ЗАЩИТА WEB-ПРИЛОЖЕНИЙ ОТ SQL-ИНЪЕКЦИЙ

А. Н. Долбик
Кафедра ИТАС, Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: alexandr.dolbik@gmail.com

Этот доклад раскрывает популярные способы внедрения SQL-инъекций. Рассматриваются популярные методы защиты веб-приложений. Описывается и проверяется опытным путем собственная система защиты веб-приложения, основанная на лучших решениях рассмотренных методов.

Введение

SQL-инъекции (англ. SQL injection) или внедрение SQL-кода — один из самых распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода. Внедрение SQL, в зависимости от типа используемой СУБД и условий внедрения, может дать возможность атакующему выполнить произвольный запрос к базе данных. Атака путем внедрения SQL-кода возможна при некорректной обработке входных данных, используемых в SQL-запросах.

I. Виды SQL-инъекции и способы их внедрения

1. Использование символа «кавычка» и комментирование остального кода.
   Практически на любом сайте при авторизации пользователя используется SQL запрос, который выбирает запись из таблицы с пользователями, которая соответствует введенным пользователем данным (как правило, это Username и Password). Именно в этом случае злоумышленники и пытаются внедрить инъекцию, рассчитывая на плохую защиту сайта. Так, введя в соответствующие поля следующие данные: username: user'+OR+1=1- - password: password гарантирована SQL-инъекция. В данном случае даже при неправильном пароле злоумышленник сможет авторизоваться. Это стало возможным благодаря кавычке в данных. Не важно, что было указанно в качестве username, добавив конструкцию OR 1=1 мы получим список всех пользователей. Символы «- -» означают комментарии. Т.е. после этих символов код не учитывается в качестве запроса.
2. Использование поиска по целочисленным данным.
   Еще одним вариантом внедрения SQL инъекции является использование адреса вида «http://site.ru/users/?id=12», который обычно используется для просмотра информации о любом пользователе. В результате такого запроса вызывается скрипт, который выполняет поиск пользователя, чей id равен 12. Но лишь немного исправив запрос на «http://site.ru/users/?id=12;DELETE FROM users- -» мы лишимся всей информации о пользователях. Такая ситуация возможно из-за того, что мы преобразуем стандартный запрос в два, поставив после id символ «;», который в итоге и разграничит запрос на два: первый делает выборку пользователя, второй удаляет всех пользователей из таблицы.
3. Указание неверных идентификаторов.
   Часто сайт предоставляет нам возможность отсортировать выходные данные. Например, выполняя запрос по адресу «http://site.ru/users/?field=id&order=; DELETE users». В этом случае, злоумышленник может выполнить SQL-инъекцию, которая удаляет информацию о пользователях. А все из-за того, что вместо предполагаемого направления сортировки (ASC, DESC) злоумышленник вводит инъекцию.

Всех этих sql-инъекций можно избежать, тщательно проверяя и обрабатывая данные.

II. Методы защиты от SQL-инъекций

1. Приведение типов и ограничение формата. Благодаря этом методу все строковые данные, которые введены в поле с числовыми данными, будут отсечены, и внедрение кода инъекции для числовых значений будет практически исключено.
2. Экранирование специальных символов. В данном случае введенные пользователем кавычки будут экранированы символом «\» и не будут учитываться в запросе в качестве специального символа, что позволит избежать нарушения целостности запроса.
3. Плейсхолдеры — подстановка данных. Данные попадают в запрос не напрямую, а через своего рода представителя, подстановочное выражение. Существуют два варианта реализации плейсхолдеров — серверный и клиентский. В первом случае запрос уходит на сервер с плейсхолдерами, а данные отправляются отдельно от него. Во втором случае данные форматируются и подставляются в строку запроса на ме-

сто плейсхолдеров прямо на клиенте, формируя классический SQL запрос, который затем уходит в базу обычным порядком.

4. Использование белых списков идентификаторов и ключевых слов. Суть метода заключается в том, что все возможные варианты выбора должны быть жёстко прописаны в коде, и в запрос могут попадать только они, на основании соответствия пользовательского ввода и списка разрешенных значений.

III. Анализ методов

Экранирование (эскейпинг) и приведение типов могут обезопасить от небольшого количества используемых инъекций, поэтому эти методы усложняют задачу взлома, но не исключат полностью его возможность. Кроме того, использование длинных функций по всему коду не очень удобно и может запросто вызвать путаницу у разработчиков, если будет необходимо выяснить, обрабатывались ли данные ранее либо они еще не обработаны предлагаемыми функциями.

Работа с плейсхолдерами дает достаточно надежную защиту, но имеет такие недостатки как многословность, недостаточность функционала, невозможность получить классический SQL запрос для целей отладки, проблемы с производительностью

IV. Тестирование предложенного решения

Целью данной работы было написание функционала по работе с собственными плейсхолдерами, основанными на плейсхолдерах mysqli, что позволит упростить разработку и значительно улучшить защиту от SQL-инъекций веб-приложения. Основные меры по защите, который предоставляет данный функционал, заключается в обработке всех специальных символов в пользовательских данных (экранирование), числа рассматриваются как символы и обрабатываться аналогичным образом. Везде, где необходимо, используются белые списки SQL идентификаторов. Еще одним важным достоинством разработанных плейсхолдеров является их разделение по типам. Для каждого типа, будь это строка, число, массив или идентификатор применяется свой тип обработки данных. Проведено тестирование, во время которого предпринимались попытки получения защищенной информации.

1. Например, рассмотрим запрос получения информации о пользователе через username. При попытке внедрения SQL инъекции в адресе «/index.php?username=Alex';DROP user- -» система защиты успешно обработает пришедшую строку, экранировав символ кавычки, благодаря чему целостность запроса не нарушится и пользователь просто не будет найден.
2. Попытаемся применить SQL инъекцию к числовому формату. Так, попытавшись получить данные о пользователе по id с SQL-инъекцией: «/index.php?id=12;DROP user--». В данном случае введенный строка преобразуется к челочислнному типу и код инъекции не дойдет до базы.
3. Также стоит проверить работу «белых списков». Предположим, клиентское приложение хочет получить информацию о пользователе, причем наименование конкретных полей можно указать в запросе: «/users.json?fielsdet=id,password,username&id=1». Можно заметить, что среди полей есть поле «password», которое мы не можем показать. Используя «белые списки», мы показываем, какие поля можно выбирать. Если же указанно поле, которое не входит в этот список, то оно не будет выведено. Таким образом, инъекции и этого вида будет отражена.
4. SQL-инъекции, основанные на вставке вредоносного кода вместо идентификаторов сортировки (ASC, DESC) также будут отражены путем их проверки в разрешенном списке. В случае, если указанное поле не найдено в списке, то мы возвращаем FALSE, и взломщик получит вместо данных сообщение об ошибке.

В результате проведенной работы установлено, что использование разработанных плейсхолдеров для библиотеки mysqli позволило упростить работу с БД, сделать ее более удобной и безопасной. При этом возможность внедрения SQL инъекций практически сведена к нулю.

1. Википедия [Электронный ресурс] / SQL injection. – Режим доступа: http://en.wikipedia.org/wiki/SQL_injection. – Дата доступа: 05.09.2015.
2. Официальный сайт PHP [Электронный ресурс] / Оф. сайт PHP. – Режим доступа: http://www.php.net/manual/ru/security.database.sql-injection.php. – Дата доступа: 20.09.2015/
3. Тараканов, А. Н. Модели данных и СУБД / А. Н. Тараканов // БГУИР. – 2008. – 56 с.
4. Гарсиа-Молина, Г. Системы баз данных / Г. Гарсиа-Молина, Дж. Ульман // Издательский дом «Вильямс», 2003. – 1088 с.