti zjištění verze WordPressu.

V následující tabulce (Tabulka 2) jsou zjištěné poznatky, které jsou bodově ohodnoceny, přičemž 0 znamená žádná ochrana a 10 existence plně funkční ochrany.

Tabulka 2 - Zjištěné hodnoty při analýze zabezpečení souborů a složek

Název pluginu	**Práva souborů a složek**	**Directory Browsing**	**Vypnutí editoru**	**Verze WordPressu**
Jetpack	0	0	0	0
Wordfence Security	0	0	0	5
iThemes Security	6	10	10	5
All In One WP Security & Firewall	8	10	10	5
Sucuri Security	0	0	10	0
Acunetix WP Security	7	10	0	5
BulletProof Security	6	10	0	5
Shield Security for WordPress	0	0	10	5
NinjaFirewall a NinjaScanner	0	0	10	0
SecuPress	8	10	10	5

Zdroj: vlastní zpracování

U ochrany proti zjištění verze WordPressu bylo uděleno pouze 5 bodů u pluginů, které ochranu sice obsahovaly a snížilo se množství způsobů, jak verzi zjistit, ale nástroji WPScan se podařilo verzi zjistit ze struktury a velikosti souborů.

V případě kontroly a nápravy práv souborů a složek bylo nutné udělit jen část z maximálního počtu bodů. U pluginu iThemes Security a BulletProof Security bylo uděleno jen 6 bodů, protože oba pluginy neumožňují špatná práva přenastavit. U pluginu All In One WP Security a SecuPress bylo uděleno 8 bodů, protože změnu práv umožňují, ale u soubor *wp-config.php* vyžadují práva 644 nebo 600 místo 440 nebo 400. Pluginu Acunetix WP Security bylo uděleno pouze 7 bodů, protože stejně jako v přechozím případě, nastavuje špatná práva u souboru *wp-config.php* a navíc nekontroluje práva u adresáře *wp-content/themes* a *wp-content/plugins*.

4.2.5 SQL Injection

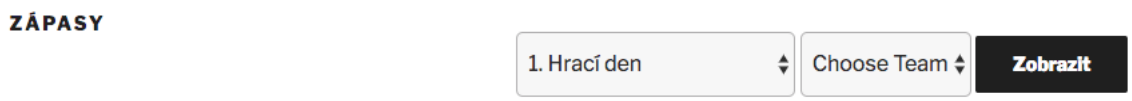
WordPress verze 4.9.1 nemá v době psaní této diplomové práce žádnou nalezenou zranitelnost, která by umožňovala provést SQL Injection. K tomu, aby se analyzovala efektivnost vybraných pluginů z hlediska obrany proti SQL injektáži, byl vybrán plugin, který takovou zranitelnost obsahuje. V databázi zranitelností byl nalezen plugin LeagueManager, který má více jak 3 tis. aktivních instalací a přes 150 tis. stažení (WordPress). Verze tohoto pluginu nižší nebo rovna 3.9.1.1 umožňuje provést útok na datové úložiště pomocí SQL Injection (Exploit Database, 2015). Plugin slouží k organizaci ligových utkání a jejich zobrazení na webové stránce. Nejnovější verze pluginu v době psaní této práce je 4.2-RC1.3.

Společně s odehranými zápasy je na stránce zobrazeno filtrování (Obrázek 15), které na základě zvoleného hracího dne a týmu zobrazí relevantní zápasy. Formulář je odesílán metodou *GET*, a tudíž se zvolené parametry zobrazí v URL, která vypadá následovně:

http://domena.cz/zapasy/?season=1&league_id=1&match_day=1&team_id=1.

Nálezce zranitelnosti (Exploit Database, 2015) uvádí, že právě parametr *league_id* je nedostatečně ošetřený a umožňuje provést útok do databáze. U zobrazení konkrétního zápasu je přidán do URL parametr *match*, který je na SQL Injection také náchylný.

Obrázek 15 - Formulář vygenerovaný pluginem LeagueManager



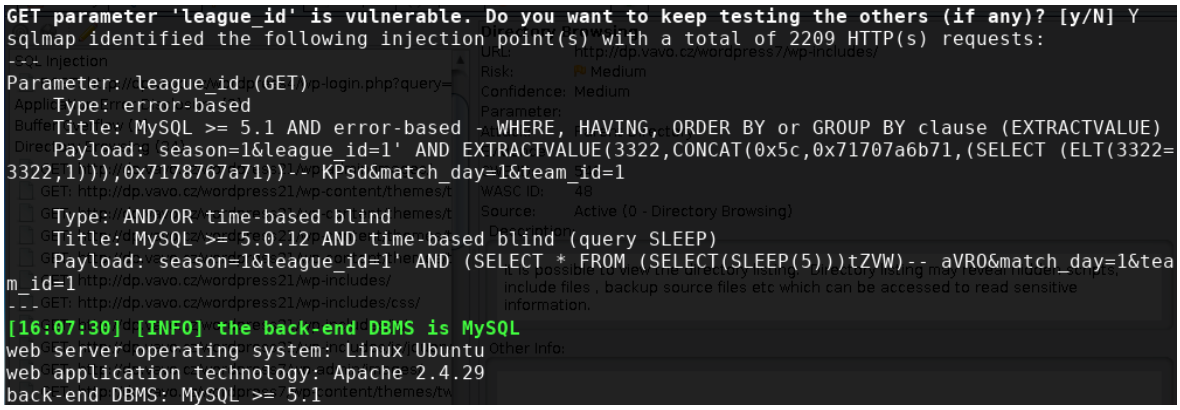
Zdroj: vlastní zpracování

Existence zranitelnosti byla ověřena pomocí nástroje *sqlmap*, který je dostupný v Kali Linux. Jedná se o penetrační nástroj, který aplikováním několika škodlivých kódů dokáže detekovat přítomnost zranitelnosti (sqlmap). Zároveň dokáže detekovat tabulky, uživatele a další informace o databázovém serveru (Stampar, 2017). Příkaz pro nalezení zranitelnosti pomocí nástroje *sqlmap* je následující:

sqlmap --url --dbms=MySQL --level 5 --risk 3 -p league_id
http://domena.cz/zapasy/?season=1&league_id=1&match_day=1&team_id=1 .

Nástroj úspěšně nalezl zranitelnost v parametru *league_id*, který lze využít k SQL Injection útoku. Na následujícím obrázku (Obrázek 16) je výstup nástroje *sqlmap* s použitým škodlivým kódem.

Obrázek 16 - Výstup z nástroje sqlmap po úspěšném nazelení zranitelnosti



Zdroj: vlastní zpracování

Hodnota parametru *match* značí identifikační číslo zápasu v databázi. Při zadání URL s odpovídajícím ID se na stránce zobrazí soupeřící týmy a skóre zápasu. Existence zranitelnosti v parametru *match* byla ověřena manuálně. K jeho hodnotě byl přidán škodlivý kód, pomocí kterého se úspěšně podařilo získat uživatelské jméno a otisk hesla uživatele z databáze. Kód, který byl přidán je následující:

UNION SELECT 1,2,3,4,5,6,7,8,9,10,11,12,user_pass,user_login,15,16,17,18,19,20 FROM wp_users ORDER BY year DESC -- .

Vytvořit škodlivý kód bylo snadné díky znalosti názvů tabulek, sloupců a chybovým zprávám z databáze, které se v průběhu útoku zobrazovaly. Výsledek úspěšného útoku je k nahlédnutí na následujícím obrázku (Obrázek 17).

Obrázek 17 - Získání uživatelského jména a otisku hesla pomocí SQL Injection

Zdroj: vlastní zpracování

Bezpečnostní plugin systému WordPress by měl zamezit spuštění škodlivého kódu pomocí firewallu a vyžádat aktualizaci starého pluginu. Dále by měl umožnit změnit prefix názvu tabulek a zamezit zobrazování chybových zpráv z databáze.

Jetpack

Při instalaci pluginu Jetpack je současně nainstalován plugin VaultPress, který umožňuje zálohovat celou aplikaci a pravidelně skenuje aplikaci a hledá malware, škodlivý kód či jiné bezpečnostní problémy. Zjištěné problémy jsou pak zobrazeny v pluginu Jetpack. VaultPress zaznamenal nainstalovaný plugin LeagueManager a upozornil na existenci veřejně známé zranitelnosti. Změnu prefixu tabulek či potlačení chyb z databáze ani jeden z pluginů neumožňuje. Úspěšně se povedly provést oba výše uvedené útoky.

Wordfence Security

Plugin Wordfence neumožňuje vypnout chybové zprávy z databáze ani změnit prefix tabulek. Aktualizaci všech pluginů vyžaduje až po provedení skenu celé aplikace. Útok na parametr *league_id* proběhl pomocí nástroje *sqlmap* úspěšně. Vložení škodlivého kódu do parametru *match* se zprvu nepodařilo, protože firewall v pluginu Wordfence blokuje některé SQL výrazy jako *UNION* a *SELECT*. Po testování několika možných zápisů těchto výrazů se nakonec podařilo ochranu obejít zápisem: *+UNION+DISTINCT+SELECT+*. Získání uživatelského jména a otisku hesla tak bylo nakonec úspěšné.

iThemes Security

Plugin iThemes Security neumožňuje vypnout chybové zprávy z databáze a ani neupozorňuje na zastaralou verzi pluginu. Útok nástrojem sqlmap se podařil bez problému provést. Plugin iThemes Security stejně jako Wordfence Security filtruje některé SQL výrazy. Nakonec se útok úspěšně podařil provést nahrazením *UNION SELECT* za *u%6eion se%6cect.* Plugin dále umožňuje změnit prefix tabulek.

All In One WP Security & Firewall

Plugin All In One WP Security umožňuje změnit prefix tabulek, ovšem chybové zprávy nepotlačuje a ani administrátora neinformuje o zastaralém pluginu s existující zranitelností. Útok pomocí nástroje *sqlmap* plugin zablokoval. Škodlivý kód v parametru *match* byl však proveden. Při analýze bylo dále zjištěno, že při zapnuté ochraně proti zjištění uživatelského jména se škodlivý kód nepodaří spustit. Důvodem je fakt, že ve škodlivém kódu je název

tabulky, kde jsou uloženy uživatelé. Plugin celou URL vyhodnotí jako pokus o získání uživatelského jména prostřednictvím parametru *author*. Při změně tabulky, např. na tabulku *wp_usermeta*, kde jsou mimo jiné uloženy session tokeny, se dotaz úspěšně provede.

Sucuri Security

Plugin Sucuri Security nepotlačuje chybové zprávy z databáze, neumožňuje změnit prefix tabulek a ani neupozorňuje na zastaralý plugin. Zároveň nezablokoval útok nástrojem *sqlmap* a ani škodlivý kód přidaný do URL.

Acunetix WP Security

Plugin Acunetix WP Security dokáže potlačit zobrazení chybových zpráv a změnit prefix tabulek. Informaci o zastaralém či zranitelném pluginu neobsahuje. Oba útoky proběhly úspěšně a administrátor se v tomto pluginu může spolehnout pouze na funkci Live Traffic, ve které jsou zobrazeny jednotlivé útoky.

BulletProof Security

Plugin BulletProof Security umožňuje změnit prefix tabulky. Potlačit chybové zprávy z databáze nedokáže a o staré verzi pluginu administrátora neinformuje. Nástroj *sqlmap* zaznamenal existenci firewallu a injektáž parametru provedl neúspěšně. Škodlivý kód v parametru *match* byl po stejné úpravě jako u pluginu iThemes spuštěn a citlivá data se podařilo z databáze získat.

Shield Security

Plugin Shield Security umožňuje nastavit automatické aktualizace pluginů. Chybové zprávy z databáze nepotlačuje a zároveň neumožňuje změnit prefix tabulek. Útok nástrojem *sqlmap* proběhl úspěšně. Shield Security chrání webové stránky proti vložení škodlivého kódu do parametru *match* lépe než doposud analyzované pluginy. SQL Injection se podařilo provést až tehdy, byl-li výraz *UNION SELECT* nahrazen následujícím výrazem:

union%23foo%2F*bar%0D%0Aselect%23foo%0D%0A* .

NinjaFirewall a NinjaScanner

Plugin NinjaFirewall neumožňuje změnit prefix tabulek a ani v nastavení neobsahuje možnost, kterou by šlo potlačit chybové zprávy z databáze. Informaci o zastaralé verzi pluginu nezobrazuje. Během analýzy tohoto pluginu byly vyzkoušeny všechny možné způsoby zápisu *UNION SELECT*, dostupné na webu OWASP (OWASP, 2017).

NinjaFirewall nepropustil ani jeden způsob a získání citlivých údajů z databáze se nepodařilo. Útok nástrojem *sqlmap* byl taktéž neúspěšný.

SecuPress

Plugin SecuPress umožňuje změnit prefix tabulek a v placené verzi administrátora informuje o zastaralé verzi pluginu. Dále je možné zamezit instalaci nových pluginů nebo instalaci pluginů či šablon ze souboru ZIP. Chybové zprávy z databáze nepotlačuje. Útok pomocí nástroje *sqlmap* byl úspěšný, a to i přesto, že bylo v nastavení povoleno blokování nástrojů, které hledají SQL Injection zranitelnosti. U útoku využívající zranitelnost v parametru *match* musel být výraz *UNION SELECT* nahrazen stejným výrazem jako u pluginu Wordfence. Díky této změně se podařilo získat uživatelské jméno a otisk hesla z databáze.

4.2.6 Cross-Site Scripting

Pro analýzu obrany proti Cross-Site Scripting byl do webových stránek nainstalován plugin, který obsahuje zranitelnost umožňující provést Stored XSS. Jedná se o plugin Embed Images in Comment (Ascic, a další, 2017), který v komentáři u článku vytvoří náhled obrázku na základě vložené URL. Plugin má v době psaní této práce více jak 300 aktivních instalací a téměř 3 000 stažení. Zranitelnost byla nalezena ve verzi 0.5 (The WPScan Team, 2017). Úspěšný útok je na obrázku níže (Obrázek 18) a URL, která byla do komentáře vložena je následující:

"image" http://codeseekah.com/1.jpg"onload="alert(‘xss‘)".jpg.

Obrázek 18 - XSS útok provedený díky zranitelnosti v pluginu

Zdroj: vlastní zpracování

má 4 body. Zbylé 2 body byly přiděleny pluginu All In One WP Security & Firewall za upozornění při stejném uživatelském jméně a jméně zobrazovaném vedle článku. Přidělené body a pořadí pluginů v ochraně proti zjištění uživatelského jména je v následující tabulce (Tabulka 4).

Tabulka 4 - Komparace pluginů z hlediska obrany proti zjištění uživatelského jména

Název pluginu	**Body**	**Pořadí**
Jetpack	0	7-10
Wordfence Security	8	2-4
iThemes Security	4	5-6
All In One WP Security & Firewall	10	1
Sucuri Security	0	7-10
Acunetix WP Security	0	7-10
BulletProof Security	0	7-10
Shield Security for WordPress	8	2-4
NinjaFirewall a NinjaScanner	8	2-4
SecuPress	4	5-6

Zdroj: vlastní zpracování

4.3.2 Útok hrubou silou

U obrany proti útoku hrubou silou bylo analyzováno šest bezpečnostních funkcí, přičemž každá z nich je pro celkovou bezpečnost jinak podstatná. Kvůli tomu je každé z analyzovaných funkcí přidělen jiny maximální počet bodů. Funkce, společně s maximálním počtem bodů, jsou v následujícím výčtu, přičemž pořadí odpovídá názvům sloupců v tabulce (Tabulka 5), kde jsou výsledky komparace:

1. Blokování uživatele či IP adresy, z které je útok na přihlašovací formulář prováděn (30 bodů).
2. Zaznamenání a přehledné zobrazení chybných přihlášení do administrace a blokovaných uživatelů, resp. IP adres (5 bodů).

Veškeré komentáře musí být schváleny předtím, než jsou zobrazeny ostatním uživatelům. Při schvalování si ovšem administrátor či jiný uživatel systému WordPress nemusí škodlivého kódu všimnout. Útočník tak v tomto případě může velice snadno získat citlivé informace z prohlížeče.

Při analýze bezpečnostních pluginů bylo zjištěno, že ani jeden z vybraných pluginů neblokuje odeslání komentáře se škodlivým kódem a ani nevylepšuje defaultně implementovanou validaci výstupu z databáze. Hláška „xss“ signalizující úspěšný útok se zobrazila u všech pluginů.

U Stored XSS, tedy Cross-Site Scripting útoku, kdy je škodlivý kód ukládán do databáze, je užitečná funkce, která dokáže identifikovat škodlivý kód v databázi. Z vybraných pluginů touto funkcí disponuje pouze plugin BulletProof Security, který ovšem žádný škodlivý kód v databázi nenalezl.

Poslední analyzovaným prvkem byla možnost nastavit HTTP hlavičky, které chrání webovou aplikaci proti XSS či dalším útokům na uživatele a atribut *HttpOnly* u hlavičky *Set-Cookies*, který JavaScriptu zabraňuje v přístupu ke cookies. Analyzovány byly následující HTTP hlavičky:

1. X-Frame-Options,
2. X-XSS-Protection,
3. Content-Security-Policy,
4. X-Content-Type-Options.

V následující tabulce (Tabulka 3) je přehled vybraných pluginů a informace, zda umožňuje danou hlavičku nastavit či nikoliv. Pořadí hlaviček ve výčtu odpovídá názvům sloupců. Atribut *HttpOnly* má číslo 5. Ověření funkčnosti nastavených hlaviček a atributu *HttpOnly* bylo provedeno přes průzkumník v prohlížeči Chrome.

Tabulka 3 - Bezpečnostní HTTP hlavičky ve vybraných pluginech

Název pluginu	**1**	**2**	**3**	**4**	**5**
Jetpack	Ne	Ne	Ne	Ne	Ne
Wordfence Security	Ne	Ne	Ne	Ne	Ne
iThemes Security	Ne	Ne	Ne	Ne	Ne
All In One WP Security & Firewall	Ano	Ne	Ne	Ne	Ne
Sucuri Security	Ano	Ano	Ne	Ano	Ne
Acunetix WP Security	Ne	Ne	Ne	Ne	Ne
BulletProof Security	Ne	Ne	Ne	Ne	Ne
Shield Security for WordPress	Ano	Ano	Ano	Ano	Ne
NinjaFirewall a NinjaScanner	Ano	Ano	Ano	Ano	Ano
SecuPress	Ne	Ne	Ne	Ne	Ne

Zdroj: vlastní zpracování

4.3 Komparace vybraných pluginů

K získání celkového pořadí bezpečnostních pluginů z hlediska jejich efektivnosti byly nejprve provedeny komparace pro jednotlivé hackovací techniky, resp. útoky. Celkové pořadí pak bylo sestaveno na základě pořadí z těchto komparací.

Jelikož neexistují data, pomocí kterých by se dala stanovit priorita analyzovaných obranných mechanismů a funkcí, tak byla tato priorita stanovena subjektivně na základě odborných zkušeností autora této práce. Z důvodu objektivity komparace byl vytvořen komparační nástroj, resp. webová aplikace, ve které uživatel v interaktivním dotazníku stanoví priority obranných mechanismů a funkcí. Získané hodnoty z dotazníku a analýzy bezpečnostních pluginů aplikace vyhodnotí a poskytne výsledné pořadí pluginů.

4.3.1 Zjištění uživatelského jména

Ochrana proti zjištění uživatelského jména je hodnocena 10 body. Body jsou rovnoměrně rozděleny do zabezpečení parametru *author* a URL *wp-json/wp/v2/users*. Každé zabezpečení

3. Možnost nastavení počtu pokusů, po kterých je IP adresa zablokována, délka její blokace a změna URL přihlašovacího formuláře (15 bodů).
4. Vynucení silného uživatelského hesla (5 bodů).
5. Možnost nastavení captchy a dvoufázového přihlášení u přihlašovacího formuláře (15 bodů).
6. Zabránění útoku hrubou silou na XML-RPC (30 bodů).

Ochrana proti útoku hrubou silou na přihlašovací formulář a XML-RPC byla na základě grafu (Obrázek 4) z webu Wordfence, kde je patrné velice podobné množství útoků, ohodnocena stejným počtem bodů. Zablokování uživatele nebo IP adresy po několika neúspěšných pokusech o přihlášení je nejdůležitější ochranou, a proto tvoří 60 % z celkového možného bodového ohodnocení.

Monitorování neočekávaných událostí, jako jsou neúspěšné pokusy nebo blokované IP adresy, je na základě teoretických poznatků důležitou části obranného mechanismu. Administrátor může být včas upozorněn a IP adresu zablokovat ručně. Samotné monitorování ovšem útok hrubou silou nechrání, a proto je pluginu přiděleno za tuto funkci maximálně 5 bodů.

Nastavení počtu povolených pokusů o přihlášení, časové délky blokace IP adresy a změna URL v přihlašovacím formuláři je podle použitých zdrojů z hlediska bezpečnosti velice důležitá. Přísnější nastavení hodnot snižuje pravděpodobnost úspěšného útoku, a proto byla tato funkce ohodnocena 15 body.

Silné heslo jednoznačně patří mezi nejdůležitější opatření proti útoku hrubou silou. Problém je, že se zvyšující se mírou složitosti hesla se častěji stává, že si uživatel heslo někam zapíše. Vzniká tak riziko krádeže hesla. Vynucení silného hesla bylo hodnoceno stejným počtem bodů jako monitorování neočekávaných událostí, tj. 5 bodů.

Stejně jako u přísnějšího nastavení blokací IP adres dochází použitím captchy nebo dvoufázového přihlášení k snížení pravděpodobnosti úspěšného útoku, a proto byla tato obrana hodnocena také 15 body. Dále je popsáno, jakým způsobem byly jednotlivé body udělovány.

Pokud plugin blokoval IP adresu a útok nástrojem WPScan se zcela nepodařil, pak byl přidělen plný počet bodů, tj. 30. Plugin Shield Security for WordPress získal pouze 15 bodů, protože útok nástrojem WPScan nezablokoval. Při nastavení dlouhé délky blokace se útok

nepodařil, ale při menší hodnotě se nástroj WPScan trefil do doby, kdy pokus o přihlášení blokován nebyl a heslo tak bylo uhodnuto. U pluginu Sucuri Security, bylo u této funkce odečteno 5 bodů, protože blokaci IP adresy provedl firewall, který je složitý ke konfiguraci a není zcela součástí pluginu.

Zaznamenání chybných přihlášení bylo hodnoceno 2 body a dostupnost přehledu se zablokovanými IP adresami 1 bodem. Pokud jsou oba přehledy přímo na nástěnce pluginu nebo je-li administrátor informován e-mailem, pak byl udělen další 1 bod. Existence funkce pro monitorování neočekávaných událostí v reálném čase byla hodnocena 1 bodem. Pluginu Sucuri Security byl odečten 1 bod, protože blokované IP adresy a monitorování neočekávaných událostí v reálném čase je dostupné pouze na webových stránkách Sucuri.

Změna URL přihlašovacího formuláře byla hodnocena 5 body. Nastavení povoleného počtu pokusů o přihlášení a délka blokace IP adresy byla hodnocena dohromady 10 body. Pokud plugin určoval horní a spodní hranici těchto hodnot a neumožňoval zadat libovolnou hodnotu, pak byl odečten 1 bod.

Defaultně je možné ve WordPressu zadat heslo označené jako „průměrné“. Plugin, který vynucuje heslo označené jako „bezpečné“ dostal 3 body. Za vynucenou kombinaci čísla a textu nebo kombinaci malého a velkého písma byl v obou případech udělen 1 bod.

Možnost nastavit dvoufázovou autentizaci byla hodnocena 10 body. Přidání captchy do přihlašovacího formuláře bylo ohodnoceno pouze 5 body, protože je oproti dvoufázovému přihlášení snazší captchu obejít, např. pomocí OCR.

Plugin, který neumožnoval zcela vypnout podporu XML-RPC získal o 5 bodů méně, než je maximální počet bodů za „Zabránění útoku hrubou silou na XML-RPC“. Pokud se podařilo útok na XML-RPC provést i přes zapnutou ochranu proti útoku hrubou silou a plugin zároveň obsahuje možnost podporu XML-RPC zcela vypnout, pak mu bylo přiděleno 15 bodů. U pluginu NinjaFirewall bylo odečteno 5 bodů, protože sice umožnuje vypnout podporu XML-RPC a dokázal zablokovat útok, ovšem pouze při nastavení captchy nebo přihlašovacího formuláře na *xmlrpc.php*, což dělá napojení mobilní aplikace na tuto webovou službu složitějším. Je tak velká pravděpodobnost, že administrátor toto zabezpečení nenastaví.

Tabulka 5 - Komparace pluginů z hlediska obrany proti útoku hrubou silou

Název pluginu	**1**	**2**	**3**	**4**	**5**	**6**	**Body celkem**	**Pořadí**
Jetpack	30	0	0	0	10	0	40	8
Wordfence Security	30	5	9	4	0	25	73	4
iThemes Security	30	3	15	3	0	30	81	2
All In One WP Security & Firewall	30	3	15	0	5	30	83	1
Sucuri Security	25	4	0	0	15	25	69	5
Acunetix WP Security	0	1	0	0	0	0	1	10
BulletProof Security	0	3	10	0	5	0	18	9
Shield Security for WordPress	15	3	10	0	15	15	58	7
NinjaFirewall a NinjaScanner	30	5	0	0	5	25	65	6
SecuPress	30	4	15	0	15	15	79	3

Zdroj: vlastní zpracování

4.3.3 Malware

Ochrana proti malwaru byla hodnocena 10 body. Rozeznání každého vloženého nebo upraveného souboru bylo hodnoceno 2 body. Další 4 body získal plugin za možnost nápravy, tj. smazání souborů či vrácení do původního stavu, a možnost zobrazit konkrétní změny v upraveném souboru. Jeden bod byl odečten v případě, že šlo skenování složitě spustit nebo v případě, že šlo skenování změn v souborech obejít. Výsledky komparace pluginů z hlediska obrany proti vloženému malwaru jsou v následující tabulce (Tabulka 6).

Tabulka 6 - Komparace pluginů z hlediska obrany proti vloženému malwaru

Název pluginu	**Body**	**Pořadí**
Jetpack	0	7-10
Wordfence Security	9	2
iThemes Security	5	3-5
All In One WP Security & Firewall	5	3-5
Sucuri Security	2	6
Acunetix WP Security	0	7-10
BulletProof Security	0	7-10
Shield Security for WordPress	0	7-10
NinjaFirewall a NinjaScanner	5	3-5
SecuPress	10	1

Zdroj: vlastní zpracování

4.3.4 Přístup k souborům s citlivými informacemi

Ke komparaci pluginů z hlediska ochrany proti přístupu k souborům s citlivými informacemi byla použita tabulka (Tabulka 2) v kapitole 4.2.4, kde jsou již jednotlivé obranné funkce bodově ohodnoceny. V následující tabulce (Tabulka 7) je součet bodů a konečné pořadí pluginů.

Tabulka 7 - Komparace pluginů z hlediska obrany proti přístupu k souborům s citlivými informacemi

Název pluginu	**Body**	**Pořadí**
Jetpack	0	10
Wordfence Security	5	9
iThemes Security	31	3
All In One WP Security & Firewall	33	1-2
Sucuri Security	10	7-8
Acunetix WP Security	22	4
BulletProof Security	21	5
Shield Security for WordPress	15	6
NinjaFirewall a NinjaScanner	10	7-8
SecuPress	33	1-2

Zdroj: vlastní zpracování

4.3.5 SQL Injection

V kapitole 4.2.5 byly popsány dva útoky SQL Injection, které bylo možné provést díky bezpečnostní díře v jednom z nainstalovaných pluginů. K prvnímu útoku se využil nástroj *sqlmap,* který sekvencí několika možných škodlivých kódů detekoval zranitelnost v parametru *league_id.* V případě zablokování tohoto útoku byly pluginu uděleny 3 body.

Druhý útok byl proveden manuálně a využila se zranitelnost parametru *match.* Vložením škodlivého kódu se podařilo získat uživatelské jméno a otisk hesla z databáze. Některé pluginy filtrovali výraz *UNION SELECT* z URL, avšak nahrazením tohoto výrazu za jiný ekvivalent se podařilo filtrování obejít a škodlivý kód spustit. Plugin získal 4 body, pokud zcela blokoval útok na parametr *match*. V případě, že bylo nutné výraz *UNION SELECT* nahradit, tak byly uděleny 2–3 body podle toho, kolik ekvivalentních výrazů, dostupných na (OWASP, 2017), se podařilo provést. Plugin All In One WP Security & Firewall získal 1 bod navíc, za filtrování názvu tabulky *wp_users* v URL.

Dále byla analyzována možnost změnit prefix tabulek v databázi, potlačit chyby z databáze a upozornění na starou verzi pluginu či jeho automatická aktualizace. Každý ze zmíněných prvků byl hodnocen 1 bodem. Celkem mohl plugin za obranu proti SQL Injection získat 10 bodů.

Výsledky komparace pluginů z hlediska obrany proti SQL Injection jsou v následující tabulce (Tabulka 8).

Tabulka 8 - Komparace pluginů z hlediska obrany proti SQL Injection

Název pluginu	**Body**	**Pořadí**
Jetpack	1	9
Wordfence Security	3	6-7
iThemes Security	3	6-7
All In One WP Security & Firewall	5	3
Sucuri Security	0	10
Acunetix WP Security	2	8
BulletProof Security	6	2
Shield Security for WordPress	4	4-5
NinjaFirewall a NinjaScanner	7	1
SecuPress	4	4-5

Zdroj: vlastní zpracování

4.3.6 Cross-Site Scripting

K porovnání pluginů z hlediska obrany proti Cross-Site Scripting byla použita tabulka (Tabulka 3) v kapitole 4.2.6. V tabulce jsou informace o tom, které z bezpečnostních hlaviček sloužících k ochraně proti XSS a dalším útokům na uživatele umožňuje plugin nastavit. Kromě hlaviček bylo analyzováno, zda plugin blokuje vložení komentáře se škodlivým kódem. Ukázalo se, že žádný z pluginů tak nečinní, a proto již dále nebyla tato ochrana hodnocena.

Každá hlavička byla hodnocena 2 body. Celkem mohl plugin získat 10 bodů. V následující tabulce (Tabulka 9) je výsledek komparace pluginů z hlediska obrany proti XSS a dalším útokům na uživatele.

Tabulka 9 - Komparace pluginů z hlediska obrany proti XSS a dalším útokům na uživatele

Název pluginu	**Body**	**Pořadí**
Jetpack	0	5-10
Wordfence Security	0	5-10
iThemes Security	0	5-10
All In One WP Security & Firewall	2	4
Sucuri Security	6	3
Acunetix WP Security	0	5-10
BulletProof Security	0	5-10
Shield Security for WordPress	8	2
NinjaFirewall a NinjaScanner	10	1
SecuPress	0	5-10

Zdroj: vlastní zpracování

4.3.7 Souhrnné zhodnocení

Následující tabulka (Tabulka 10) importovaná z programu Excel obsahuje pořadí ze všech šesti komparací. Pořadí, které se skládalo z rozmezí 2 hodnot bylo zprůměrováno. Hodnoty jsou zvýrazněny barevnými kolečky. Červená barva značí nejhorší umístění, žlutá umístění dobré a zelená barva výborné.

Tabulka 10 - Pořadí pluginů v jednotlivých komparacích

Název pluginu	**1**	**2**	**3**	**4**	**5**	**6**
Jetpack	8,5	8	8,5	10	9	7,5
Wordfence Security	3	4	2	9	6,5	7,5
iThemes Security	5,5	2	4	3	6,5	7,5
All In One WP Security & Firewall	1	1	4	1,5	3	4
Sucuri Security	8,5	5	6	7,5	10	3
Acunetix WP Security	8,5	10	8,5	4	8	7,5
BulletProof Security	8,5	9	8,5	5	2	7,5
Shield Security for WordPress	3	7	8,5	6	4,5	2
NinjaFirewall + NinjaScanner	3	6	4	7,5	1	1
SecuPress	5,5	3	1	1,5	4,5	7,5

Zdroj: vlastní zpracování

V další tabulce (Tabulka 11) je souhrnné zhodnocení jednotlivých komparací pluginů. Pro stanovení konečného pořadí byl proveden prostý součet pořadí z jednotlivých komparací. Získaná hodnota byla následně odečtena od součtu nejhorších možných pořadí, tj. od 60. Výsledná hodnota je ve sloupci „Body“. Plugin s největším počtem bodů je z hlediska obrany proti vybraným útokům nejefektivnější.

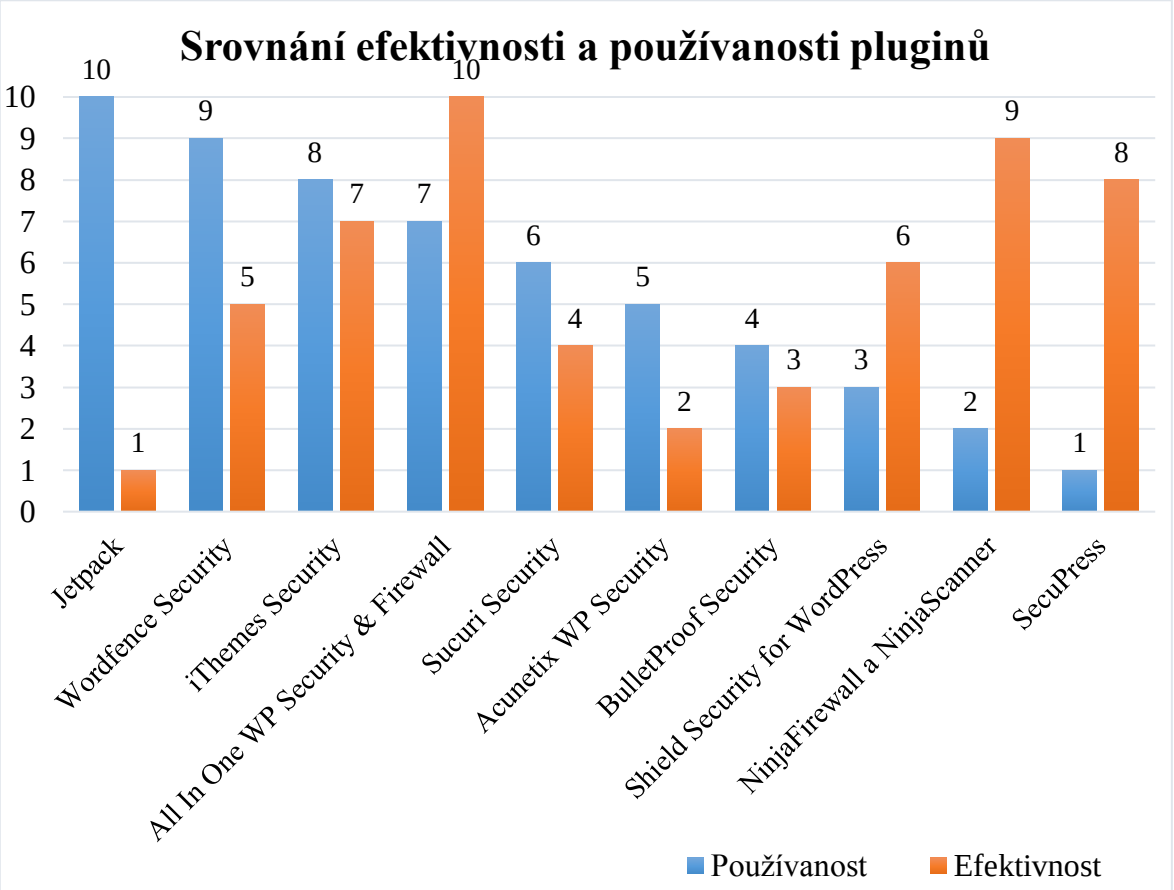
Tabulka 11 - Souhrnné zhodnocení jednotlivých komparací pluginů

Název pluginu	**Verze**	**Body**	**Pořadí**
Jetpack	5.7, placená (216 CZK/měsíc)	8,5	10
Wordfence Security	6.3.22, bezplatná	28	6
iThemes Security	6.8.0, bezplatná	31,5	4
All In One WP Security & Firewall	4.3.1, bezplatná	45,5	1
Sucuri Security	1.8.11, placená (10 USD/měsíc)	20	7
Acunetix WP Security	4.0.5, bezplatná	13,5	9
BulletProof Security	2.8, bezplatná	19,5	8
Shield Security for WordPress	6.2.2, bezplatná	29	5
NinjaFirewall a NinjaScanner	3.6 a 1.1, bezplatná	37,5	2
SecuPress	1.3.3, placená (50 EUR/rok)	37	3

Zdroj: vlastní zpracování

Na následujícím obrázku (Obrázek 19) je graf, ve kterém je srovnání pluginů z hlediska popularity a zjištěné efektivnosti. Z důvodu lepší přehlednosti v grafu bylo pořadí převedeno na body. Hodnota 10 tedy značí nejlepší umístění. Velkého rozdílu efektivnosti a používanosti si lze všimnou u pluginu SecuPress a NinjaFirewall. Právě tyto dva pluginy skončily na druhém, resp. třetím místě v celkové komparaci. Lze tedy očekávat, že popularita těchto pluginů časem poroste.

Obrázek 19 - Graf porovnávající pořadí pluginů v jejich efektivnosti a popularitě



Zdroj: vlastní zpracování

4.3.8 Komparační nástroj

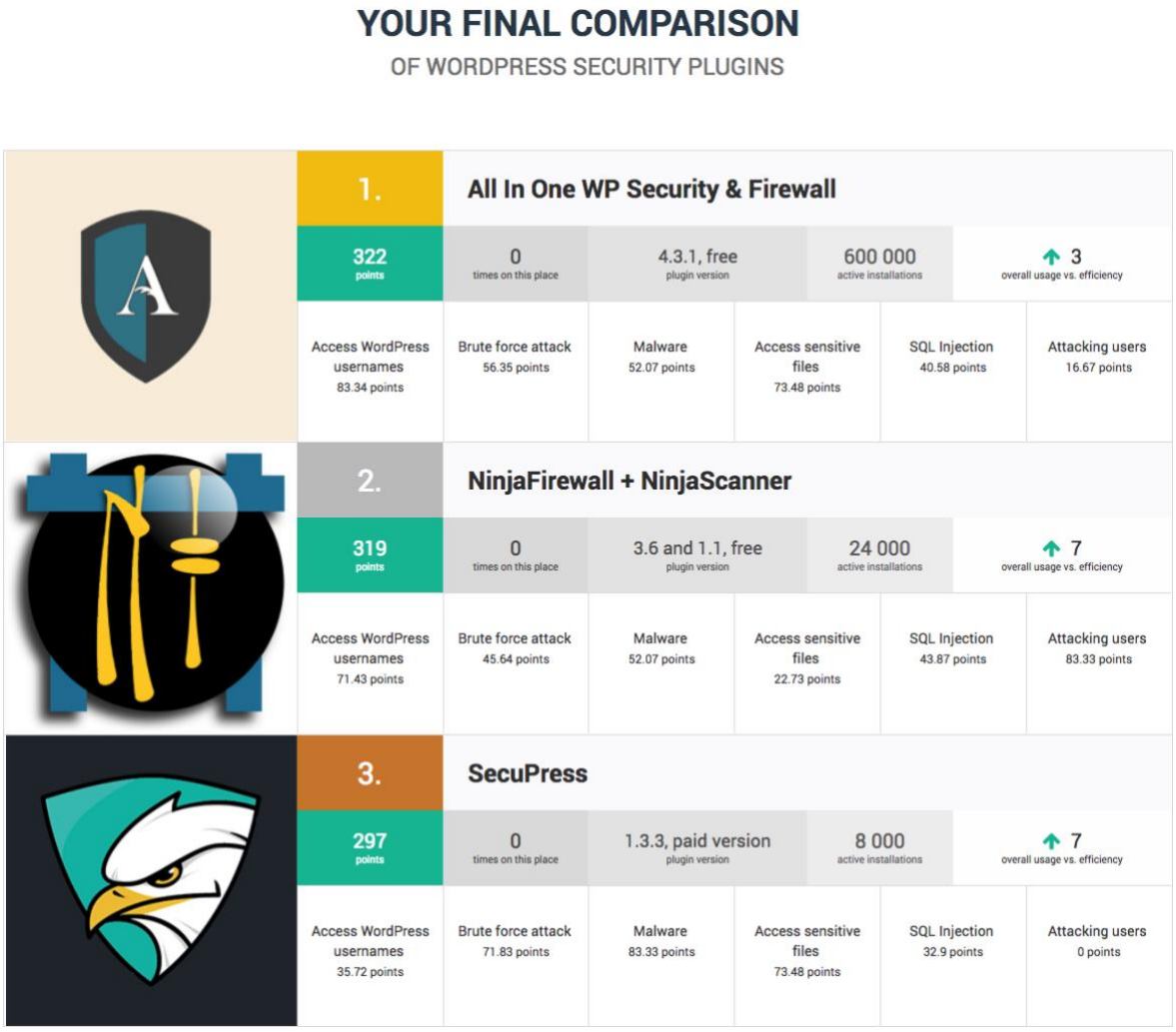
V rámci komparace bezpečnostních pluginů byly body u jednotlivých obraných mechanismů a funkcí stanoveny na základě odborných zkušeností autora této práce. Z důvodu zvýšení objektivity komparace byl vytvořen komparační nástroj, resp. webová aplikace, který běží na adrese *https://wpsecurity.vavo.cz*.

Webová aplikace je záměrně psaná v anglickém jazyce, aby byla použitelná pro větší skupinu lidí. Kromě základních informací o diplomové práci a analyzovaných pluginech, obsahuje aplikace interaktivní dotazník, ve kterém uživatel určí priority analyzovaných obranných mechanismů a funkcí, které byly použity v dříve popsané komparaci. Dotazník má celkem sedm otázek, přičemž šest z nich odpovídá analyzovaným útokům. Poslední otázka slouží k určení priority těchto útoků.

Na základě priorit z dotazníku a zjištěných nedostatků při analýze bezpečnostních pluginů jsou vypočteny body pro všechny útoky. Tyto body jsou dále procentuálně sníženy na základě priorit útoků stanovených v poslední otázce dotazníku. Následně je proveden součet těchto bodů u každého pluginu a je sestaveno konečného pořadí bezpečnostních pluginů.

Na následujícím obrázku (Obrázek 20) je ukázka výstupu generovaného webovou aplikací pro pluginy, které se umístili na prvním, druhém a třetím místě.

Obrázek 20 - Ukázka celkového pořadí z komparačního nástroje



Zdroj: *(Vodička, 2018)*

5 Výsledky a diskuse

Přestože existuje spousta bezpečnostních pluginů, tak jen několik z nich obsahuje dostatek funkcí proti nejběžnějším útokům. Celkem bylo analyzováno deset nejpoužívanějších bezpečnostních pluginů proti šesti nejběžnějším útokům. Nejdůležitější výsledky analýzy pluginů z hlediska obrany proti jednotlivým útokům jsou popsány v následující kapitole.

5.1 Výsledky

Během analýzy bezpečnostních pluginů bylo zjištěno, že pouze plugin Wordfence Security, All In One WP Security, Shield Security a NinjaFirewall chrání webovou aplikaci proti získání uživatelského jména pomocí nástroje WPScan. Všechny tyto pluginy dokázaly také zabránit zjištění uživatelského jména z parametru *author* v URL a z adresy */wp-json/wp/v2/users*. Plugin iThemes Security a SecuPress zabránil alespoň v zjištění uživatelského jména z adresy */wp-json/wp/v2/users*. V některých šablonách, lze uživatelské jméno získat z URL v odkazu umístěném vedle názvu článku. Odkaz vede na stránku se všemi články daného autora. Uživatel s neomezenými právy by nikdy neměl přidávat článek, protože bylo zjištěno, že ani jeden z analyzovaných pluginů tento způsob získání uživatelského jména nechrání. Pouze plugin All In One WP Security alespoň upozorňuje na shodné uživatelské jméno se jménem zobrazovaným vedle názvu článku.

Útok hrubou silou byl prováděn na přihlašovací formulář a webovou službu XML-RPC. Kromě zablokování obou útoků bylo analyzováno, zda plugin zaznamenává a přehledně zobrazuje chybná přihlášení a blokované IP adresy nebo zda vynucuje silné uživatelské heslo. Dále bylo analyzováno, zda plugin umožňuje změnit URL přihlašovacího formuláře, nastavit captchu či dvoufázové přihlašování nebo nastavit počet neúspěšných pokusů, po kterých je IP adresa zablokována a délku této blokace. Při analýze bylo zjištěno, že plugin Acunetix WP Security neobsahuje funkci, která by blokovala IP adresu nebo uživatele po neúspěšných pokusech o přihlášení. Plugin BulletProof Security sice blokovaní má, ale ani jeden útok zablokován nebyl. Oba pluginy skončily v komparaci z hlediska obrany proti útoku hrubou silou nejhůře. O něco lépe dopadl plugin Jetpack, který zablokoval útok na přihlašovací formulář, ale útok na XML-RPC se mu zablokovat nepodařilo. Ostatní pluginy dokázaly oba útoky zcela nebo alespoň částečně zablokovat. Ze všech pluginů dopadl z hlediska obrany proti útoku hrubou silou nejlépe plugin All In One WP Security.

Malware byl vložen do existujícího souboru */wp-content/themes/twentysixteen/404.php* a jako nový soubor do adresáře */wp-admin/* a /*wp-content/themes/twentysixteen/*. Pluginu Jetpack, Acunetix WP Security, BulletProof Security a Shield Security for WordPress se nepodařilo najít ani jeden ze škodlivých soborů. Ostatní pluginy nalezly vždy alespoň jeden malware, a to v adresáři */wp-admin/*. Z analyzovaných pluginů dopadl v tomto ohledu nejlépe SecuPress.

Z hlediska obrany proti přístupu k souborům s citlivými informacemi bylo u každého pluginu analyzováno, zda zjišťuje nedostatečná práva souborů a složek a zda umožňuje jejich nápravu nebo zamezuje procházení souborů ve složce skrze prohlížeč. Dále bylo analyzováno, zda plugin umožňuje vypnout podporu editoru šablon a pluginů a zda chrání webovou aplikaci proti zjištění verze použitého WordPressu. Během analýzy bylo zjištěno, že polovina z vybraných pluginů nekontroluje práva souborů ani složek a neumožňuje vypnout procházení obsahu složky z prohlížeče. Vypnout editor se podařilo ve všech pluginech kromě pluginu Jetpack, Wordfence Security, Acunetix WP Security a BulletProof Security. Verzi použitého WordPressu se podařilo pomocí nástroje WPScan zjistit u všech analyzovaných pluginů. Většina pluginů ale redukuje počet způsobů, jakými lze verzi zjistit. Pouze plugin Jetpack, Sucuri Security a NinjaFirewall vůbec nedokázali zjištění verze WordPressu zabránit.

K provedení útoku SQL Injection byl nalezen plugin se zranitelností ve dvou parametrech v URL. Oba parametry umožňovaly vložit a následně spustit škodlivý kód. Zároveň bylo analyzováno, zda plugin umožňuje změnit prefix v názvu tabulek, potlačuje chybové zprávy z databáze a zda vyžaduje aktualizaci zastaralého pluginu s existující zranitelností nebo alespoň na tuto skutečnost upozorňuje. Útok na jeden z parametrů byl proveden nástrojem *sqlmap,* a kromě pluginu BulletProof Security a NinjaFirewall byl u všech ostatních pluginů úspěšný. Do druhého parametru byl škodlivý kód vložen manuálně a pouze plugin NinjaFirewall dokázal útok zcela zablokovat. U pluginu Wordfence, iThemes Security, BulletProof Security, Shield Security a SecuPress byl škodlivý kód kvůli výrazu *UNION SELECT* zablokován, ale podařil se najít ekvivalentní zápis, který již bez problému prošel. Chybové zprávy dokázal potlačit pouze plugin Acunetix WP Security, avšak v celkovém hodnocení obrany proti SQL Injection skončil společně s pluginem Jetpack a Sucuri Security mezi třemi nejhoršími pluginy. Naopak mezi třemi nejlepšími skončil plugin NinjaFirewall, BulletProof Security a All In One WP Security.

Pro analýzu bezpečnostních pluginů z hlediska obrany proti Cross-Site Scripting, byl do WordPressu nainstalován plugin, který tento útok umožňoval prostřednictvím nedostatečně ošetřeného uživatelského vstupu v komentářích u článků. U pluginů bylo analyzováno, zda plugin dokáže škodlivý kód v komentáři rozeznat a odeslání zablokovat. Žádný z vybraných pluginů komentář zablokovat nedokázal. Plugin BulletProof Security jako jediný obsahuje funkci, která dokáže nalézt škodlivý kód v databázi. Vložený škodlivý kód rozeznat ale nedokázala. Posledním analyzovaným prvkem byla možnost nastavit bezpečnostní HTTP hlavičky a atribut *HttpOnly* u hlavičky *Set-Cookies*. Z vybraných pluginů pouze plugin NinjaFirewall, Shield Security for WordPress, Sucuri Security a All In One WP Security umožňují nastavit vždy alespoň jeden ze zmíněných hlaviček, resp. atributů.

5.2 Diskuse

Analýza efektivnosti bezpečnostních pluginů byla prováděna na webových stránkách běžících na nejnovější verzi systému WordPress. Použita byla verze 4.9.1 a během psaní diplomové práce došlo k vydání verze 4.9.4. Verze 4.9.1 byla vydána v listopadu 2017 a verze 4.9.4 v únoru 2018. Vývoj WordPressu je tak velice rychlý a za vydáváním nových verzí stojí oprava zjištěných chyb v kódu, implementace nových funkcí nebo oprava bezpečnostních nedostatků. Z analyzovaných bezpečnostních mechanismů pouze obrana proti zjištění uživatelského jména, útoku hrubou silou a přístupu k souborům s citlivými informacemi přímo souvisí s funkcemi WordPressu a k nim patřícím zranitelnostem. Vzhledem k tomu, že se jedná o funkce, které ve WordPressu existují již několik let, tak lze předpokládat, že v nejbližší době nedojde k jejich odstranění či výrazné úpravě. Existuje ovšem riziko, kdy dojde např. k odstranění uživatelského jména v URL u odkazu na autora článku, odstranění XML-RPC nebo jiným změnám a výsledné pořadí bezpečnostních pluginů nebude odpovídat použité verzi systému WordPress v této diplomové práci.

SQL Injection a XSS útok využíval zranitelnosti nainstalovaných pluginů. V nové verzi pluginu je většinou bezpečnostní díra odstraněna a útok následně nelze provést. Pro analýzu obrany proti zmíněným útokům byly záměrně z oficiálních stránek WordPressu staženy staré verze pluginů obsahující zranitelnost. S nejnovějšími verzemi pluginů by se stejný útok nepovedl. Existuje ale spousta dalších pluginů nebo šablon, které zranitelnost vůči SQL Injection nebo XSS mají.

Bezpečnostní pluginy byly staženy z oficiálních stránek WordPressu v nejnovější verzi. Jednotlivé verze jsou zmíněny u popisu každého pluginu nebo v tabulce se souhrnným zhodnocením jednotlivých komparací (Tabulka 11). U některých pluginů byla zakoupena placená verze, která oproti bezplatné verzi, obsahovala všechny funkce potřebné k obraně proti vybraným útokům. Zakoupení placené verze i u ostatních pluginů by v jednotlivých komparacích vedlo k získání většího počtu bodů a pravděpodobně i ke změně celkového pořadí. Stejně tak by se celkové pořadí mohlo změnit při vybraní jiných číselných verzí bezpečnostních pluginů.

Každý bezpečnostní plugin byl důkladně prozkoumán a před prováděním útoku nebo použitím funkce pluginu byly nastaveny všechny dostupné obranné mechanismy, které mohly útoku zabránit nebo maximalizovat účinek dané funkce. Kvůli nepřehlednému uživatelskému rozhraní a velkému počtu nastavení v některých pluginech, se mohlo stát, že některé nastavení bylo opomenuto. Je nutné si ovšem uvědomit, že i kvalitní uživatelské rozhraní bezpečnostního pluginu je důležité pro zabezpečení webové aplikace. Pokud bylo některé nastavení opomenuto, tak adekvátně získal plugin méně bodů.

6 Závěr

Tato práce měla za cíl analyzovat a interpretovat efektivnost vybraných pluginů, které slouží k zabezpečení webových stránek založených na systému WordPress. Vybráno bylo deset nejpoužívanějších pluginů, mezi které patří Jetpack, Wordfence Security, iThemes Security, All In One WP Security & Firewall, Sucuri Security, Acunetix WP Security, BulletProof Security, Shield Security for WordPress, NinjaFirewall společně s NinjaScanner a SecuPress. Vynechány byly pluginy, které jsou méně používané, specializují se pouze na jeden typ útoku nebo neobsahují dostatečný počet funkcí. U pluginu Jetpack, Sucuri Security a SecuPress musela být zakoupena placená verze, protože v bezplatné verzi plugin nepovoloval potřebné funkce.

Na základě publikovaných článků byly vybrány nejběžnější útoky proti systému WordPress. Vybrané útoky byly provedeny na webové stránky k těmto účelům vytvořeným. Každá webová aplikace běžela na WordPressu a měla nainstalovaný jeden bezpečnostní plugin. Analyzována byla efektivnost bezpečnostního pluginu z hlediska obrany proti zjištění uživatelského jméno, útoku hrubou silou, malwaru, přístupu k souborům s citlivými informacemi, SQL Injection a Cross-Site Scripting.

U každého pluginu byly zjištěné výsledky ze všech prováděných útoků bodově ohodnoceny. Na základě získaných bodů bylo sestaveno pořadí pluginů pro každý prováděný útok. Celkové pořadí pluginů bylo stanoveno metodou pořadí, tedy součtem pořadí ze všech prováděných komparací. Jako nejefektivnější bezpečnostní plugin vyšel All In One WP Security následovaný pluginem NinjaFirewall s NinjaScannerem, SecuPress, iThemes Security, Shield Security for WordPress, Wordfence Security, Sucuri Security, BulletProof Security a Acunetix WP Security. Poslední skončil plugin Jetpack, který je se 4 mil. aktivními instalacemi nejpoužívanějším z vybraných pluginů. Plugin kromě zabezpečení webové aplikace nabízí další užitečné funkce, které jsou pravděpodobně hlavním důvodem jeho vysoké popularity. Na druhém a třetím místě skončily pluginy, které jsou ze všech analyzovaných pluginů používány nejméně. Lze tak očekávat, že jejich popularita poroste.

Z důvodu zvýšení objektivity komparace byl vytvořen komparační nástroj, resp. webová aplikace, ve které uživatel v interaktivním dotazníku stanoví priority obranných mechanismů a funkcí. Získané hodnoty z dotazníku a analýzy bezpečnostních pluginů aplikace vyhodnotí a vygeneruje celkové pořadí pluginů.

Práce by se dále dala rozšířit o zbylou část bezpečnostních pluginů, které ovšem neobsahují ochranu proti všem běžným útokům. Některé pluginy se například specializují pouze na jeden typ útoku. Celková komparace by pak ztrácela smysl a jednotlivé pluginy by musely být hodnoceny z hlediska obrany proti jednomu konkrétnímu útoku. Struktura této práce by tak musela být pozměněna. Lze očekávat, že v budoucnu přibydou nové bezpečnostní pluginy nebo se rozšíří dostupnost funkcí u pluginů současných. Tyto pluginy by mohly být do práce zahrnuty a porovnány s pluginy analyzovanými v této práci. Dále by se práce mohla rozšířit o analýzu bezpečnostních pluginů z hlediska síly generovaných tokenů, kvality otisku hesla v databázi nebo obrany proti méně běžným útokům na webovou aplikaci.

Diplomová práce uceluje základní principy hackingu z několika českých, a především zahraničních knižních zdrojů a publikovaných článků. Nabyté vědomosti umožňují pochopit bezpečnostní problémy webových aplikací a provést opatření k jejich zabezpečení. Zjištěné poznatky v analýze efektivnosti bezpečnostních pluginů systému WordPress a komparaci pluginů, lze využít k výběru vhodného pluginu pro zabezpečení vlastní webové aplikace. Administrátor webové aplikace, která má již bezpečnostní plugin nainstalován může díky zjištěným výsledkům v této práci přehodnotit svou dřívější volbu a nainstalovat plugin efektivnější. Ukázky jednotlivých útoků v praktické části mohou být použity v rámci penetračního testování vlastní webové aplikace. V některých případech i bez ohledu na to, zda aplikace využívá systém WordPress.

Zjištěné nedostatky v bezpečnostních pluginech mohou vývojáři využít k úpravě dosavadních funkcí nebo vytvoření funkcí nových, jež zabrání útokům, které se podařilo během analýzy úspěšně provést. Získané poznatky z této diplomové práce mohou vývojáři využít při vývoji nových bezpečnostních pluginů nebo webových aplikací, které neběží na systému WordPress.

Citovaná literatura

Caddy, Tom. 2011. *Encyclopedia of Cryptography and Security.* místo neznámé : Springer US, 2011. 978-1-4419-5906-5.

caniuse.com. 2017. Can I use. *HTTP/2 protoco.* [Online] 18. 11 2017. [Citace: 18. 11 2017.] https://caniuse.com/#feat=http2.

Chess, Brian, O'Neil, Yekaterina Tsipenyuk a West, Jacob. 2007. JavaScript Hijacking. *j11y.io.* [Online] 12. 3 2007. [Citace: 29. 12 2017.] https://j11y.io/wp-content/uploads/2009/03/javascript_hijacking.pdf.

Clark, Tim. 2013. *Encyclopedia of Systems Biology.* New York : Springer New York, 2013. 978-1-4419-9863-7.

Clarke, Justin. 2012. *SQL Injection: Attacks and Defense.* Waltham : Elsevier, 2012. 978-1-59749-963-7.

Context. 2016. The Security of HTTP-Headers. *context.* [Online] 18. 5 2016. [Citace: 09. 10 2017.] https://www.contextis.com/blog/security-http-headers.

CVE Details. 2017. PHP 7.0.1 : Related security vulnerabilities . *CVE details.* [Online] 2017. [Citace: 28. 9 2017.] https://www.cvedetails.com/version/189015/PHP-PHP-7.0.1.html.

Acunetix Ltd. Acunetix WP Security. *WordPress.* [Online] [Citace: 12. 01 2018.] https://wordpress.org/plugins/wp-security-scan/.

Abela, Robert. 2014. WordPress Backdoor to Create Administrator Account. *WP WhiteSecurity.* [Online] 06. 12 2014. [Citace: 08. 01 2018.] https://www.wpwhitesecurity.com/wordpress-security-hacks/wordpress-backdoor-administrator-account/.

adaptic.cz. Co je CMS. *adaptic.cz.* [Online] [Citace: 13. 08 2017.] http://www.adaptic.cz/znalosti/slovnicek/cms/.

—. Co je URL. *adaptic.cz.* [Online] [Citace: 13. 08 201.] http://www.adaptic.cz/znalosti/slovnicek/url/.

AITpro Website Security. BulletProof Security. *WordPress.* [Online] [Citace: 13. 01 2018.] https://wordpress.org/plugins/bulletproof-security/.

Amin, Rahul, Iverson, Korin a Peterski, Peter. Download All In One WP Security & Firewall. *WordPress.* [Online] [Citace: 13. 01 2018.] https://wordpress.org/plugins/all-in-one-wp-security-and-firewall/.

Ascic, Zeljko a Kovshenin, Gennady. 2017. Embed Images in Comments. *WordPress.* [Online] 2017. [Citace: 01. 02 2018.] https://wordpress.org/plugins/embed-comment-images/.

Automatic. Jetpack by WordPress.com . *WordPress* . [Online] [Citace: 16. 01 2018.] https://cs.wordpress.org/plugins/jetpack/.

DigiCert. Behind the Scenes of SSL Cryptography. *DigiCert.* [Online] [Citace: 27. 08 2017.] https://www.digicert.com/ssl-cryptography.htm.

Erikson, Jon. 2008. *Hacking: The Art of Exploitation.* [editor] 2nd Edition. San Francisco : No Strach Press, Inc., 2008. 1-59327-144-1.

Žára, Ondřej. 2015. *JavaScript: Programátorské techniky a webové technologie.* Brno : Computer Press, 2015.

Exploit Database. 2015. WordPress Plugin LeagueManager 3.9.11 - SQL Injection. *Exploit Database.* [Online] 02. 06 2015. [Citace: 30. 1 2018.] https://www.exploit-db.com/exploits/37182/.

GitHub. 2017. p0wny-shell. *GitHub.* [Online] 2017. [Citace: 08. 1 2018.] https://github.com/flozz/p0wny-shell.

GODFRED, OWUSU-ANSAH. 2013. XAMP. *Global Health Data Management.* [Online] 14. 7 2013. [Citace: 13. 08 2017.] https://globalhealthdatamanagement.tghn.org/community/blogs/post/5122/2013/07/xampp/ .

gotowebs. 2017. Attacking Access Control for Vulnerabilities. *Gotowebsecurity.* [Online] 16. 11 2017. [Citace: 15. 12 207.] http://gotowebsecurity.com/attacking-access-controls-access-controls-vulnerabilities/.

—. 2017. Everything About Client Side Controls and Bypassing Them. *Gotowebsecuirty.* [Online] 3. 12 2017. [Citace: 15. 12 2017.] http://gotowebsecurity.com/bypassing-client-all-about-side-controls/.

Grigorik, Ilya. 2017. Introduction to HTTP/2. *Google Web Fundamentals.* [Online] 26. 09 2017. [Citace: 18. 11 2018.] https://developers.google.com/web/fundamentals/performance/http2/.

Grossman, Jaremiah, a další. 2007. *XSS Attacks: Cross Site Scripting Attacks: XSS Exploits and Defense.* Boston : Syngress Publishing, 2007. 1-59749-154-3.

Hoi, Sunny. 2017. How To Setup Proxychains With Tor In Kali Linux. *Sunny.* [Online] 27. 10 2017. [Citace: 30. 01 2018.] https://www.sunnyhoi.com/how-to-setup-proxychains-with-tor-in-kali-linux/.

Hunt, Taylor, a další. 2017. HTTP authentication. *MDN web docs.* [Online] 13. 6 2017. [Citace: 09. 10 2017.] https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication.

iControlWP. Shield Security for WordPress. *WordPress.* [Online] [Citace: 24. 01 2018.] https://cs.wordpress.org/plugins/wp-simple-firewall/.

Jackson, Brain. 2016. Hardening Your HTTP Security Headers. *keycdn.com.* [Online] 8. 6 2016. [Citace: 9. 10 2017.] https://www.keycdn.com/blog/http-security-headers/.

JavaScripting.com. 2017. JavaScripting. *JavaScripting.* [Online] 2017. [Citace: 16. 9 2017.] https://www.javascripting.com/.

Jean, Chris. 2017. iThemes Security (formerly Better WP Security). *WordPress.* [Online] 2017. [Citace: 07. 01 2018.] https://cs.wordpress.org/plugins/better-wp-security/.

Jetpack. Powerful services for your WordPress Site. *Jetpack.* [Online] [Citace: 16. 01 2018.] https://jetpack.com/pricing/.

—. Security Features. *Jetpack.* [Online] [Citace: 22. 01 2018.] https://jetpack.com/support/security-features/.

Kangas, Erik. 2016. SSL versus TLS – What's the difference? *LuxSci.* [Online] 19. 07 2016. [Citace: 27. 08 2017.] https://luxsci.com/blog/ssl-versus-tls-whats-the-difference.html.

Kettle, James. 2015. Server-Side Template Injection. *Portswigger.* [Online] 5. 08 2015. [Citace: 28. 12 2017.] http://blog.portswigger.net/2015/08/server-side-template-injection.html.

Kim, Peter. 2015. *Hacking: Praktický průvodce penetračním testováním.* Brno : Zoner Press, 2015. 978-80-7413-313-8.

—. 2015. *The Hacker Playbook 2: Practical Guide To Penetration Testing.* North Charleston : Practical Guide To Penetration Testing, 2015. 978-1512214567.

Klein, Amit. 2005. DOM Based Cross Site Scripting or XSS of the Third Kind. *Web Application Security Consortium.* [Online] 7. 4 2005. [Citace: 19. 12 2017.] http://www.webappsec.org/projects/articles/071105.shtml.

Kümmel, Roman. 2011. *XSS: Cross-SIte Scripting v praxi: O reálných zranitelnostech ve virtuálním světě.* Zlín : Tigris, 2011. 978-80-86062-34-1.

Long, Johnny. 2005. *Google HACKING.* Brno : Zoner Press, 2005. 80-86815-31-5.

Lubbers, Peter, Albers, Brian a Salim, Frank. 2011. *HTML5: Programujeme moderní webové aplikace.* Brno : Computer Press a.s, 2011. 978-80-251-3539-6.

McKinnon, Jenni. 2017. The Ultimate Guide to WordPress Security. *wpmudev.* [Online] 13. 07 2017. [Citace: 13. 08 2017.] https://premium.wpmudev.org/blog/ultimate-guide-wordpress-security/.

Maunder, Mark. 2015. WordPress XML-RPC Brute Force Attacks with multiple logins. *Wordfence.* [Online] 10. 10 2015. [Citace: 18. 01 2018.] https://www.wordfence.com/blog/2015/10/wordpress-xml-rpc-brute-force-attacks-amplification-multiple-logins/.

—. 2017. XMLRPC or WP-Login: Which do Brute Force Attackers Prefer. *Wordfence.* [Online] 31. 01 2017. [Citace: 18. 01 2018.] https://www.wordfence.com/blog/2017/01/xmlrpc-wp-login-brute-force/.

Moreno, Helga. 2014. HomeWeb DesignWordPressCSSToolsTutorialsFontsFreebiesPhotographyIconsShowcasesMore Twitter Facebook RSS Pinterest 4 inShare 2 The 7 Disadvantages of Using WordPress. *instantShift.* [Online] 9. 10 2014. [Citace: 13. 08 2017.] http://www.instantshift.com/2014/10/09/disadvantages-of-wordpress/.

Mozilla. 2018. Content Security Policy (CSP). *MDN web dosc.* [Online] 16. 01 2018. [Citace: 2. 2 2018.] https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP.

NinTechNet. 2018. NinjaFirewall - Documentation. *NinTechNet.* [Online] 2018. [Citace: 01. 02 2018.] https://nintechnet.com/ninjafirewall/wp-edition/doc/.

Olsson, Mikael. 2016. *PHP 7 Quick Scripting Reference.* Berkeley : Apress, 2016. 978-1-4842-1921-8.

OWASP. 2013. Code Injection. *OWASP.* [Online] 31. 12 2013. [Citace: 15. 12 2017.] https://www.owasp.org/index.php/Code_Injection.

—. 2010. Broken Access Control. *OWASP.* [Online] 22. 4 2010. [Citace: 12. 12 2017.] https://www.owasp.org/index.php/Broken_Access_Control.

—. 2017. DOM based XSS Prevention Cheat Sheet. *OWASP.* [Online] 25. 11 2017. [Citace: 27. 12 2017.] https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet.

—. 2016. OWASP. *Cross-site Scripting (XSS).* [Online] 04. 06 2016. [Citace: 19. 12 2017.] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

—. 2017. SQL Injection Bypassing WAF. *OWASP.* [Online] 16. 06 2017. [Citace: 31. 01 2018.] https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF.

—. 2017. SQL Injection Prevention Cheat Sheet. *OWASP.* [Online] 11. 9 2017. [Citace: 18. 12 2017.] https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet.

—. 2017. Types of Cross-Site Scripting. *OWASP.* [Online] 19. 3 2017. [Citace: 19. 12 2017.] https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting.

Pedersen, Torben. 2011. *Encyclopedia of Cryptography and Security.* místo neznámé : Springer US, 2011. 978-1-4419-5905-8.

Regalado, Daniel, a další. 2015. *Gray Hat Hacking: The Ethical Hacker's Handbook.* Fourth Edition. United States : McGraw-Hill Education, 2015. 978-0-07-18328-0.

Rouse, Margaret. 2014. What is access Control? *Tech Target.* [Online] 6 2014. [Citace: 12. 12 2017.] http://searchsecurity.techtarget.com/definition/access-control.

—. 2015. What is Web server? *Whatkls.com.* [Online] 7 2015. [Citace: 13. 08 2017.] http://whatis.techtarget.com/definition/Web-server.

Scambray, Joel a Schema, Mike. 2002. *Hacking Exposed: Web Applications.* Osborne : Corel VENTURA, 2002.

Scambray, Joel, Liu, Vincent a Sima, Caleb. 2001. *Hacking Exposed: Web Applications: Web Application Security Secrets and Solutions.* 3. Osborne : McGraw-Hill Education, 2001. 978-0-07-174064-7.

Schema, Mike. 2012. *Hacking Web Apps: Detecting and Preventing Web Application Security Problems.* Waltham : Elsevier, 2012. 978-1-59749-951-4.

Satrapa, Pavel. 2015. Jak funguje nový protokol HTTP/2. *root.cz.* [Online] 04. 03 2015. [Citace: 18. 11 2017.] https://www.root.cz/clanky/jak-funguje-novy-protokol-http-2/.

SecuPress. SecuPress Free — WordPress Security. *WordPress.* [Online] [Citace: 31. 1 2018.] https://cs.wordpress.org/plugins/secupress/.

Selecký, Matúš. 2012. *Penetrační testy a exploitace.* Brno : Computer Press, 2012. 978-80-251-3752-9.

Sirovich, Jaimie a Darie, Cristian. 2008. *SEO v PHP: Programujeme profesionálně.* Brno : Computer Press, 2008. 978-80-251-2083-5.

Sochor, Tomáš. 2013. *Počítačové sítě 1.* Ostrava : Ostravská univerzita vOstravě, 2013. 978-80-7464-269-2.

sqlinjection.net. 2017. SQL Injection Using UNION. [Online] 2017. [Citace: 16. 12 2017.] http://www.sqlinjection.net/union/.

sqlmap. sqlmap: automatic SQL injection and database takeover tool. *sqlmap.* [Online] [Citace: 30. 1 2018.] http://sqlmap.org/.

Stampar, Miroslav. 2017. sqlmap. *GitHub.* [Online] 25. 09 2017. [Citace: 30. 1 2018.] https://github.com/sqlmapproject/sqlmap/wiki/Usage.

Stuttard, Dafydd a Pinto, Marcus. 2011. *The web application Hacker's Handbook:Findindg and Exploiting Security Flaws.* 2nd edition. Indianapolis : John Wiley & Sons, Inc., 2011. 978-1-118-02647-2.

Sucuri. 2016. Free Website Malware and Security Scanner. *Sucuri.* [Online] 2016. [Citace: 08. 01 2018.] https://sitecheck.sucuri.net/.

Sucuri Inc. 2018. Website Security Platform. *Sucuri.* [Online] 2018. [Citace: 17. 01 2018.] https://sucuri.net/website-security-platform/signup.

—. Sucuri Security – Auditing, Malware Scanner and Security Hardening. *Wordpress.* [Online] [Citace: 13. 01 2018.] https://wordpress.org/plugins/sucuri-scanner/.

Suehring, Steve. 2008. *JavaScript: Krok za krokem.* Brno : Computer press, 2008.

TechTerms. LAMP. *TechTerms.* [Online] [Citace: 13. 08 2017.] http://techterms.com/definition/lamp.

The Ninja Technologies Network. NinjaFirewall (WP Edition). *WordPress.* [Online] [Citace: 24. 01 2018.] https://cs.wordpress.org/plugins/ninjafirewall/.

—. 2017. NinjaFirewall Full WAF vs WordPress WAF modes. *https://blog.nintechnet.com/full_waf-vs-wordpress_waf/.* [Online] 8. 01 2017. [Citace: 24. 01 2018.]

The Tor Project. Option one: Tor on Debian Stretch - stable, Debian Buster - testing, or Debian Sid - unstable . *Tor.* [Online] [Citace: 19. 01 2018.] https://www.torproject.org/docs/debian.html.en.

—. Tor: Overview . *Tor.* [Online] [Citace: 19. 01 2018.] https://www.torproject.org/about/overview.html.en.

The WPScan Team. 2017. Embed Images in Comments <= 0.5 - Unauthenticated Stored XSS. *WPScan Vulnerability Database.* [Online] 17. 08 2017. [Citace: 01. 02 2018.] https://wpvulndb.com/vulnerabilities/8891.

Tips and Tricks HQ. 2018. About. *Tips and Tricks HQ.* [Online] 2018. [Citace: 12. 01 2018.] https://www.tipsandtricks-hq.com/about.

US-CERT. 2015. Web Shells. *US-CERT.* [Online] 10. 11 2015. [Citace: 08. 01 2017.] https://www.us-cert.gov/ncas/alerts/TA15-314A.

Vodička, Václav. 2018. WP Security Plugins. *WP Security Plugins.* [Online] 2018. [Citace: 11. 03 2018.] https://wpsecurity.vavo.cz.

Vrána, Jakub. 2012. *PHP: 1001 tipů a triků pro PHP.* Brno : Computer Press, 2012. 978-80-251-2940-1.

W3C. Header Field Definitions. *W3C.* [Online] [Citace: 20. 08 2017.] https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html.

W3schools. HTTP Methods: GET vs. POST. *W3schools.* [Online] [Citace: 20. 08 2017.] https://www.w3schools.com/tags/ref_httpmethods.asp.

W3Schools. 2017. PHP 5 Sessions. *W3Schools.* [Online] 2017. [Citace: 29. 10 2017.] https://www.w3schools.com/php/php_sessions.asp.

W3tech. W3tech. *Usage of web servers for websites.* [Online] [Citace: 13. 08 2017.] https://w3techs.com/technologies/overview/web_server/all.

Walls, Colin. 2006. *Embedded Software: The Works.* London : Newnes, 2006. 0-7506-7954-9.

Web Developers Notes. What is web server – a computer OR a program? *Web Developers Notes.* [Online] [Citace: 13. 08 2017.] http://www.webdevelopersnotes.com/what-is-web-server.

Weiss, Aaron. 2016. How to Prevent SQL Injection Attacks. *eSecurity Planet.* [Online] 28. 6 2016. [Citace: 18. 12 2017.] https://www.esecurityplanet.com/hackers/how-to-prevent-sql-injection-attacks.html.

—. 2012. Top 5 WordPress Vulnerabilities and How to Fix Them. *eSecurity Planet.* [Online] 20. 4 2012. [Citace: 25. 1 2018.] https://www.esecurityplanet.com/open-source-security/top-5-wordpress-vulnerabilities-and-how-to-fix-them.html.

whichcmstochoose.com. 2017. WordPress. *Which CMS to choose.* [Online] 13. 08 2017. http://whichcmstochoose.com/wordpress.html.

Wodehouse, Carey. 2015. Front-End Web Development: Client-Side Scripting & User Experience. *Upwork.* [Online] 5. 3 2015. [Citace: 2. 9 2017.] https://www.upwork.com/hiring/development/how-scripting-languages-work/.

Wordfence. 2016. How to Restrict WordPress File Permissions. *Wordfence.* [Online] 25. 1 2016. [Citace: 25. 1 2018.] https://www.wordfence.com/learn/how-to-restrict-wordpress-file-permissions/.

—. 2017. Wordfence Security Plugin. *WordPress.* [Online] 2017. [Citace: 06. 01 2018.] https://cs.wordpress.org/plugins/wordfence/.

—. 2017. WordPress Security Plugin. *Wordfence.* [Online] 2017. https://www.wordfence.com/#features.

WordPress. Changing File Permissions. *WordPress.* [Online] [Citace: 29. 1 2018.] https://codex.wordpress.org/Changing_File_Permissions.

—. LeagueManager. *WordPress.* [Online] [Citace: 30. 1 2018.] https://cs.wordpress.org/plugins/leaguemanager/.

—. Plugins. *WordPress.org.* [Online] [Citace: 01. 01 2018.] https://wordpress.org/plugins/.

WordPress.com. About Us. *WordPress.com.* [Online] [Citace: 13. 08 2017.] https://wordpress.com/about/.

WordPress.org. WordPress.org. *About WordPress.* [Online] [Citace: 13. 08 2017.] https://wordpress.org/about/.

WPScan. WPScan. *WPScan.* [Online] [Citace: 09. 01 2018.] https://wpscan.org/.

Wright, Kristen. 2017. 5 Common WordPress Security Issues. *iThemes.* [Online] 16. 01 2017. [Citace: 03. 02 2018.] https://ithemes.com/2017/01/16/wordpress-security-issues/.

Yaworski, Peter. 2016. *Web hacking 101: How to Make Money Hacking Ethically.* místo neznámé : Leanpub, 2016.

Zakas, Nicolas C., McPeak, Jeremy a Fawcett, Joe. 2007. *Professional Ajax.* 2. Indianapolis : Wiley Publishing, 2007.

Zhivilo, Igor. 2017. Wordpress brute force password attack using XML-RPC API. *Thoughts on Web Development.* [Online] 14. 05 2017. [Citace: 18. 01 2018.] http://warolv.net/blog/2017/05/14/wordpress-brute-force-password-attack-using-xmlrpc-api/.

—. 2017. Wordpress scripts for information gathering and pen testing. *Github.* [Online] 2017. [Citace: 18. 01 2018.] https://github.com/warolv/wordpress-scripts.