А. М. Токарчук

ПРИМЕНЕНИЕ СРЕДСТВ ORM ДЛЯ РАЗРАБОТКИ БЕЗОПАСНЫХ ВЕБ-ПРИЛОЖЕНИЙ

Сейчас мы живем в глобальном информационном обществе, система ценностей людей меняется, и особую значимость приобретает информация. В настоящее время ее значение трудно переоценить. Можно сказать, что важнее информации нет ничего.

Компании затрачивают огромные средства на обеспечение информационной безопасности: производится установка антивирусных систем, систем предотвращения атак, различных систем фильтрации трафика, настройка списков контроля доступа к информационной системе и многое другое. Но проблема незащищенности публично-доступных (а также локально-доступных) систем состоит в другом.

Ошибки — это неизбежный атрибут программного кода.

Проанализировав около 180 известных программных продуктов с открытым исходным кодом, Департамент внутренней безопасности США (US Department of Homeland Security) пришел к выводу, что на каждую тысячу строк кода в этих продуктах приходится в среднем одна ошибка или уязвимость [1].

Такой вывод был сделан в результате спонсируемого Департаментом внутренней безопасности исследования, получившего название Open Source Hardening Project. А ведь в проекте не были затронуты коммерческие программные продукты с закрытым исходным кодом (по понятным причинам), в которых процент ошибок еще больше, так как у коммерческой компании может не быть того огромного штата сотрудников, который будет заниматься поиском уязвимостей в программном обеспечении. Компания может полагаться на автотесты, сканеры уязвимостей и другие методики, но, тем не менее, ошибки в программном коде все равно неизбежны.

В связи с этим встает вопрос о сокращении количества ошибок в программном продукте. Для этого необходимо разобраться, в чем сущность работы программиста веб-приложений, а именно коснуться вопроса хранения данных и взаимодействия веб-приложения с ними. Как правило, всё взаимодействие с данными происходит посредством CRUD-операций (Create, Read, Update, Delete) [2]. Create — создание экземпляра сущности (модели, объекта) в БД (базе данных), Read — чтение экземпляра из БД, Update — обновление экземпляра в БД, Delete — удаление экземпляра из БД. Для осуществления любой из этих операций при работе с SQL[1] СУБД необходимо выполнить ряд шагов:

1. Подготовка данных для запроса к БД. При этом из программных моделей отбираются данные, необходимые для запроса.
2. Подготовка текста SQL-запроса с учетом полученных в п. 1 данных.
3. Непосредственное выполнение запроса сервером БД.
4. Обработка исключений, которые могут возникнуть при выполнении запроса.
5. Получение результата выполненного запроса.
6. Запись части данных (или всех данных) из полученной в результате запроса выборки в программные модели.

Сразу необходимо отметить, что при недостаточной проверке входных данных в п. 1 могут быть реализованы такие типы атак, как SQL-injection[2] [3], т. е. вставка злонамеренного кода в

[1] SQL (Structured Query Language — язык структурированных запросов) — универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.

[2] Внедрение SQL-кода (англ. SQL injection) — один из распространенных способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода.

тело запроса. При этом после его выполнения может возникнуть ситуация, когда злоумышленник получит доступ к серверу с правами пользователя сервера баз данных. Программисту требуется постоянно выполнять эти рутинные операции, и наряду с возможностью атаки типа SQL-injection следует учитывать человеческий фактор. Программист может устать и пропустить часть проверки входных данных или уделить недостаточно внимания обработке результата SQL-запроса, из-за чего в системе может возникнуть брешь. Для решения проблемы автоматизации программирования рутинных операций предлагается использовать системы объектно-реляционного отражения (ORM) для подготовки запроса к БД, его выполнения, анализа результатов и формирования программных моделей.

ORM (англ. Object-relational mapping, Объектно-реляционная проекция) — запись объектов программы в реляционную базу данных, отображение объекта и его представления в виде набора таблиц. ORM — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как коммерческие, так и свободные реализации этой технологии. ORM позволяют сэкономить время разработчиков, а значит, и сократить бюджет разработки веб-сервиса, а также обезопасить приложение уже на этапе разработки.

ORM построены на основе шаблонов («паттернов») проектирования. Шаблоны проектирования (паттерны, англ. design pattern) — это многократно применяемая архитектурная конструкция, предоставляющая технологическую поддержку проектирования в рамках конкретного контекста и описывающая значимость конкретного решения [4]. Паттерн не является законченным образцом проекта, который может быть прямо преобразован в код (что отличает его от близкого понятия идиомы), скорее, это описание или образец для того, как решить задачу таким образом, чтобы это можно было использовать в различных ситуациях. Объектно-ориентированные шаблоны зачастую показывают отношения и взаимодействия между классами или объектами без определения того, какие конечные классы или объекты приложения будут использоваться [5].

Предлагаемая к рассмотрению ORM-система ActiveModel — это метод проектирования, положенный в основу ORM-системы и использующий абстракцию от базы данных методом объектно-реляционного отражения. При задании требований к разработке ORM-системы ActiveModel были учтены следующие возможности:

1. Поиск моделей в базе данных, с использованием атрибутов моделей или выражений на языке SQL;

2. Разные модели могут быть в разных таблицах разных баз данных;

3. Одна модель может состоять из нескольких таблиц, размещенных в разных базах данных и даже на разных серверах;

4. Возможно использование моделей, составленных из нескольких таблиц;

5. Поддержка таблиц вида: classic (обычная таблица), x-fields (таблица с произвольным количеством виртуальных колонок и фиксированным количеством реальных), x-flags (аналог x-fields для хранения флагов);

6. Возможность автоопределения виртуальных колонок для таблиц типа x-fields и x-flags;

7. Использование «Lazy-loading» для связей таблиц (загрузка связей по требованию);

8. Возможность использования программных моделей и как объектов, и как массивов (технология ArrayAccess) (Код $model -> varName и $model['varName'] возвратят один и тот же результат);

9. Использование коллекции объектов в случае, когда в результате выборки образуется множество моделей;

10. Возможность фильтрации моделей после выборки;

11. Инкремент/декремент свойств модели с учетом многопоточности;

12. Возможность использовать таблицы «только для чтения»;

13. Возможность использовать «Fluent Interface», т. е. цепочки вызовов наподобие нижеприведенной:

$user = User::model() -> findByAttributes(array('username' => 'andrey')) → populate($data) -> save()[3].

Также следует заметить, что применение ActiveModel делает процесс разработки более гибким. Эта гибкость характеризуется тем, что при изменении таблиц в БД нет необходимости переписывать все функции, работающие с ними, а нужно всего лишь переписать функцию __construct() модели. Таким образом, применение паттерна ActiveModel может значительно ускорить как процесс разработки, так и процесс сопровождения кода веб-приложения и обезопасить веб-приложение уже на этапе разработки.

СПИСОК ЛИТЕРАТУРЫ:

1. Миф о надежности open-source развеян: обнаружена 1 ошибка на каждую 1000 строк // Softodrom.ru. URL: http://news.softodrom.ru/ap/b2388.shtml.
2. *Веллинг Л., Томсон Л.* Разработка Web-приложений с помощью PHP и MySQL. Вильямс, 2009.3. *Дюбуа П.* MySQL. Сборник рецептов. Символ, 2007.
4. *Фаулер М.* Архитектура корпоративных программных приложений. Вильямс, 2009.
5. *Хоп Г., Вульф Б.* Шаблоны интеграции корпоративных приложений. Вильямс, 2009.

[3] Представлен код на языке веб-программирования PHP.

Ю. М. Туманов, С. В. Гаврилюк

ИСПОЛЬЗОВАНИЕ ПОВЕДЕНЧЕСКОГО АНАЛИЗА С ПРИМЕНЕНИЕМ ПОВЕДЕНЧЕСКИХ СИГНАТУР ДЛЯ ВЫЯВЛЕНИЯ ВРЕДОНОСНОГО КОДА НА ПРИМЕРЕ JAVASCRIPT-СЦЕНАРИЕВ

Интернет построен на протоколе передачи данных HTTP, который позволяет передавать HTML-текст страницы интернет-браузерам пользователей. Данная технология повсеместно используется в сети Интернет и постоянно развивается. Очередным этапом развития технологии HTTP в 1995 г. стало создание языка JavaScript.

На данный момент количество интернет-страниц, использующих технологию JavaScript, составляет 75 % [1]. В свою очередь, около 80 % интернет-страниц, использующих JavaScript, содержат в себе различные уязвимости [2].

В антивирусной индустрии широко используются два подхода к детектированию вредоносного программного обеспечения — детектирование вредоносного кода по его сигнатуре либо по его поведению.