SQL INJECTION VULNERABILITY LOCATOR FOR KENTICO CMS

Dominik Pintér
Bachelor Degree Programme(3), FIT BUT
E-mail: xpinte02@stud.fit.vutbr.cz

Supervised by: Peter Solár
E-mail: isolar@fit.vutbr.cz

ABSTRACT

SQL injection vulnerability is one of the most dangerous vulnerabilities for web applications today. This paper is focused on design of application which locates SQL injections in source code of Kentico CMS for ASP.NET. Paper is mainly aimed to present how application works and on basic principles which are used in it´s implementation. Paper also describes SQL injection vulnerability, consequences of exploitantion and possible protection methods.

1. ÚVOD

Mezi největší problémy dnešního internetu patří nedostatečné zabezpečení přenášených infromací tak, aby se k nim dostali pouze oprávnění uživatelé. Webové servery i aplikace na nich běžící nepřetržitě čelí atakům útočníků, kteří se snaží pomocí různých zranitelností proniknout do cizích systémů. SQL injection patří mezi nejvíce zkoušené možnosti. Ačkoli tato zranitelnost není žádnou novinkou, je velmi dobře zdokumentovaná včetně možné obrany proti ní, nemalá část aplikací je denně úspěšně napadána a zneužívána právě tímto typem útoku. Přitom následky úspěšně provedeného útoku mohou být fatální.

Tato práce si klade za cíl představit aplikaci – vyhledávač SQL injections. Program se snaží pomocí statické analýzy nad zdrojovým kódem odhalit zranitelné místa. Aplikace pracuje se zdrojovým kódem CMS systému Kentico CMS for ASP.NET.

2. SQL INJECTION

Útok SQL injection spočívá v tom, že útočník dokáže přes webovou aplikaci spustit téměř libovolný SQL kód nad databázovým serverem. Provede to tak, že předá aplikaci na vstup speciální řetězec – část SQL dotazu. A pokud tento řetězec není nijakým způsobem upravován a předá se jako část dotazu databázovému serveru, jedná se o zranitelnost typu SQL injection.

2.1. Příklad SQL injection

Mějme webovou aplikaci, například knihovnu. Aplikace uchovává svá data v databázi. Tabulka `titul` obsahuje všechny tituly, tabulka `uzivatel` registrované uživatele systé-

mu. Aplikace umožňuje vyhledávání v titulech podle názvu titulu. Ten uživatel zadává do textového políčka a tlačítkem odesílá webové aplikaci. Stisknutím tlačítka se odešle na webový server následující URL:

`http://www.knihovna.tld/?search=vyraz`

kde `vyraz` je uživatelův vstup. Webová aplikace sestaví následující SQL dotaz:

```
SELECT * FROM titul WHERE nazev = 'vyraz'
```

kde `vyraz` je nijak neupravený řetězec získaný z URL. Uživatel pravděpodobně zadá řetězec jako „Hamlet“ nebo „Romeo a Julie“. Ale nic mu nebrání v tom, aby vložil například tento řetězec: „'; SELECT * FROM uzivatel --“. Výsledný dotaz pak bude vypadat:

```
SELECT * FROM titul WHERE nazev = ''; SELECT * FROM  uzivatel --'
```

a uživatel tak dostane výpis všech uživatelů registrovaných v systému.

2.2. Obrana před SQL injection

Pro útok je klíčové vložení libovolného řetězce do dynamicky sestavované části dotazu, zejména pak možnost vložení znaku ' (apostrof). Prvním logickým způsobem obrany je validace dat. Ovšem ne vždy je možné vyloučit z množiny vstupních dat právě znak '. V takových situacích lze použít parametrických dotazů a nebo nahrazením znaku ' tzv. escape sekvencí '' (dvojice apostrofů). Více o možnostech obrany lze nalézt v [1] a [2].

3. PRINCIP APLIKACE

Jak už bylo zmíněno v úvodu, aplikace nepracuje na principu pokus-omyl, ale analyzuje zdrojový kód. Cílem je vytvořit vyhledávač pro konkrétní systém – Kentico CMS. Lze tedy využít vlastností tohoto CMS, jazyka C#, ve kterém je napsán a platformy ASP.NET, na které pracuje.

Prvním krokem v návrhu je nalezení místa, odkud začít vyhledávací proces. Zde vyjdeme z toho, že Kentico CMS je funkčně rozdělený na jednotlivé vrstvy. Nejnižší z nich je datová, která se stará o přístup do databáze. Identifikujeme tedy konkrétní metody, které přímo spouštějí databázové dotazy. Dále využijeme toho, že jazyk C# je silně typovaným jazykem. Vybereme pouze metody, které mají jako parametr řetězcový typ `string`, protože pouze ty jsou potencionálně zneužitelné. Nyní již stačí vyhledat ve zdrojovém kódu všechna volaní těchto metod a od nich začít vyhledávání.

Dále je třeba specifikovat, jak odlišit bezpečnou část kódu od zneužitelné. Z analýzy zdrojového kódu webové projektu Kentico CMS lze zjistit, že pro ošetření SQL injection se ve zdrojovém kódu používá nahrazování znaku ' znaky ''.

Dalším krokem je určení místa, kde ukončit hledání. Toto místo je určené vstupními body prohledávané aplikace, v případě ASP.NET platformy jsou to ASP.NET události. Druhou možností je nalezení ochrany před SQL injection. Pokud je nalezena, nemá smysl hledat další.

Nyní se zaměřme na samotné vyhledávání a jeho postup od počátečního bodu po koncový. Program bude provádět v každém kroku tzv. dekompozici. To je proces určování, zda-li je daná část kódu ošetřena proti SQL injection. Probíhá tak, že se daný kód dekomponuje (rozkládá) na jednotlivé části a ty se rekurzivně prochází než se nalezne koncový bod. Typ koncového bodu pak rozhoduje o tom, zda-li je kód označen jako bezpečný či nikoli. De-

kompozice začíná v místě volání metody datové vrstvy, první se dekomponují tedy parametry této metody. Dekompozice je různá pro rozdílné typy parametrů:

- Lokální proměnná - najde se výskyt, kde všude je proměnná použita jako L-value a pravé části se dekomponují.
- Property - hledá se pouze v metodě `Get`. Naleznou se výskyty klíčového slova `return`, které značí výstupní bod aplikace. Parametr tohoto klíčového slova se dekomponuje. Metoda `Set` se neprohledává, protože ta property nastavuje, její hodnota se získá metodou `Get`. Proto stačí SQL injection ošetřit pouze v metodě `Get`.
- Metoda - dekompozice probíhá stejným způsobem jako dekompozice `Get` metody u property.

3.1. ALGORITMUS

Algoritmus, který aplikace implementuje vypadá následovně:

1) Vyhledej volání všech metod datové vrstvy a vytvoř z nich seznam <metody>.
2) Profiltruj seznam <metody> tak, aby obsahoval pouze potencionálně zneužitelné metody.
3) Dokud není seznam <metody> prázdný, vyber první metodu ze seznamu a pokračuj bodem 4, jinak vypiš výsledky a ukonči se.
4) Vyhledej parametry typu `string` metody a vytvoř z nich seznam <parametry>.
5) Dokud není seznam <parametry> prázdný, vyber parametr ze seznamu, proveď dekompozici a opakuj bod 5 v případě, že při dekompozici byla nalezena ochrana před útokem a nebo jdi na bod 6 v případě, že ochrana nalezena nebyla. V případě, že je seznam prázdný, aktuální metodu označ jako bezpečnou a jdi na bod 3.
6) Vymaž seznam <parametry>, aktuální metodu označ jako nebezpečnou a jdi na bod 3.

4. ZÁVĚR

Tato práce měla představit princip vyhledávače zranitelností typu SQL injections pro zdrojový kód Kentico CMS. Důraz byl kladem zejména na princip samotné aplikace, naopak implementace.

Aplikace je prototyp a je nyní ve stavu vývoje, datum dokončení základní verze je stanoven na Květen 2010. Prostor pro další vývoj je velký, zejména v možnostech optimalizace implementace. Dále by bylo možné zobecnit vyhledávací algoritmus tak, aby nebyl závislý na konkrétním zdrojovém kódu, případně, aby nad zdrojovým kódem kontroloval i další typy zranitelností.

REFERENCE

[1] Meier, J. D. How To: Protect From SQL Injection in ASP.NET [online]. Publikováno: Květen 2005 [cit. 26.12.2009]. Dostupné z: <http://msdn.microsoft.com/en-us/library/ms998271.aspx>

[2] Howard, Michael - LeBlanc David. Writing secure code. Second edition. Redmond, Washington: Microsoft Press, 2003. 768s. ISBN 0-7356-1722-8.