Technische Universität Sofia
Fakultät für deutsche Ingenieur- und Betriebswirtschaftsausbildung

SS 2016

Seminar: “Wissenschaftliches Schreiben”
Seminarleiterin: Stanka Murdsheva
Fachliche Betreuung: Blagovest Kirilov

Thema: SQL Injection Prevention

Kalin Petrov, Nr.201213035
kalin.petrov@tu-sofia.bg
Mobil:+359895357540
Datum: 27.03.2016

Inhaltsverzeichnis

1. Einleitung

Betrachtet man die Entwicklung der letzten Jahre, so kann man sagen, dass das Thema über die Sicherheit unserer Information in verschiedenen Softwaresysteme und auch im Internet immer mehr aktueller wird. Dieses Thema gewinnt an Bedeutung, weil durch die Digitalisierung viele persönliche Informationen ins Internet verlagert werden. Softwaresysteme werden notwendiger in unserem täglichen Leben, deshalb wird es immer wichtiger, dass die Leistungen, die sie erbringen, sind verfügbar, wenn wir sie brauchen. Man muss sich aber auf die Integrität des Systems verlassen können und damit die Informationen, die sie halten zur Verfügung stellen. Was mehr ist, unsere Gesellschaft und Wirtschaft hängen von dem Vertrauen bestimmter Daten ab. Deshalb muss jedes heutige Softwaresystem die notwendigen Sicherheitsanforderungen erfüllen.

Die zunehmenden Datenmengen erfassen immer mehr Information über Patienten, Verbraucher und Bürger. Als weitere Möglichkeiten der Vernetzung und Nutzung dieser Daten entstehen, entstehen auch Angelegenheiten, die sich aus der Vielzahl von Beteiligten kommen, über die Behandlung der persönlichen Daten. Verständlicherweise haben Fragen in Bezug auf die Datenbank und Anwendungen von Sicherheit in den letzten Jahren gestiegen.

Viele Entwickler sind sich nicht bewusst, wie man sich an SQL (Structured Query Language) Abfragen zu schaffen machen kann und nehmen an, dass eine SQL Abfrage ein vertrauenswürdiges Kommando ist. Deshalb können oft Hacker die Schwachpunkte in dem Database verwenden. Die SQL Injection (*SQL-Einschleusung*) ist eine häufige und effiziente Angriffsform. Sie ist eine der verheerendsten Sicherheitslücken zu Auswirkungen eines Unternehmens, da es zu der Belichtung aller sensiblen Daten in einer Anwendung führen kann, einschließlich nützliche Informationen wie Benutzernamen, Kennwörter, Namen, Adressen, Telefonnummern und Kreditkarte. SQL Injection ist gegenwärtig die am weitesten verbreitete Form der Website Angriff. Diesem Web Forms sind sehr häufig, weil sie sind oft nicht korrekt kodiert sind und die Hacking Tools verwendet, um Schwachstellen zu finden, leicht im Internet verfügbar sind. Diese Art von Nutzen ist einfach genug zu erreichen, dass auch unerfahrene Hacker vollenden können. Wenn das passiert, könnte ein Angreifer mit Hilfe von SQL Injection die Authentifizierung umgehen oder sogar Identität von bestimmten Benutzern bekommen. Das führt zu anderen Bedrohungen, wie die Beseitigung oder Veränderung von Daten in einer Datenbank. Deswegen wird es notwendig

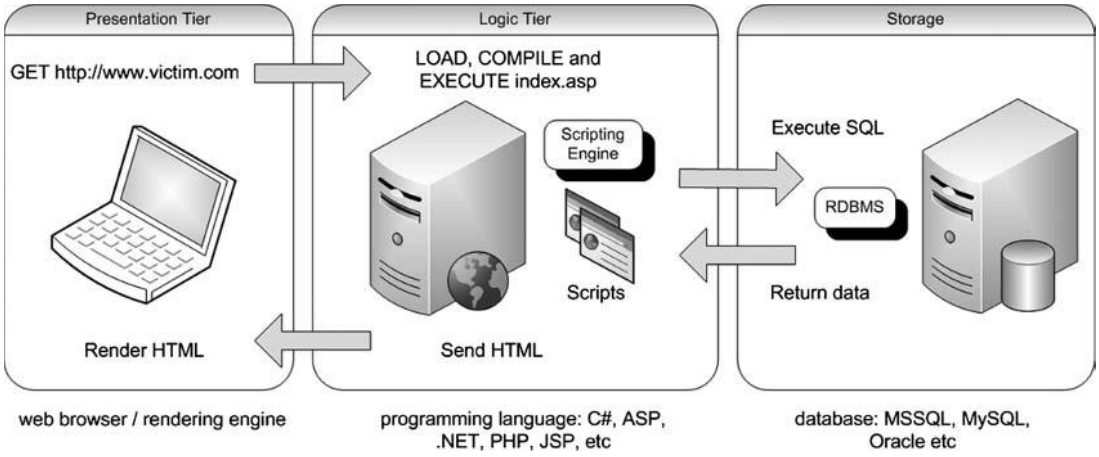
in einem zunehmend datengesteuerte Welt, dass man lernt, wie solche Entwürfe zugeordnet werden könnten, damit sie die praktischen Sicherheitsanforderungen erfüllen.
Die vorliegende Arbeit beschäftigt sich genau mit SQL Injection. Um das Thema besser zu verstehen, umfasst die Arbeit kurz auch die Grundaspekte der SQL-Sprache und auch die Anwendungsarchitektur(Three-Tier Architecture), die sehr oft verwendet wirdt. Dann wird das SQL Injection definiert und erkärt. In dieser Arbeit werden die folgende Fragen behandelt: Welche sind die Folgen und Risikos von einem SQL Injection? Warum ist diese Angriffsart so popular und benutzt? Wie könnte man SQL Injection verhindern und vermeiden? Auf den folgenden Seiten untersuche ich die Antworte zu diesen Fragen zu finden. Ich werde auch die Vor- und Nachteile der verschiedenen Schütze gegen SQL Injection vorstellen. Basis der Überlegungen sind verschiedene Online- und Printquellen.
Ich habe dieses Thema gewählt, weil ich interessiere mich für Datenbanken und ihre Rolle in den Softwaresystemen. Dieser Bereich ist sehr populär und anziehend, deshalb würde ich gern in der Zukunft darin arbeiten. Meiner Ansich nach ist es wichtig, dass man weiß, was es hinter dem Interface der Software gibt. Viele Menschen haben keine Ahnung , was ein Softwaresystem enthaltet. Die meisten Menschen benutzen die Möglichkeiten, die das System ermöglicht, ohne darüber zu denken, dass einige Systeme auch Gefahren in sich enthalten. Deshalb trägt jede Firma, die Software erstellt, Verantwortung. Es ist notwendig, dass man seine Kunden vor diesen Gefahren schützen kann. Das passiert wenn man mit der wichtigsten Sicherheitsanforderungen bekannt ist. Eine sichere Software schützt gegen die ernste Bedrohungen und somit vermeidet sehr teuere Folgekosten.

2. Übersicht über SQL und die Three-Tier Architektur

Was versteht man unter SQL Injection? Damit wir auf diese Frage antworten können, müssen wir zuerst erklären, was SQL ist und wie die mesten Web-Anwendungen funktioniren. Mit Hilfe der Überblich der Anwendungen, wird man leichter verstehen, wie SQL Injection erscheint.Was ist SQL?
“The name SQL is derived from Structured Query Language. SQL is a comprehensive database language: It has statements for data definition, query, and update. In addition, it has facilities for defining views on the database, for specifying security and authorization, for defining integrity constraints, and for specifying transaction controls. It also has rules for embedding SQL statements into a general-purpose programming language such as Java or COBOL or C/C++” (Fundamentals of Database Systems, S.208). SQL steht kurz für Structured Query Language. Es ist eine umfassende Datenbanksprache. Sie hat Erklärungen

für Definition, Abfrage und Aktualisierung der Daten. Außerdem hat die Sprache Einrichtungen für die Definition von Sichten auf die Datenbank, für die Angabe von Sicherheit und Zulassung, zur Definition von Integrität und zur Angabe von Transaktionen. Sie hat auch Regeln für die Einbettung von SQL-Anweisungen in Programmiersprache wie Java, COBOL oder C/C++. SQL wurde von einer Gruppe in IBM Research im 1972 erfunden und seitdem fast alle Datenbanksysteme SQL unterstützen. Offensichtlich sind die Befehle von SQL der natürlichen englischen Sprache nachempfunden. SQL-Anweisungen werden für Speichern, Abrufen und Bearbeiten von Daten in einer relationalen Datenbank. SQL ist die nahezu universelle Sprache der Datenbanken, erlaubt die Speicherung, Manipulation, und Wiederauffinden von Daten.

Webanwendungen sind sehr häufig in der heutigen Web-fähige Gesellschaft Die Datenbanken sind die Gemeinsamkeit zwischen die verschiedenen Arten von diesen Anwendungen.



SQL Injection Attacks and Defense, S.4

Auf dem gegebenen Bild ist die Three-Tier Anwendungsarchitektur dargestellt. Eine datenbankgestützte Webanwendung hat normalerweise drei Stufen: Präsentation (ein Web-Browser oder Rendering Engine), Logik (eine Programmiersprache wie C#, ASP, .NET, PHP, JSP, etc.), und Speicher (eine Datenbank wie Microsoft SQL Server, MySQL, Oracle, etc.). Der Webbrowser (der Präsentationsebene, wie Internet Explorer, Safari, Firefox, etc.) sendet Anfragen an die mittlere Stufe (die Logik), deren Dienste die Zugriffe durch Abfragen und Updates gegen die Datenbank (der Speicher) machen. "A fundamental rule in a three-tier architecture is that the presentation tier never communicates directly with the data tier; in a three-tier model, all communication must pass through the middleware tier. Conceptually, the three-tier architecture is linear" (SQL Injection Attacks and Defense, S.5). Die Grundregel dieser Architektur ist linear,dass bedeutet,dass die Präsentationstufe nie direkt mit dem Specher mitteilt. Die Kommunikation ist

immer durch die Logikstufe. Es ist gegen Sicherheitsmaßnahmen. Was interessiert uns hier am meisten ist der Specher (Datenbank). Dieses Tier hält Daten unabhängig von Anwendungsservern. Die Logik wird durch Database Connector mit der Datenbank verbunden und führt SQL-Anweisungen an die Datenbank. Die Datenbank liefert die Daten für die Database Connector, das an die Scripting Engine innerhalb der Logik überliefer werden. Die Logik implementiert alle Anwendungen bevor sie zum Beispiel eine Webseite im HTML-Format an den Webbrowser des Benutzers zurückgibt. Four-Tier Architecture werden immer mehr populär. Sie enthalten auch Anwendungstufe.

Es gibt mehrstufige Architekture. Tiers können sich auf verschiedenen Maschinen oder auf der gleichen Maschine befinden. Je mehr Stufen man verwendet, desto spezifischer jedes Tier ist. Die Trennung der Zuständigkeiten einer Anwendung in mehrere Stufen erleichtert die Skalierung der Anwendung und erlaubt die Entwickler eine bessere Trennung von Entwicklungsaufgaben. Zum Beispiel, eine Entscheidung zur Änderung der Datenbankanbieter könnte nicht mehr als ein Paar Veränderungen der anwendbaren Teile der Anwendungsebene. Die Präsentation und Logik Tiers bleiben unverändert. Drei - und vierstufige Architekturen sind die häufigsten eingesetzten Architekturen auf dem Internet heute.

Die Verfügbarkeit dieser Systeme und der Sensibilität der Daten, die sie speichern und verarbeiten, wird immer wichtiger für alle großen Unternehmen. Webanwendungen benutzen verschiedene Technologien und enthalten eine bedeutende Menge von angepasstem und modifiziertem Code. Die Natur dieser Anwendungen macht sie ein beliebtes Ziel für Angriffe.

3. Sicherheitsprobleme und Injection in SQL

SQL Injection ist gegenwärtig die meistgenutzte Form zum Angriff der Website. “84 percent of respondents say the most common gateway attack experienced by their organization over the past 12 months is an SQL injection, followed by cross site scripting (23 percent of respondents) and cross site request forgery (at 18 percent)” (“SQL injection has surfaced as the no.1 attack in 2015"). Die SQL Injection (*SQL-Einschleusung*) ist die häufigste Angriffsform für 2015 nach dem vorgegebenen Forschung. Dieser Angriff wird weitgehend von Angreifern verwendet, um unberechtigten Zugang von Systemen zu bekommen. “SQL injection has probably existed since SQL databases were first connected to Web applications. However, Rain Forest Puppy is widely credited with its discovery — or at least

for bringing it to the public's attention in 1998" (SQL Injection Attacks and Defense S.2). Der Autor erklärt,dass SQL Injection wahrscheinlich seit dem Zusammenhang der Datenbanken und der Webanwendungen vorlag. Allerdings wurde es von Rain Forest in 1998 entdeckt. Seitdem ist dieser Art von Angriff sehr häufig. Aber was ist genau SQL Injection?
Es ist die Schwachstelle, die entsteht, wenn man einen Angreifer die Möglichkeit gibt, einen Einfluss auf die SQL-Abfragen zu haben. Diese Abfragen werden dann zu dem Back-End-Datenbank gesendet. Wenn SQL zum Anzeigen von Daten auf einer Webseite dient ,ist es üblich, Netzbenutzer ihre eigene Suche Werte zu schreiben. Da SQL-Anweisungen sind nur Text, es ist leicht, mit einem kleinen Stück des Computercodes, SQL-Anweisungen dynamisch zu ändern, um den Anwender mit ausgewählten Daten zur Verfügung zu stellen. Die besten Beispiele dafür sind Webseite Fähigkeiten wie Kontaktformulare, Anmeldeseiten, Supportanfragen, Suchfunktionen oder Feedback. Alle sind anfällig für SQL Injectionsangriff. Ihre Website ist in unmittelbarer Gefahr, wenn Ihre Firma Daten von hohem Wert speichert oder wenn Ihre Firma oder Organisation ist, die in einem hart umkämpften Feld des Unternehmens existiert. Datenbanken steuern viele Webseite und fast alle Websites laden Input von Besuchern, deshalb sind so viele Webformulare anfällig. SQL Injection ist seit Jahren die häufigste Form der Webseiteangriff.

Falsch kodierten Formen ermöglichen einen Hacker sie als Einstiegspunkt für Ihre Datenbank zu benutzen. Web-basierte Formulare müssen einige Zugriff auf Ihre Datenbank von Daten erlauben. Diese Art von Angriff umgeht Firewalls und Endpunkt. Dann werden die Daten in der Datenbank sichtbar und der Zugriff auf andere Datenbanken auf demselben Server oder einem anderen Server im Netzwerk möglich. Der Angreifer kann die Syntax und Funktionen von SQL selbst beaufsichtigen, weil er beeinflusst, was zum Datenbank überliefer wird. SQL-Injection ist nicht eine Schwachstelle, die ausschließlich Webanwendungen betrifft. Angreifbar ist jeder Code, der Eingangssignale von einer nicht vertrauenswürdigen Quelle annihmt und verwendet dann die Eingänge dynamischer SQL-Anweisungen zu formen. Jede Prozedur mit SQL-Statements könnte potenziell angreifbar sein, weil die verfügbaren Methoden für den Aufbau und die Nutzungvon von Datenbanken viele Optionen bieten.

"The primary form of SQL injection consists of direct insertion of code into parameters that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata" (SQL Injection

Attacks and Defense S.7). Die primäre Form der SQL-Injection besteht aus der direkten Einfügen von Code in Parametern, die zusammen mit SQL-Befehlen verbunden und ausgeführt werden. Eine nicht so direkte Angriff spritzt malicious Code in Zeichenfolgen, die bestimmt für die Lagerung in einer Tabelle oder als Metadaten sind. Wenn ein Angreifer in der Lage ist zu ändern, um eine SQL-Anweisung, die ausgeführt wird, mit den gleichen Rechten wie der Benutzer der Anwendung. Der Erfolg des Angriffs hängt stark von der zugrundeliegenden Datenbank und vernetzten Systemen unter Beschuss ab. Manchmal braucht man viel Geschick und Beharrlichkeit, um eine Sicherheitsanfälligkeit zu seinem vollen Potential anzuwenden.

4. SQL Injection Beispiel

Jedes Mal, wenn ein Besucher Daten in Formular auf einer Webseite einträgt, wird eine SQL-Abfrage erzeugt und an Ihre Datenbank geliefert. Im Falle eines einfachen Login-Formular werden Benutzername und Passwort in der Datenbank ausgeliefert und wenn gültig, kann der Benutzer zugreifen (oder auch nicht). Egal wie einfach das Formular oder Webprozess ist,ist der Zugriff auf die Datenbank erforderlich, und wird eine Antwort erwartet.

Mit Hilfe SQL Injection wird ein Hacker versuchen, eine speziell gestaltete SQL Befehle in ein Formularfeld statt der erwarteten Informationen einzubringen. Die Absicht ist, eine Antwort von der Datenbank zu bekommen, damit der Hacker der Aufbau einer Datenbank wie Tabellennamen versteht. Der nächste Schritt wäre die Daten aufzurufen und in den wichtigsten Tabellen anzuzeigen. Der dritte Schritt ist ungefähr den Zugriff auf die Datenbank zu entdecken und der Sicherheitseinstellungen auf einem Server zu verändern. Das würde dem Hacker administrativen Zugriff erlauben.

Das folgende Beispiel (Java) ist unsicher, und würde einem Angreifer ermöglichen, Code in die Abfrage zu verpressen, die in dir Datenbank ausgeführt werden.

```
String query = "SELECT Username, Password
                From users
                Where Username'" + Username + "' AND
                      Password='" + Password+ "' ";

try {
       Statement statement = connection.createStatement( … );
       ResultSet results = statement.executeQuery( query );
}
```

Sie werden bemerken, dass Benutzereingabe erforderlich ist, um diese Abfrage auszuführen. Der Interpreter wird den Befehl abhängig von der Eingänge, die für die Felder für Benutzername und Kennwort sind, ausführen.Wenn ein Angreifer bietet 'oder 0=0' wie der Benutzername und das Kennwort, dann werden die Abfrage abgebaut als :

```
String query = "SELECT Username Password
                FROM user
                WHERE Username = " or 0=0"
                "' AND Password=" or 0=0"'";
```

Da unter allen Umständen Null gleich Null wird, wird die Abfrage alle Datensätze in der Datenbank erfüllt. Dies geschieht, weil die Ergebnisse in der Erklärung angehängt oder Operanden der Abfrage immer true zurückgeben, das heißt, 1 wird immer gleich 1. Auf diese Weise können autorisierte Benutzer wichtige Informationen anzeigen.
Das einfache Beispiel zeigt, wie ein Angreifer eine dynamisch erstellte SQL-Anweisung manipulieren kann. Die Anweisung wird von der Eingabe gebildet, der noch nicht validiert ist.

5. Folgen von dem Angriff

Hacker gewinnt administrativen Zugriff zu dem Server. Das bedeutet, dass man effektiv verloren alle Daten auf diesem Server verloren hat. Auf diese Weise können SQL Injection den Zugriff auf alle Unternehmen oder persönliche Daten ermöglichen. Eine gesamte Tabelle könnte dauerhaft mithilfe einer einzigen SQL-Befehl gelöscht werden. Aber eine kompliziertere SQL-Injektionsangriff könnte massive Korruption von großen Datenbanken betreffen und sogar Vernichtung von Sicherungskopien. “The impact of unauthorized disclosure of confidential information can range from violation of the Data Privacy Act to the jeopardizing of national security. Unauthorized, unanticipated, or unintentional disclosure could result in loss of public confidence, embarrassment, or legal action against the organization” (Fundamentals of Database Systems, S.737). Zufolge dem Buch “Fundamentals of Database Systems” gibt es verschiedene Arte von Folgen der unbefugten Offenlegung. Zum Beispiel Verlust des öffentlichen Vertrauens, Verlegenheit, oder rechtliche Schritte gegen das Unternehmen.
Die Kosten sind auch nicht zu vergessen. “LinkedIn spent nearly $1 million investigating and unraveling the theft of 6.5 million passwords in June and plans to spend up to $3 million more updating security on its social networking site” (Breach clean-up cost LinkedIn nearly

$1 million, Online Quelle). Im Juni 2012 wurden mehr als 6,5 Millionen Kennwörter gestohlen. Der Angriff auf LinkedIn führte zu Wiederherstellungskosten von 1 Million Dollar und auch 3 Millionen für neue und notwendige Upgrades zur Abwehr zukünftiger Angriffe.

6. SQL Injection Prävention Methode

SQL ist die Standardsprache für den Zugriff auf Microsoft SQL Server, Oracle, MySQL, und Informix (sowie andere) Datenbank Server. Die meisten Webanwendungen brauchen mit einer Datenbank zu interagieren, und die meisten Webanwendungen Programmiersprachen wie ASP, C#, Java und PHP bereitstellen diese Möglichkeit. SQL Injection Schwachstellen treten am häufigsten auf, wenn der Web Application Developer nicht gewährleistet, dass Werte, die von einem Web-Formular, Cookie, Eingangsparameter und so weiter validiert werden, bevor sie SQL-Abfragen auf einem Datenbankserver ausgeführt werden.
Firewalls bieten wenig oder keine Verteidigung gegen Angriffe durch SQL Injection, weil Websites ständig Zugriff auf die Datenbank erfordern. Eine Website ist öffentlich und Firewalls müssen so eingestellt sein, dass jeder Besucher der Website den Zugriff auf die Datenbank hat. Antivirus Programme sind ebenso wirkungslos gegen Angriffen durch SQL Injection. Sie sollen erkennen und stoppen Sie eine völlig andere Art von eingehenden Daten.

6.1. Parameterized Queries

Dynamic string building ist eine Programmiertechnik, die es Entwicklern ermöglicht, SQL-Anweisungen dynamisch zur Laufzeit zu erstellen. Entwickler können für allgemeine Zwecke flexible Anwendungen mit Hilfe von dynamischem SQL erstellen. Eine dynamische SQL-Anweisung ist zur Ausführungszeit für die unterschiedliche Bedingungen erzeugen unterschiedliche SQL-Statements aufgebaut.
Es kann nützlich sein, die Entwickler diese Aussagen dynamisch zu konstruieren, wenn Sie während der Laufzeit entscheiden müssen, welche Felder zurückzuholen. Zum Beispiel SELECT-Anweisungen, die verschiedenen Kriterien für Abfragen besitzen und vielleicht verschiedene Tabellen zur Abfrage auf unterschiedlichen Bedingungen erstellen.
Problem: Eines der Probleme mit dem Aufbau dynamischer SQL ist, wenn der Code nicht validiert ist oder die Eingaben bevor der Übergabe an den dynamisch erstellten Aussage nicht verschlüsselt werden. Dann könnte der Angreifer SQL-Anweisungen als Eingabe für das Programm eingeben und seine SQL-Statements an die Datenbank übergeben und ausführen.

Lösung: Entwickler können das gleiche Ergebnis durch eine sicherere Methode erzielen, wenn sie parametrisierte Abfragen benutzen. Parametrisierte Abfragen sind Abfragen, die eine oder mehrere eingebettete Parameter in der SQL-Anweisung haben. Parameter können zu diesen Abfragen zur Laufzeit übergeben werden. Parameter mit eingebetteten Benutzereingaben werden nicht als ausführbare Befehle interpretiert, und es würde keine Möglichkeit für Code Injection geben. Vorbereitete Anweisungen sorgen dafür, dass ein Angreifer nicht in der Lage ist, die Absicht einer Abfrage zu ändern, auch wenn die SQL Befehle durch einen Angreifer eingefügt sind.

Man kann in dem Beispiel unten sehen,dass wenn ein Angreifer zur Eingabe der userID von Tom' oder '1'='1 schreibt, wird den parametrisierten Abfrage nicht anfällig sein und wird stattdessen nach den Benutzernamen suchen, die buchstäblich gleich der gesamte String "Tom' oder '1'='1" sind.

```
String custname = request.getParameter("Username");
// perform input validation to detect attacks
 String query = "SELECT Username,Password FROM users WHERE Username = ? ";

 PreparedStatement pstmt = connection.prepareStatement( query );
 pstmt.setString( 1, custname);
 ResultSet results = pstmt.executeQuery( );
```

Vorteile: Parametrisierte Abfragen zwingen die Entwickler zuerst den SQL-Code zu definieren, und dann jeder Parameter zum der Abfrage zu geben. Das Coding Style ermöglicht die Datenbank zwischen Code und Daten zu unterscheiden, unabhängig von dem User Input. Diese Methode der Einbettung der Parameter in SQL ist effizienter und viel sicherer als Dynamic string building und Ausführen von SQL-Anweisungen mit String-Techniken.

Nachteile: Parametrisierte Abfragen (Prepared Statement) können wirksam gegen eine Klasse von SQL Injection Schwachstelle. In keiner Weise könnten sie alle andere Klassen des Angriffs verhindern.

6.2. Stored Procedures

Gespeicherte Prozeduren sind in der Datenbank gespeicherte Programme. Sie haben die gleiche Wirkung wie die Verwendung von Prepared Statements. Sie benötigen die Entwickler nur SQL-Anweisungen mit Parametern zu bilden. Der Unterschied zwischen Prepared Statements und Stored Procedures ist, dass die SQL-Code für eine gespeicherte Prozedur definiert ist und in der Datenbank selbst gespeichert, und dann aus der Anwendung rief.

Problem: Dynamisches SQL nimmt an der Anwendung oder an der Datenbank teil, und wird an die Datenbank zur Ausführung gesendet.Dann müssen alle Daten innerhalb der Datenbank, die lesbar, beschreibbar oder updatefähig sein müssen , zugänglich für die Benutzer der Datenbank sein. Das wird verwendet, um auf die Datenbank zuzugreifen. Deshalb, wenn eine SQL Injection Problem auftritt, kann der Angreifer möglicherweise Zugriff auf alle Informationen in der Datenbank bekommen.

Lösung: Durch die Verwendung von gespeicherten Prozeduren können diese Situation verändert werden. In diesem Fall erstellt man gespeicherte Prozeduren, die zur Durchführung aller von den Zugriff auf die Datenbank der Anwendung angepasst werden. Die Datenbank Benutzer, die die Anwendung verwenden, um Zugriff auf die Datenbank zu erhalten, wird die Erlaubnis gegeben, die gespeicherten Prozeduren auszuführen. Die Anwendung benötigt sie, aber keine anderen Daten haben die Berechtigungen innerhalb der Datenbank.

Das folgende Codebeispiel zeigt CallableStatement, Java Implementierung der gespeicherten Prozedurschnittstelle zur Ausführung der gleichen Datenbank Abfrage. Der "sp_getAccountBalance" gespeicherte Prozedur wird in der Datenbank vordefiniert.

```
String custname = request.getParameter("Username"); // This should be
validated
 try {
      CallableStatement cs = connection.prepareCall("{call
sp_getPassword(?)}");
      cs.setString(1, custname);
      ResultSet results = cs.executeQuery();
      // … result set handling
 } catch (SQLException se) {
      // … logging and error handling
 }
```

Vorteile: Sie sind einfach zu schreiben, und leichter zu verstehen als dynamische Abfragen. Gespeicherte Prozeduren können sehr nützlich für die Milderung der Schwere einer möglichen Sql injection-Schwachstelle sein. Es ist möglich, Access Controls auf der Datenbankebene bei der Verwendung gespeicherter Prozeduren auf die meisten Datenbanken zu konfigurieren. Das ist wichtig, weil es bedeutet, dass, wenn einem ausnutzbaren SQL Injection Problem gefunden wird, kann der Angreifer nicht in der Lage sein, Zugang zu sensiblen Informationen innerhalb der Datenbank zu haben.

Nachteile: “Some developers believe that if they do not build and execute dynamic SQL statements and instead only pass data to stored procedures as parameters, their code will not be vulnerable. However, this is not true, as stored procedures can be vulnerable to SQL injection also” (SQL Injection Attacks and Defense S.99). Einige Entwickler glauben, dass

wenn sie dynamische SQL-Anweisungen nicht ausführen und stattdessen nur Daten für gespeicherte Prozeduren als Parameter benutzen, deren Code nicht anfällig würde. Das ist jedoch nicht wahr, weil gespeicherte Prozeduren auch angreifbar zu SQL Injection sein können. Parametrisierte Abfragen (Prepared Statement) können wirksam gegen eine Klasse von SQL Injection Schwachstelle. In keiner Weise könnten sie alle andere Klassen des Angriffs verhindern. Es gibt auch mehrere Fälle, in denen gespeicherte Prozeduren Risiko erhöhen können.

7. Zusammenfassung

“84 percent of respondents say the most common gateway attack experienced by their organization over the past 12 months is an SQL injection, followed by cross site scripting (23 percent of respondents) and cross site request forgery (at 18 percent). SQL injection has surfaced as the no. 1 attack in 2015” (SQL injection has surfaced as the no.1 attack in 2015, Online Quelle). Die Forschung von Ponemon Institute zeigt, dass SQL Injection der häufigste Hackerangriff. SQL Injection ist ein Angriff , der benutzt wird, um Code durch Änderung Back-End SQL-Anweisungen durch Manipulation zu verwerten. Obwohl SQL Injection ein bekanntes Problem seit Jahren ist, gibt es mehrere Ursachen zu den Erhöhung der Angriffe. SQL Injections sind dann möglich, wenn Daten wie Benutzereingaben in den SQL-Interpreter gelangen. Einen SQL-Injektionsangriff besteht aus Insertion oder "Injektion" von einer SQL-Abfrage über den Eingang der Daten vom Client an die Anwendung. Webseite Fähigkeiten wie Kontaktformulare, Anmeldeseiten, Supportanfragen, Suchfunktionen oder feedback Felder sind alle anfällig zu Angriffen durch SQL-Injektion. Das Problem ist, dass alle diese Fähigkeiten Teile der heutigen Webseite sind. Möglicherweise kann ein Angreifer die Daten in der Datenbank manipulieren und möglicherweise Betriebssystembefehle auf dem Datenbankserver ausführen. Das führt zum Verlust des öffentlichen Vertrauens, der Verlegenheit, oder rechtliche Schritte gegen das Unternehmen. Deshalb muss man verschiedene Methode wie Parameterized Queries und Stored Procedures benutzen, um diesen Angriff zu verhindern.

8. Literaturverzeichnis

Printquellen

Justin Clarke (2009): SQL Injection Attacks and Defense, Second Edition. Burlington

Ramez Elmasri/ Shamkant B. Navathe (2003): Fundamentals of Database systems 4th Edition. Boston

Bob Beauchemin/ Dan Sullivan (2006): Das SQL-Server-2005-Entwicklerbuch. München

Online-Quellen

John Fontana (August 3, 2012): Breach clean-up cost LinkedIn nearly $1 million, another $2-3 million in upgrades, verfügbar über: http://www.zdnet.com/article/breach-clean-up-cost-linkedin-nearly-1-million-another-2-3-million-in-upgrades/ (26.03.2016)

o.A (11.12.2015): SQL injection has surfaced as the no. 1 attack in 2015, verfügbar über: https://www.helpnetsecurity.com/2015/12/11/sql-injection-has-surfaced-as-the-no-1-attack-in-2015/ (24.03.2016)

Arved Graf von Stackelberg (23.12.2015): SQL Injection als Einfallstor: So können Unternehmen sich schützen, verfügbar über: http://www.tecchannel.de/sicherheit/management/3201640/sql_injection_als_einfallstor_so_koennen_unternehmen_sich_schuetzen/ (25.03.2016)