ÚROVEŇ BEZPEČNOSTI INTERNETOVÝCH APLIKACÍ V ČESKÉ REPUBLICE

SECURITY OF INTERNET APPLICATIONS IN THE CZECH REPUBLIC

Petr Zelenka

Anotace:
Článek se věnuje problematice bezpečnosti internetových aplikací. V článku jsou stručně shrnuty základní techniky útoku na webovou prezentaci a chyby, kterých se mohou vývojáři při tvorbě internetové aplikace dopustit. Součástí článku je vyhodnocení testu několika vybraných českých internetových prezentací, které byly prověřeny z hlediska uvedených technik útoku.

Klíčová slova:
Bezpečnost, internetová aplikace, SQL injekce

Abstract:
This article deals with problems of Internet applications security. The article presents the basic techniques of attack upon the Internet application and the main bugs, which can be found in many web-based applications. The most important part of this article presents results of the test of several Czech Internet applications.

Keywords:

Security, internet application, SQL injection

1. ÚVOD

Bezpečnost internetových aplikací je jedním z nejsledovanějších a nejčastěji diskutovaných témat z oblasti internetových technologií. Většinou se v tomto kontextu mluví o kryptografii, protokolu SSL nebo nejrůznějších způsobech autentizace a autorizace. Tento článek se zabývá jinými aspekty bezpečnosti internetových aplikací. Je zaměřen především na vývojáře a nejčastější prohřešky proti bezpečnosti, kterých se mohou dopoustit.

V článku budou představeny některé velice rozšířené a nebezpečné chyby včetně způsobů, jakým se tyto chyby dají zneužít při případném útoku na webovou prezentaci. Součástí příspěvku bude prezentace výsledků menšího průzkumu, jehož cílem bylo zjistit úroveň zabezpečení vybraných českých internetových prezentací.

2. METODIKA PRÁCE

V úvodu práce budou popsány nejčastější prohřešky proti bezpečnosti internetových prezentací včetně způsobů, jakým se dají tyto nedostatky využít při případném útoku na internetovou prezentaci. Příklady budou realizovány v jazyce PHP.

Samotný průzkum úrovně zabezpečení bude představovat především výběr vhodných internetových prezentací a provedení pokusu o využití potenciálních nedostatků v jejich implementaci. Internetové prezentace budou vybrány pomocí katalogového vyhledávače

Seznam.cz velice jednoduchým způsobem. Budou zvoleny 4 sekce katalogu, které zde kvůli ochraně testovaných prezentací nebudou uvedeny. Z těchto sekcí budou vybrány prezentace, které se nacházejí na prvních pozicích. Vzhledem k obchodní politice společnosti Seznam.cz (první pozice v příslušných sekcích jsou placené) se tedy jedná o profesionální nebo alespoň poloprofesionální prezentace s poměrně vysokou návštěvností.

U jednotlivých prezentací bude provedeno několik testů, upravených dle technologií, které byly použity pro tvorbu příslušné webové prezentace. Výsledky testů budou shrnuty v kapitole č. 4. Autor příspěvku prohlašuje, že veškerá slabá místa příslušných aplikací nezneužil žádným destruktivním způsobem a všechny chyby ihned nahlásil provozovatelům prezentace.

3. POPIS NEJČASTĚJŠÍCH PROHŘEŠKŮ PROTI BEZPEČNOSTI

Testování bezpečnosti by mělo být samozřejmou součástí vývoje každé netriviální internetové aplikace. Bohužel tento typ testování představuje velice časově náročnou a tedy i nákladnou činnost, která nemá na první pohled pro autora ani zákazníka žádný efekt. Konkurence v oblasti tvorby internetových aplikací je velice vysoká a ceny tím pádem značně stlačené dolů. Výsledkem tohoto stavu je často záměrné opomenutí testování. Pokud na testování prostředky přesto zbydou, je testována především funkčnost aplikace a na bezpečnost se zapomíná. Následující odstavce budou věnovány jednotlivým prohřeškům proti bezpečnosti.

3. 1. SQL injekce

SQL injekce je poměrně stará technika útoku na webovou prezentaci. Přesto stále patří k těm nejoblíbenějším a nejvíce nebezpečným. Je zřejmé, že tento způsob útoku se týká pouze dynamických internetových prezentací, tedy prezentací generovaných na základě dat uložených v databázi.

SQL injekce využívá, jak již samotný název říká, k útoku jazyka SQL, respektive záměrné modifikace dynamicky generovaných SQL dotazů. Při generování dynamické webové prezentace se obvykle použije několika SQL dotazů, které jsou sestavovány na základě parametrů aktuální URL. Tyto parametry jsou v příslušném aplikačním kódu webové stránky přímo vkládány jako součást řetězce, který je posléze poslán jako SQL dotaz na databázový server. Pokud útočník ovládá jazyk SQL, zná použitý databázový server a odhalí fungování prezentace, na kterou útočí, může správným zápisem příslušného parametru URL převzít kontrolu nad databází.

3. 1. 1. Příklady SQL injekce

Příklad č. 1

Celou problematiku ilustruje následující příklad. Jedná se o zjednodušený dotaz na seznam zboží pro příslušnou kategorii v elektronickém obchodě. Na první pohled je dotaz zcela neškodný, pokud jsou ovšem na všechna místa doplněny očekávané údaje.

URL www stránky:

```
http://www.web.cz/list.php?cat_id=4
```

Kód generovaného dotazu:

```
$item_query='SELECT t_items.name FROM t_items WHERE
t_items.t_items_cat_id='.$_GET['cat_id'].' ORDER BY t_items.name';
```

V parametru $_GET['cat_id'] očekáváme celé číslo, které udává primární klíč záznamu příslušného záznamu. Pokud tento vstupní parametr, který přichází metodou GET pomocí požadavku na URL příslušné stránky, není ošetřen, můžeme provést například následující útok.

Modifikované URL:

```
http://www.web.cz/list.php?cat_id=4;drop%20table%20t_items;#
```

Vygenerovaný dotaz:

```
SELECT t_items.name FROM t_items WHERE t_items.t_items_cat_id=4;
DROP table t_items;# ORDER BY t_items.name';
```

Středníkem jsme ukončili první dotaz, poté jsme provedli výmaz databázové tabulky a pomocí # jsme zakomentovali zbytek dotazu, tak aby byl dotaz korektní. Komentáře se liší podle použitého SQL serveru. U MSSQL nebo PostgreSQL stačí # nahradit pomocí --.

Příklad č. 2

Dalšími velice častými způsoby útoku pomocí SQL injekce je využití klauzule UNION. V tomto případě musíme znát počet atributů, které má mít výsledek vrácený databázovým serverem (lze zjistit postupem pokus omyl). Poté můžeme k původnímu dotazu připojit například následující dotaz, kterým si do výpisu zboží přidáme i seznam uživatelských účtů a hesel.

Doplněný SQL dotaz:

```
union select concat(uname,passwd) as name from user;
```

Zápis příslušného URL:

```
http://www.web.cz/list.php?cat_id=4%20union%20select%20concat(user,
password)%20as%20name%20from%20user;#
```

Je zřejmé, že útočník musí mít dobré znalosti syntaxe příslušného databázového serveru. Potřebné jsou někdy také dodatečné informace o struktuře databáze, názvech databázových tabulek a názvech jejich atributů. K těmto informacím se lze dostat většinou velice jednoduše. Pokud vyvoláme záměrně chybu v SQL dotazu vložením nekorektního obsahu do URL, je většinou vráceno chybové hlášení, které popisuje kde chyba nastala. Z popisu chyby se můžeme dozvědět dostatek informací pro provedení útoku.

3. 1. 2. Způsoby ochrany proti SQL injekci

Způsoby ochrany proti SQL injekci jsou poměrně jednoduché. Obecně je můžeme rozdělit na dvě skupiny.

- Zabezpečení ze strany autorů databázového serveru**. Společnosti zabývající se vývojem databázových systémů jsou si nebezpečí SQL injekce rovněž vědomi. Proto se snaží proti tomuto způsobu bránit nejrůznějšími způsoby. Například v MySQL nelze od verze 4.1 provádět mnohonásobné dotazy. Výše uvedený příklad by tedy u novější verze tohoto databázového serveru nefungoval.**
- Zabezpečení ze strany vývojáře
  - Testovat hodnoty vstupních parametrů **pro generování dotazu pomocí metod, které testují datový typ vstupního parametru, případně pomocí regulárních výrazů.**
  - Omezovat práva uživatele**, na kterého se připojujeme do databáze. Pro většinu tabulek stačí právo na SELECT. Nepřipojovat se jako administrátor.**
  - Ošetřit apostrofy **escape sekvencí. V mnoha případech SQL injekce potřebuje útočník použí apostrof pro ukončení řetězce v SQL dotazu nebo dopsání nebezpečné části dotazu. Proto bychom měli na vstupu tyto hodnoty ošetřit.**
  - Nepoužívat neověřený a neotestovaný software **jako součást naší aplikace. Mnoho zdarma dostupných systémů nemá ošetřenu možnost SQL injekce.**
  - Potlačení výpisu chybových hlášení **na obrazovku. Lze využíl log soubory, ladění povolit pouze při vývoji a po nasazení do ostrého provozu ladění vypnout.**

3. 2. Ošetření vstupních dat od uživatelů

Každý uživatelský vstup, který je zpracováván v internetové aplikaci, je potenciálně nebezpečný. Velké nebezpečí pro webovou aplikaci představují především ty části prezentace,

kde je uživateli umožněno určitým způsobem modifikovat obsah webu. Respektive z dat přijmutých od uživatele se generuje obsah prezentace. Typickým příkladem takových aplikací jsou diskuzní fóra, návštěvní knihy, online bazary atd. Zanesení škodlivého obsahu může zásadním způsobem ovlivnit chování prezentace nebo její vzhled. Obecně lze rozlišovat dva způsoby útoku.

Prvním způsobem útoku je vložení klientského skriptu do vstupního textového pole (např. při přidávání inzerátu v online bazaru). Typickým příklad útoku, je vložení následujícího textu do jakéhokoliv vstupního pole webového formuláře.

```
<script
type='text/javascript'>window.close();</script>
```

Pokud je tento text uložen do databáze beze změny a je posléze použit pro generování stránky, je zřejmé, že si tuto stránku nepřečte již nikdo, kdo má v prohlížeči povolený JavaScript.

Druhým typickým příkladem je zanesení (X)HTML kódu do webové stránky. Pro formátování a tvorbu layoutu webu se většinou používá tabulek <table> nebo bloků . Velké nebezpečí potom představuje ukončení bloku nebo tabulky pomocí nebo </table> kdekoliv uprostřed stránky. Nejčastějším způsobem útoku je například vložení následujícího kódu.

```
</div></div></div></table></table></table>
```

Tento kód může velice závažným způsobem narušit celkový vzhled stránky. Většinou vede k totálnímu rozpadnutí layoutu celého webu.

Ochrana proti tomuto útoku je poměrně jednoduchá. Většina jazyků používaných pro tvorbu dynamických webů obsahuje funkce nebo metody pro převod těchto značek na html entity nebo minimálně na odstranění tagů uzavřených do špičatých závorek. Velice vhodným nástrojem pro ošetření vstupů od uživatele jsou regulární výrazy.

3. 3. Práce se soubory a adresáři

Velmi nebezpečné může být také mazání souborů a adresářů na webovém serveru pomocí skriptů, které jsou řízeny požadavky z URL. Pokud například v URL předáváme název souboru, který má být smazán, může dojít k situaci popsané v následujícím příkladu.

Aplikační kód: `unlink('images/'.$_GET['file']);`

Bezpečné URL:

`http://www.web.cz/delete.php?file=image.jpg`

Nebezpečné URL:

`http://www.web.cz/delete.php?file=../../configuration/config.php`

Jak je vidět z ukázky, pokud neošetříme vstup $_GET[‘file’], můžeme skrze URL mazat libovolné soubory, ke kterým má skript přístupová práva na zápis.

Jednoduchým způsobem, kterým se dá těmto problémům předejít, je využívat pro ošetření vstupů regulární výrazy nebo speciální funkce či metody příslušného programovacího jazyka.

3. 4. Ukládání konfiguračních údajů

Konfigurační údaje pro příslušnou webovou aplikaci se často ukládají do samostatných konfiguračních souborů, které jsou aplikací dle potřeby načítány. Součástí těchto souborů bývají většinou i přihlašovací údaje do databázového systému nebo na ftp server. Konfigurační údaje jsou většinou definovány v příslušném programovacím jazyce jako konstanty programu.

Nejdůležitější krok, který může vést k zásadní snížení bezpečnosti aplikace, představuje pojmenování tohoto souboru. Při jeho pojmenování musíme brát ohled na použitý webový server a serverovou skriptovací technologii. Pokud totiž souboru dáme koncovku .ini nebo .conf a používáme například webový server Apache nebo ISS, je při požadavku na vystavení takového souboru zaslán do prohlížeče kompletní obsah souboru. Tzn. obsah souboru není

interpretován. Proto se pro pojmenování konfiguračních souborů doporučuje používat takové koncovky, které zaručí vykonání kódu i při požadavku na samostatný soubor. Konfigurační soubory pro PHP skripty by tedy měly mít koncovku .php, pro ASP skripty .asp atd. Zde záleží na konfiguraci webového serveru.

4. PRŮZKUM ÚROVNĚ ZABEZPEČENÍ VYBRANÝCH INTERNETOVÝCH PREZENTACÍ

Celkem bylo otestováno 20 prezentací, které byly vybrány dle postupu popsaného v kapitole č. 2.

Důležité je poznamenat, že výsledky testů jsou subjektivní a závisí především na znalostech a schopnostech testera. Testy probíhaly vždy na základě stejného scénáře, ovšem samotná konfigurace testu se pro každou prezentaci lišila, především na základě technologií použitých pro implementaci příslušného webu. Podrobnosti o jednotlivých testech zde nebudou z bezpečnostních důvodů uvedeny. Nicméně všechny vycházely z obecných principů popsaných v kapitole č. 3.

Velice důležité je poznamenat, že při testech nedošlo k žádnému omezení činnosti příslušných prezentací a veškerá nalezená slabá místa byla ihned nahlášena provozovatelům prezentací.

Typ testu	Vyhovělo	Nevyhovělo
SQL injekce	15	5
Ošetření uživatelských vstupů	14	6
Nechráněná konfigurace	19	1

5. ZÁVĚR

I přesto, že test popsaný ve 4. kapitole byl především z časových důvodů málo rozsáhlý a výběr testovaných prezentací byl proveden subjektivně, můžeme z výsledků částečně vyvodit určité závěry.

Testování bezpečnosti je v českých firmách věnujících se vývoji webových aplikací stále nezvykle často opomíjeno. O prohřešcích proti bezepečnosti, které byly popsány ve 3. kapitole se můžeme dočíst v téměř každé kvalitní publikaci o webové tvorbě nebo manuálech k příslušným technologickým nástrojům. I přesto zůstává neúměrně velké množství aplikací z bezpečnostního hlediska naprosto nevyhovujících.

Důvody bychom mohli pravděpodobně hledat v malé kvalifikaci tvůrců webových aplikací a snahách snižovat náklady na vývoj za každou cenu. Pokud ovšem chceme provozovat tak kritické aplikace, kterými jsou například elektronické obchody, a chceme, aby uživatelé iternetovým aplikacím důvěřovali, nesmí být tato fáze vývoje v žádném případě opomíjena.

Literatura:

1. WELLING, L.,THOMSON, L. PHP a MySQL: rozvoj webových aplikací. Praha: Soft Press, 2002. 718 s. ISBN 80-86497-20-8.
2. ANLEY, Chris. Advanced SQL Injection In SQL Server Applications [online]. 5. 10. 2002, [cit. 2004-05-05]. Dostupné z: <http://www.nextgenss.com/papers/advanced_sql_injection.pdf>.
3. HARPER, Mitchel. SQL Injection Attacks - Are You Safe? [online]. 17. 5. 2002, [cit. 2004-05-05]. Dostupné z: <http://www.sitepoint.com/article/794>.

Kontaktní adresa autora:

Ing. Petr Zelenka,
Katedra informačního inženýrství, Provozně ekonomická fakulta, ČZU Praha,
e-mail: zelenka@pef.czu.cz