Little Bobby Tables – Wie Dual-Use-Tools Ihr Unternehmen schützen können

SOURCEPARK GmbH

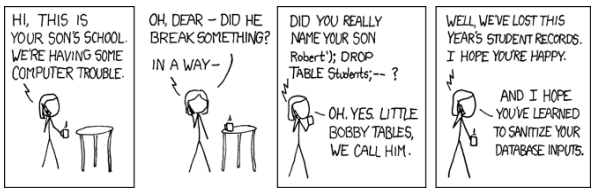
Zusammenfassung

Der Beitrag zeigt auf, welche Gefahren durch Sicherheitslücken in Datenbanksystemen und die sie nutzenden (Web-) Applikationen entstehen und wie die Verwendung sog. ”Dual-Use-Tools” ein Unternehmen dabei unterstützen kann, diese Gefahren zu identifizieren und abzuwenden.

Schlüsselwörter
Datenbank – Kompromittierung – Injection – SQL-Injection – Dual-Use – Hacking

Little Bobby Tables

Kennen Sie die Geschichte von „Little Bobby Tables“? „Bobby Tables“ ist ein Synonym, das in der IT-Welt verwendet wird, um auf die Gefahren von SQL-Injection-Angriffen hinzuweisen und für das Sanitizing[1] von Nutzereingaben zu sensibilisieren (siehe Abbildung [9]).



In der obigen Abbildung hat die Mutter ihren Sohn „`Robert'); DROP TABLE students;--`“ genannt. Dieser ungewöhnliche Name, eingegeben in das Datenbanksystem der Schule des Jungen, führte dazu, dass alle Aufzeichnungen über Schüler (in der Tabelle `students`) gelöscht wurden.

Es handelt sich hierbei um ein klassisches Beispiel für eine erfolgreiche SQL-Injection-Attacke (kurz: SQL-Injection oder auch SQLi). Von einer SQL-Injection spricht man dann, wenn es einem Angreifer gelingt, über eine von ihm kontrollierte Eingabe (z.B. das Namensfeld in einem Formular) einen Datenbankbefehl an das Datenbanksystem zu senden, das diesen dann ausführt. Im obigen Beispiel wäre dies der Befehl `DROP TABLE students;`, der die Tabelle `students` löscht.

In diesem Artikel geht es um SQL-Injections und wie Sie sich in Ihrem Unternehmen davor schützen können.

Newsflash: Datenleck bei ACME GmbH

Die fiktive Firma ACME GmbH ist ein Dienstleistungsunternehmen, das im Auftrag Kundendaten anderer Firmen verarbeitet. Die Firma hat ein eigenes, nach außen abgeschottetes Netzwerk, in dem die Daten der Kunden gesichert sind. Lediglich ein Server ist aus dem Internet erreichbar und liefert die Website der ACME GmbH an Interessenten aus. Die Website wird mit einem Content Management System (CMS) betrieben, das eine Datenbank verwendet, um die Inhalte der Website zu verwalten, und bietet dem Besucher neben statischen Informationen u.a. auch die Möglichkeit, eine Suche durchzuführen.

An einem Montagmorgen beginnen die Mitarbeiter der ACME GmbH gerade ihren Arbeitstag und wollen auf die Kundendaten eines wichtigen Kunden zugreifen, da fällt ihnen auf, dass sämtliche Daten aus dem System gelöscht wurden. Schlimmer noch, am nächsten Tag ist in den Medien zu lesen, dass es ein Sicherheitsleck bei der

[1] siehe z.B. `https://www.owasp.org/index.php/Data_Validation`

ACME GmbH gab und dass sämtliche Daten sämtlicher Kunden auf dubiosen Internettauschbörsen veräußert werden. Der wirtschaftliche und Image-Schaden für die ACME GmbH ist katastrophal.

Nach Tagen der Analyse wird klar: ein Angreifer konnte eine SQLi-Schwachstelle des Suchformulars ausnutzen, um die Daten zunächst abzuziehen und danach von den Firmenservern zu löschen. Der im Internet sichtbare Server, der die Datenbank für die Website enthält, konnte in diesem Szenario durch den Angreifer übernommen werden und diente als Sprungbrett in das Firmennetzwerk.

Dieses drastische Szenario soll verdeutlichen, welche Folgen eine SQLi-Attacke auf ein Unternehmen haben kann. Selbst weniger pessimistische Szenarien, wie z.B. das Defacement[2] der Unternehmenswebsite, können ein Unternehmen gegenüber seinen Kunden und der Öffentlichkeit stark belasten.

Feindliche Übernahme

Betrachen wir die Möglichkeiten, die eine SQLi-Attacke einem Angreifer bietet. Vielleicht haben Sie bereits von SQLi-Angriffen gehört und denken sich, dass diese Ihrem Unternehmen nicht besonders schaden können, da damit doch ohnehin nur Informationen in der Datenbank angezeigt, verändert oder gelöscht werden können. Sie haben schließlich ein stets aktuelles Backup der Datenbank. Hier liegen Sie bedauerlicherweise falsch.

Datenbankmanagementsysteme (DMBS) sind hochkomplexe Software-Produkte, die über einen teilweise beachtlichen Funktionsumfang verfügen, der mitunter weit über die bloße tabellarische Verwaltung von Daten hinaus geht. Viele DBMS erlauben z.B. das Schreiben von Dateien oder sogar das Ausführen von Systembefehlen, weil diese Funktionen in bestimmten Anwendungsfällen gebraucht werden. Als Beispiele seien hier die Funktion `xp_cmdshell`, die es Microsofts MSSQL-Server erlaubt, einen Systembefehl auszuführen, und `INTO OUTFILE`, die es erlaubt, Daten in eine Datei zu schreiben, erwähnt. Beide Funktionen haben gewiss eine Daseinsberechtigung, bilden aber zeitgleich erhebliche Sicherheitslücken.

Um es kurz zu machen: eine erfolgreiche SQL-Injection kann zur vollständigen Kompromittierung eines Datenbankservers führen, die es einem Angreifer u.U. erlaubt, tiefer in das Netzwerk vorzudringen.

Anatomie

Zum besseren Verständnis wollen wir analysieren, wie es zu einer SQLi-Schwachstelle in einer Software kommen kann, bevor wir tiefer in die Materie eintauchen.

Stellen Sie sich vor, Sie entwickeln eine Webapplikation, die es dem Benutzer erlaubt, eine Suche durchzuführen. Der Benutzer tippt den Suchbegriff ein, dieser wird Bestandteil einer SQL-Abfrage, die an das DBMS gestellt wird. Der Benutzer bekommt die entsprechenden, von der Datenbank gelieferten, Werte angezeigt. Um z.B. Personen zu suchen, könnten Sie den folgenden Ausdruck verwenden: „`SELECT * FROM persons WHERE name = '$name';`“. Der Wert für `$name` stammt aus dem Suchfeld, wird also durch den Anwender kontrolliert.

Gibt ein Benutzer einen Namen ein, wird dieser Wert in die Abfrage eingefügt und in der Tabelle `persons` nach diesem Namen gesucht. Eine Überprüfung des eingegebenen Werts findet an dieser Stelle nicht statt. Stellen wir uns nun vor, dass ein Angreifer als Suchbegriff den Wert `';DROP TABLE persons--` eingibt.

Der SQL-Ausdruck ändert sich nun zu `SELECT * FROM persons WHERE name = '';DROP TABLE persons--';`. Das Semikolon trennt in der SQL-Syntax Befehle voneinander. Für das DBMS bedeutet diese Abfrage, dass zunächst alle Personen, deren Name leer ist (`WHERE name = ''`) ausgewählt und danach die Tabelle `persons` gelöscht werden soll (`DROP TABLE persons`). Vielleicht erkennen Sie die Angriffssyntax aus dem einführenden Beispiel wieder – es handelt sich hier um ein klassisches „Bobby Tables“-Beispiel.

[2] https://de.wikipedia.org/wiki/Defacement

Um noch etwas tiefer in die Materie einzusteigen, betrachten wir, wie sich die Abfrage-URL an den Server bzw. die (Web-) Applikation mit dem bösartigen Suchparameter verändert und sehen uns erneut die damit zusammenhängende Datenbankabfrage an. Eine legitime Suche, z.B. nach dem Namen „mustermann“, führt zu der URL `www.webapp.xyz/search?name=mustermann` und der Datenbankabfrage `SELECT * FROM persons WHERE name = 'mustermann';`. Der bösartige Suchbegriff `';DROP TABLE persons--` führt hingegen zur URL `www.webapp.xyz/search/?name=';DROP TABLE persons--` und zur Datenbankabfrage `SELECT * FROM persons WHERE name = '';DROP TABLE persons--';`[3] – wie Sie bereits wissen, wird damit die Tabelle *persons* gelöscht.

There's an app for that

SQLi-Schwachstellen sind tückisch – sowohl für den Entwickler, als auch für den Angreifer. SQL ist eine mächtige Abfragesprache, die unzählige Kombinationsmöglichkeiten von Abfragebefehlen erlaubt und z.T. sehr komplex werden kann. Es ist also für den Entwickler denkbar problematisch, sämtliche Angriffsszenarien auf jede SQL-Abfrage zu bedenken. So werden potentielle Einfallstore für Angreifer schnell übersehen; doch auch ein Angreifer muss u.U. zahllose sog. „Payloads“, also bösartige Eingaben, durchprobieren, bis eine SQLi-Schwachstelle genutzt werden kann.

Auf Seiten des Angreifers existieren zahlreiche Werkzeuge, die diesen Prozess nicht nur erleichtern, sondern nahezu vollständig automatisieren – ein tiefgreifendes Verständnis für das zugrundeliegende DBMS ist längst keine Voraussetzung mehr für einen erfolgreichen SQLi-Angriff!

Dieser Artikel verwendet im Folgenden beispielhaft das Open-Source-Tool *sqlmap* ([7]), ein sehr bekanntes und leicht zu bedienendes Werkzeug zur automatisierten Erkennung und Ausnutzung von SQLi-Schwachstellen.

[3] Zur besseren Lesbarkeit wurde auf das Encoding der Sonderzeichen verzichtet.

Zum Angriff!

Ein Angriff mit *sqlmap* auf eine Webapplikation ist denkbar einfach. Nach einer Initialkonfiguration (*sqlmap* liefert einen Assistenten mit) beginnt der Angriff automatisch. Anhand der vom Angreifer angegebenen URL ermittelt das Tool, welche der Abfrageparameter u.U. verwundbar sind und testet jeden dieser Parameter auf zahlreiche Schwachstellen. Wird eine Schwachstelle entdeckt, zeigt *sqlmap* diese auf und liefert darüber hinaus auch Informationen über das verwendete DBMS und ggf. auch die eingesetzte Webtechnologie. Die Abbildung auf der folgenden Seite zeigt einen Ausschnitt aus einer erfolgreichen SQLi-Attacke mittels *sqlmap*. Zunächst identifiziert das Tool den verwundbaren Parameter (1) – hier `id`. Daraufhin testet es die Art der Verwundbarkeit und verschiedene Angriffsvektoren (Payloads) (2). Zuletzt werden bei (3) die Parameter des Servers ausgegeben – dieser setzt PHP 5.3.1, Apache 2.2.1.4 und MySQL 5.0 ein.

Da das Tool nun weiß, dass das Ziel verwundbar ist, kann es in einer zweiten Phase den eigentlichen Angriff starten. Bei (4) erkennt *sqlmap*, dass die Spalte `password` evtl. Passwörter (bzw. deren Prüfsummen) enthalten könnte. Es beginnt sofort damit, die Passwörter mit einer Wortliste zu vergleichen und meldet auch schnell Erfolg (5). Abschließend gibt das Tool bei (6) die Tabelle `users` vollständig aus und fügt selbständig die ermittelten Klartext-Passwörter ein.

Der Angriff selbst dauert nur wenige Sekunden (insgesamt sendet *sqlmap* nur 23 HTTP-Requests an den Server) und liefert dem Angreifer sämtliche Zugangsdaten zur beispielhaften Webapplikation.

Payload

Das obige Beispiel zeigt nur einen kleinen Teil der Basisfunktionalität von *sqlmap*. Über die Hilfefunktion des Tools (`sqlmap -hh`) findet der Nutzer schnell heraus, dass *sqlmap* sowohl eigene Funktionen anbietet, mit der das Betriebssystem eines SQLi-anfälligen Datenbankservers gekapert werden kann, als auch eine ausgezeichnete In-

tegration der wohl bekanntesten Penetrationstest-Software *metasploit* [3]. Für den Angreifer bedeutet dies, dass der Datenbankserver, der eine SQLi-Schwachstelle aufweist, mit großer Wahrscheinlichkeit übernommen werden kann[4].

Mitigation

Nachdem wir nun die Seite des Angreifers untersucht und herausgefunden haben, dass die Ausnutzung einer SQLi-Schwachstelle einer Webapplikation erstaunlich einfach von der Hand geht, wollen wir im nächsten Schritt eruieren, wie Sie Ihr Unternehmen gegen solche Angriffe schützen können. Dabei fokussieren wir uns nicht auf Programmiertechniken[5], sondern darauf, wie die hier beschriebenen Dual-Use-Tools dazu eingesetzt werden können, das Unternehmensnetzwerk und damit die Wertschöpfungskette abzusichern. Im Folgenden wird darauf verzichtet, bestimmte Programmiersprachen oder Frameworks zu erwähnen, die besonders anfällig für SQLi-Schwachstellen sind; allgemein gilt aber immer, dass Applikationen, die über das Netzwerk oder gar das Internet erreichbar sind, möglichst nicht mit veralteten Technologien arbeiten sollten.

Je älter z.B. die PHP- oder Java-Version ist, mit der eine Applikation betrieben wird, desto wahrscheinlicher ist es, dass in ihr Sicherheitslücken wie SQLi-Schwachstellen klaffen, die nicht (mehr) behoben werden; dies gilt insbesondere dann, wenn diese Versionen bereits ihr End-of-Life erreicht haben und es keine Sicherheitsupdates mehr für sie gibt. Als erste Maßnahme sollten Sie deshalb in Ihrem Unternehmen prüfen, ob es Applikationen gibt, die über das Netzwerk oder Internet erreichbar sind und veraltete Software einsetzen – und ob es möglich ist, diese zu aktualisieren.

[4] sqlmap bietet an, eine *meterpreter*-Session aufzubauen. *meterpreter* ist die Post-Exploitation-Software, die mit *metasploit* ausgeliefert wird und ermöglicht das vollständige Übernehmen eines Computersystems uvm. (siehe [4])

[5] Gemeint ist hier z.B., Benutzereingaben zu filtern oder zu escapen

Dual-Use

Viele sog. „Hacker-Tools“ können auch dazu eingesetzt werden, die Infrastruktur eines Unternehmens gegen genau die Angriffe abzusichern, die mit ihnen durchgeführt werden – so auch *sqlmap*. Diese doppelte Nutzbarkeit nennt man *Dual-Use*, also den dualen Nutzen sowohl für potentiell kriminelle Angriffe, als auch für legitime Verteidigung.

Doch wie kann man solche Tools sinnvoll in die Unternehmensstruktur einbinden?

Ein Leitspruch der IT-Security lautet „Sicherheit ist ein Prozess – kein Zustand“. Genau dieser Ansatz muss auch verfolgt werden, um Dual-Use-Tools im Unternehmen sinnvoll einzusetzen, da sich die eingesetzten Applikationen kontinuierlich verändern und bei jeder Änderung neu analysiert werden müssen. Kehren wir zum Beispiel der ACME GmbH zurück. Bevor der Administrator des Unternehmens die Website auf den aus dem Internet erreichbaren Server überspielt und damit die Vorgänge in Gang gesetzt hat, die der fiktiven Firma zum Verhängnis wurden, hätte er die Webapplikation mit Tools wie *sqlmap* auf potentielle Schwachstellen hin untersuchen müssen, um dem Desaster-Szenario vorzubeugen. Sehen wir uns im Folgenden an, wie die Schwachstelle, die durch den fiktiven Angreifer ausgenutzt werden konnte, durch den Einsatz von *sqlmap* bereits im Vorfeld hätte erkannt werden können.

How-To

Die folgenden Schritte zeigen exemplarisch, wie eine SQLi-Schwachstelle mittels *sqlmap* aufgedeckt werden kann.

Identifikation – Im Beispiel der ACME GmbH war es die Suchfunktion, die die SQLi-Schwachstelle aufwies. Im ersten Schritt ist zu klären, welche Benutzereingaben an das DBMS weitergereicht werden und wie dies geschieht. Nehmen wir exemplarisch an, dass der Suchbegriff (Nachname einer Person) als URL-Parameter `name` übergeben wird. Die potentiell verwundbare URL würde also `http://acme.xyz/site/search/?name=mustermann` lauten (*sqlmap* analysiert die Parameter selbständig).

Test – Ist eine potentiell verwundbare URL gefunden wird sie *sqlmap* über den Parameter `-u` bekannt gemacht und eine erste Analyse gestartet: `sqlmap -u "http://acme.xyz/site/search/?name=mustermann"`. Findet *sqlmap* eine Schwachstelle, gibt es sie aus. Nun ist es an der Zeit, die Ausnutzbarkeit der Schwachstelle zu prüfen, indem *sqlmap* mitgeteilt wird, die Datenbanken auf dem Server auszulesen: `sqlmap -u "http://acme.xyz/site/search/?name=mustermann" --dbs`. *sqlmap* gibt die auf dem Datenbankserver verfügbaren Datenbanken aus.

Meldung – An dieser Stelle kann der Test durch den Administrator bereits beendet und die SQLi-Schwachstelle gemeldet werden, da die (Web-) Applikation definitiv verwundbar und damit ein Sicherheitsrisiko für das Unternehmen ist. Der Vollständigkeit halber wird dieses Beispiel jedoch fortgeführt.

Exploit – Da die verfügbaren Datenbanken nun bekannt sind, wählt der Administrator eine aus (hier: `searchdb`) und lässt sich die verfügbaren Tabellen der Datenbank mittels `sqlmap -u "http://acme.xyz/site/search/?name=mustermann" -D searchdb --tables` anzeigen. *sqlmap* listet die Tabellen der Datenbank `searchdb` auf. Der Administrator wählt eine Tabelle aus (hier: `users`) und lässt sie durch *sqlmap* anzeigen: `sqlmap -u "http://acme.xyz/site/search/?name=mustermann" -D searchdb -T users --dump`. *sqlmap* erkennt, dass die Tabelle potentielle Passwort-Prüfsummen erkennt und fragt beim Administrator nach, ob versucht werden soll, diese zu brechen. Der Administrator bestätigt dies und *sqlmap* fragt die Methodik ab (es wird ein Wortlisten-basierter Brute-Force durchgeführt). Kurze Zeit später sind dem Administrator die Passwörter bekannt[6].

Mitigation – Nachdem die SQLi-Schwachstelle nun identifiziert und im Rahmen eines Proof-of-Concept ausgenutzt wurde, ist es an der Zeit, sie zu beheben, bevor die (Web-) Applikation auf den Produktivserver überspielt und damit auch poten-

[6] *sqlmap* wird mit einer Wortliste geliefert, die viele Standardpasswörter wie `password`, `letmein` oder `abc123` sehr leicht brechen kann

tiell Angriffen aus dem Internet ausgesetzt wird. Erinnern wir uns an die Datenbankabfrage, die die SQLi-Schwachstelle in die (Web-) Applikation eingebracht hat: „`SELECT * FROM persons WHERE name = '$name';`“. Ohne auf bestimmte Programmiersprachen einzugehen, kann diese Abfrage auf mindestens zwei Arten abgesichert werden. Zunächst sollte der Parameter, der an die Datenbank weitergereicht wird, durch die Applikation überprüft werden. Ist er überdurchschnittlich lang? Enthält er Sonderzeichen, Schlüsselwörter oder auch Zahlen? Handelt es sich um den korrekten Datentyp? Ist diese Prüfung erfolgt, sollte darauf verzichtet werden, das SQL-Statement manuell zu konkatenieren, wie es im obigen Beispiel der Fall ist. Stattdessen ist es ratsam, die Features der eingesetzten Programmiersprache und Datenbankschnittstelle zu nutzen und das SQL-Statement über ein *Prepared Statement*[7] aufzubauen, das vom DBMS bzw. dessen Programmierschnittstelle verwaltet und abgesichert wird.

Lessons learned

Das obige simplifizierte Beispiel soll aufzeigen, dass die Entdeckung einer SQLi-Schwachstelle mit den richtigen Werkzeugen weder besonders schwer, noch besonders zeitaufwändig ist und zum Standard-Repertoire eines jeden Administrators zählen sollte. Es handelt sich dabei natürlich nur um eines von vielen Beispielen und nur eines von vielen Werkzeugen, die von Angreifern zur Übernahme einer Netzwerkinfrastruktur verwendet werden – und von den Verantwortlichen zu deren Schutz verwendet werden sollten.

Der Einsatz von Dual-Use-Tools in Ihrem Unternehmen kann darüber hinaus dazu beitragen, die an Ihr Unternehmen oder Sie persönlich gestellten Compliance-Anforderungen zu erfüllen und Sie ggf. gegen Vorwürfe der Fahrlässigkeit abzusichern (dies gilt u.U. auch für Verstöße gegen das Datenschutzgesetz).

[7] siehe z.B. `https://de.wikipedia.org/wiki/Prepared_Statement`

Handlungsempfehlung

Um die Netzwerkinfrastruktur Ihres Unternehmens zu schützen, sollten Sie die vermeintlichen „Hacker-Tools“ in Ihren Deployment- und Wartungsprozess aufnehmen. Tools wie *sqlmap* sind mächtige Werkzeuge, die schnell und effizient zahlreiche Schwachstellen aufdecken, die Ihrem Unternehmen zum Verhängnis werden könnten, wenn sie durch einen ausreichend motivierten Angreifer entdeckt und ausgenutzt werden. Stellen Sie sich die folgenden Fragen: Kenne ich sämtliche Software-Versionen und die eingesetzten Technologien, die in meinem Unternehmen über das Internet erreichbar sind? Weiß ich, welche Benutzereingaben an ein DBMS weitergereicht werden und wie diese abgesichert werden? Mit großer Wahrscheinlichkeit wird es Ihnen schwer fallen, beide Fragen mit einem sicheren „Ja“ zu beantworten. Dual-Use-Tools können Sie und Ihr Unternehmen dabei unterstützen, diese Fragen mit Sicherheit zu beantworten und gegen eine Vielzahl von Angriffen gewappnet zu sein.

Weiterführende Links

Natürlich kann ein kurzer Artikel nicht den gesamten Umfang des komplexen Themas *SQL-Injection* abdecken. Aus diesem Grund finden Sie unter den folgenden Links weiterführende Informationen:

SQL-Injection Wenn Sie mehr über SQL-Injection und SQL-Injection-Schwachstellen erfahren möchten, bieten sich [1], [6] und [5] als Ausgangspunkt Ihrer Recherchen an.

sqlmap Wenn Sie mehr über *sqlmap* erfahren möchten, empfehlen wir Ihnen die Hersteller-Website unter [7] und den Artikel unter [8].

Wenn Sie sich mit Webapplikationsschwachstellen im Allgemeinen beschäftigen möchten, ist die *Damn Vulnerable Web Application*, kurz *DVWA* [2], der perfekte Einstieg in die Materie. Sie können diese kostenlose Open-Source-Webapplikation auf einer Virtuellen Maschine betreiben (bedenken Sie bitte, die Netzwerkschnittstelle(n) mit NAT zu konfigurieren!) und verschiedenste Schwachstellen einer Webapplikation in Aktion erleben. Da es

sich bei DVWA um eine Open-Source-Applikation handelt, können Sie auch die Codestellen betrachten, die für die eigentlichen Schwachstellen verantwortlich sind; zu jeder Schwachstelle wird auch eine sichere Variante angeboten, so dass Sie auch direkt Maßnahmen ableiten können, wenn Sie eine entsprechende Schwachstelle in einer von Ihrem Unternehmen eingesetzten Software identifizieren.

Literatur

[1] HEISE.DE: *Giftspritze | heise Security*. http://www.heise.de/security/artikel/Giftspritze-270382.html.

[2] HTTP://WWW.DVWA.CO.UK/: *DVWA - Damn Vulnerable Web Application*. http://www.dvwa.co.uk/.

[3] METASPLOIT.COM: *Penetration Testing Software | Metasploit*. http://www.metasploit.com.

[4] OFFENSIVE-SECURITY.COM: *About the Metasploit Meterpreter - Metasploit Unleashed*. https://www.offensive-security.com/metasploit-unleashed/about-meterpreter.

[5] OWASP.ORG: *SQL Injection - OWASP*. https://www.owasp.org/index.php/SQL_Injection.

[6] SECHOW.COM: *Advanced SQL Injection - Integer base | OWASP Bricks*. http://sechow.com/bricks/docs/content-page-1.html.

[7] SQLMAP.ORG: *sqlmap: automatic SQL injection and database takeover tool*. http://sqlmap.org.

[8] TRUSTWAVE.COM: *Sqlmap Tricks for Advanced SQL Injection*. https://www.trustwave.com/Resources/SpiderLabs-Blog/Sqlmap-Tricks-for-Advanced-SQL-Injection/.

[9] XKCD.COM: *Exploits of a Mom*. http://xkcd.com/327/.