УДК 004.052.42:004.056.53

МОДЕЛЬ ОЦЕНКИ КАЧЕСТВА WEB-ПРИЛОЖЕНИЙ, ОСНОВАННАЯ НА ОБНАРУЖЕНИИ УЯЗВИМОСТЕЙ К SQL-ИНЪЕКЦИЯМ

Д.Е. ОНОШКО, В.В. БАХТИЗИН

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 19 ноября 2015

Рассматривается способ обнаружения уязвимостей к SQL-инъекциям, основанный на статическом анализе исходных кодов. Приведена модель качества web-приложений, основывающаяся на результатах автоматизированного поиска уязвимостей к SQL-инъекциям. Представлены практические результаты применения модели качества к существующим web-приложениям.

Ключевые слова: качество, надежность, сопровождаемость, web-приложение, статический анализ, SQL-инъекция.

Введение

Переход от классических desktop-приложений к web-приложениям, наметившийся в настоящее время, сопровождается ростом требований к качеству таких приложений. По данным Открытого проекта обеспечения безопасности web-приложений (OWASP) к 2013 году для различных типов приложений (включая web-приложения) наиболее распространенной угрозой были SQL-инъекции [1].

Главной причиной уязвимости web-приложений к данной угрозе является некорректная обработка данных, поступающих в приложение извне. Попадая в web-приложение, так называемые «сырые» данные (raw data) должны пройти предварительную обработку (фильтрацию), характер которой определяется тем, как эти данные используются приложением. Недостаточная или некорректная обработка данных, предназначенных для подстановки в запросы к системе управления базами данных (СУБД), позволяет злоумышленнику подобрать такие значения, при которых эти данные будут ошибочно распознаваться как управляющие. Это в свою очередь позволит выполнить действия, не предусмотренные разработчиками web-приложения.

Проверка web-приложения на наличие подобных уязвимостей является трудоемкой рутинной задачей и, следовательно, должна быть автоматизирована. Поскольку уязвимость к SQL-инъекции представляет собой ошибку в обработке данных приложением, возможны два основных подхода к решению этой задачи: статический анализ, заключающийся в выявлении уязвимостей по исходным кодам, и динамический анализ, основанный на изучении поведения web-приложения для различных исходных данных.

Таким образом, для обеспечения контроля качества web-приложения на различных этапах процессов разработки и сопровождения необходимо решить две задачи:

– автоматизировать обнаружение уязвимостей к SQL-инъекциям со сбором информации об их характере (разработать модель обнаружения уязвимостей);

– обозначить способ использования собранной информации в оценке качества (разработать модель качества).

Обнаружение уязвимостей в web-приложениях

С целью снижения сложности модели обнаружения уязвимостей необходимо принять во внимание, что эксплуатация уязвимости к SQL-инъекции происходит «на границе» web-приложения с СУБД – в момент подготовки запроса. В общем случае вместо СУБД может использоваться любое другое техническое средство, позволяющее осуществлять хранение данных и доступ к ним, поэтому целесообразно ввести для обозначения подобных технических средств понятие хранилища данных.

С точки зрения анализа web-приложения на наличие уязвимостей к SQL-инъекциям все действия, выполняемые его кодом, могут так или иначе быть сведены к выполнению следующих преобразований:

– преобразование данных HTTP-запроса в запросы к хранилищу данных;

– преобразование данных HTTP-запроса и данных, полученных из хранилища данных, в HTTP-ответ (HTML-страницу, JSON- или XML-данные и т.д.).

Исходя из этого, web-приложение можно рассматривать как конечный автомат Мили. С точки зрения клиента web-приложение является автоматом, включающим в себя код web-приложения и хранилище данных. Входным воздействием для данного автомата является HTTP-запрос, поступающий от клиента. Выходной сигнал – ответ на этот запрос. Ответ зависит от параметров, переданных с запросом, и данных, полученных из хранилища данных. Хранилище данных выступает в роли состояния автомата. Код web-приложения в таком представлении не содержит запоминающих элементов, т.е. состояния, являясь, по сути, сложной функцией, обеспечивающей преобразование данных без их длительного хранения.

Как работу с хранилищем данных, так и вывод ответа на HTTP-запрос можно свести к вызовам стандартных процедур, предоставляемых языком программирования или стандартной библиотекой. В соответствии с принципами восходящего программирования можно показать, что web-приложение представляет собой множество процедур, каждая из которых основана на более простых процедурах. При этом главный блок web-приложения (также называется точкой входа) является процедурой, параметры и возвращаемое значение которой совпадают с входными и выходными данными web-приложения, а логика реализована с использованием более простых процедур, сводящихся, в конечном счете, к набору стандартных процедур.

Для удобства дальнейшего анализа необходимо изменить подход к рассмотрению параметров процедур, определив в рамках модели следующие их виды:

– in-параметры – для описания данных, передаваемых в процедуру;

– out-параметры – для описания данных, возвращаемых из процедуры.

Возвращаемое значение функции при этом является особым случаем out-параметра. Кроме того, следует ввести два вида оценок – оценки для данных (переменных, констант) и оценки для параметров процедур. В простейшем варианте модели целесообразно использовать бинарные оценки. В дальнейшем, при реализации основанных на предлагаемой модели программных средств возможно расширение системы оценок.

Для переменных предлагается использовать следующую систему оценок:

– оценку S (safe) получают переменные, которые при подстановке в SQL-запрос не нарушат предусмотренной разработчиками web-приложения логики, т.е. содержащие надлежащим образом обработанные (экранированные) данные;

– оценку U (unsafe) получают переменные, которые могут содержать последовательности символов, имеющие в SQL специальное назначение, т.е. содержащие данные, не прошедшие должной обработки.

Система оценок для параметров процедур основывается на тех же соображениях. Для in-параметров предлагается использовать следующую систему оценок:

– оценку S (safe) получают in-параметры, для которых передаваемое значение обязательно должно быть надлежащим образом обработано (экранированы спецсимволы и т.п.);

– оценку U (unsafe) получают in-параметры, для которых передача необработанных данных не приводит к возникновению уязвимости.

Предлагаемая система оценок для out-параметров имеет следующий вид:

– оценку S (safe) получают out-параметры, через которые возвращаются надлежащим образом обработанные данные (экранированы спецсимволы и т.п.);

– оценку U (unsafe) получают out-параметры, через которые осуществляется возврат значений, которые могут содержать неэкранированные спецсимволы и управляющие последовательности.

Анализ соответствия оценок формальных и фактических параметров позволяет формально доказать корректность или некорректность обработки данных web-приложением. В рамках модели web-приложение рассматривается как множество (в том числе стандартных) процедур $F = \{P_1, P_2, \ldots P_N\}$, где N – общее количество процедур в web-приложении, а процедура P_N – главная процедура программы. Пусть на i-м шаге анализа известны оценки для параметров процедур $F_i = \{P_1, P_2, \ldots P_{C(i)}\}$, $F_i \subseteq F$, где $C(i) \le N$ – количество таких процедур (зависит от количества стандартных процедур с заранее известными оценками). При этом, поскольку все процедуры принадлежат одному и тому же приложению, существует процедура $P_{C(i)+1}$, зависящая только от процедур из множества F_i. В этом случае, анализируя операторы процедуры $P_{C(i)+1}$, можно получить оценки для ее параметров.

Поскольку в результате такого анализа некоторые процедуры, выполняющие необходимые преобразования (фильтрацию) данных, могут оказаться нераспознанными (т.е. некоторые параметры по ошибке могут получить оценку U вместо S), существует вероятность ложноположительных результатов. Для снижения их частоты при практической реализации модели можно предложить введение оценки UDS (user-defined safe), позволяющей пользователю явно указать, что какой-либо параметр процедуры должен рассматриваться как параметр с оценкой S.

Модель качества web-приложений

Анализ предметной области показывает, что в настоящее время при обсуждении вопросов качества программных средств (и, в частности, web-приложений) целесообразно опираться на международный стандарт ISO/IEC 25010:2011 [2], как отражающий последние тенденции в сфере информационных технологий и предлагающий наиболее полный набор характеристик качества по сравнению с другими, в т.ч. действующими на территории Республики Беларусь, стандартами.

Модель качества продукта, приводимая в стандарте ISO/IEC 25010:2011, является трехуровневой: оценка качества в этой модели состоит из характеристик, подхарактеристик и мер. Стандарт регламентирует состав характеристик и подхарактеристик, расширение модели допускается только путем добавления мер к уже существующим подхарактеристикам.

Прежде чем задать набор мер, которые могут использоваться для оценки качества web-приложений по результатам обнаружения уязвимостей к SQL-инъекциям, необходимо ввести несколько понятий, которые позволят более кратко и точно описывать способ вычисления мер и их физический смысл.

Точка входа данных – семантически неделимая единица данных, поступающих в приложение извне. Как правило, одной точке входа данных соответствует одно поле в окне приложения, web-форме и т.д. Точка входы данных в вышеприведенной модели обнаружения уязвимостей является in-параметром главного блока P_N.

Путь SQL-инъекции – последовательность вложенных вызовов процедур приложения, позволяющая передать необработанные данные в запрос к СУБД.

Значения, характеризующие качество web-приложения с точки зрения наличия в нем уязвимостей к SQL-инъекциям, могут быть измерены в нескольких частях web-приложения. Другими словами, наибольший интерес при разработке мер модели качества представляют:

– оценки для точек входа данных web-приложения – характеризуют наличие или отсутствие уязвимостей к SQL-инъекциям;

– оценки in-параметров процедур, предоставляющих программный интерфейс для доступа к хранилищу данных, и значений их фактических параметров – характеризуют наличие или отсутствие уязвимостей к SQL-инъекциям;

– оценки формальных и фактических параметров процедур всего приложения в целом – характеризуют вероятность возникновения уязвимостей при изменении исходных кодов в ходе доработки web-приложения;

– наличие и количество in-параметров с оценкой UDS – характеризуют возможность достоверной оценки качества web-приложения в рамках модели обнаружения уязвимостей и (косвенно) вероятность возникновения уязвимостей при изменении исходных кодов.

Предлагаемые для использования в рамках модели качества web-приложения меры, основанные на этих оценках, представлены в табл. 1.

Таблица 1. **Меры качества web-приложений, отражающие их устойчивость к атакам**

Мера	Формула
1. Устойчивость к воздействиям	$X = 1 - \frac{A}{B}$. A – количество точек входа данных, потенциально допускающих проведение атак. B – общее количество точек входа данных.
2. Устойчивость к изменениям	$X = \frac{A}{B}$. A – количество формальных in-параметров с оценкой U. B – общее количество формальных in-параметров. Стандартные процедуры не учитываются.
3. Корректность обработки данных	$X = \frac{A}{B}$. A – количество in-параметров, при передаче которых соблюдаются правила применения оценок. B – общее количество in-параметров, передаваемых при вызовах процедур.
4. Избыточность обработки данных	$X = \frac{A}{B}$. A – количество in-параметров, при передаче которых оценка фактического параметра выше (лучше) оценки формального параметра. B – общее количество in-параметров, передаваемых при вызовах процедур. Стандартные процедуры не учитываются.
5. Анализируемость обработки данных	$X = 1 - \frac{A}{B}$. A – количество in-параметров с оценкой UDS. B – общее количество in-параметров с оценкой S или UDS.

Для всех предложенных мер $A \le B,\ 0 \le X \le 1$. При этом большему значению X соответствует более высокий уровень качества.

Мера «Устойчивость к воздействиям» характеризует устойчивость web-приложения к атакам, рассматривая его, как «черный ящик», и фактически является наиболее ценной из числа предложенных для заказчика разработки web-приложения.

Мера «Устойчивость к изменениям» характеризует способность web-приложения оставаться неуязвимым к атакам при внесении изменений в его исходные коды. В ходе доработки web-приложения могут быть непреднамеренно внесены ошибки, нарушающие соответствие формальных и фактических параметров. Данная мера позволяет косвенно оценить вероятность того, что при этом web-приложение останется неуязвимым к атакам, и в первую очередь полезна для руководителей проектов при оценке рисков.

Мера «Корректность обработки данных» отражает наличие или отсутствие ошибок, допущенных разработчиками web-приложения при обработке данных. При этом фиксируются не только ошибки, непосредственно делающие web-приложение уязвимым, но и ошибки, которые могут стать причиной уязвимости в результате последующих изменений в коде web-приложения. Снижение значения данной меры может говорить о необходимости проведения дополнительного обучения разработчиков или принятия других административных мер. В отличие от меры «Устойчивость к воздействиям», рассматривающей web-приложение как «черный ящик», данная мера оценивает качество кода, составляющего web-приложение.

Мера «Избыточность обработки данных» характеризует запас устойчивости web-приложения к атакам. Близкие к 1 значения данной меры, с одной стороны, говорят о большом

запасе устойчивости, с другой – могут указывать на то, что часть кода web-приложения выполняет большую фильтрацию данных, чем необходимо, а значит, при необходимости, возможно сократить объем кода, повысив тем самым производительность web-приложения.

Мера «Анализируемость обработки данных» характеризует web-приложение с точки зрения возможности его автоматического анализа. Малые значения данной меры указывают на высокую сложность алгоритмов фильтрации данных и могут быть показанием к проведению рефакторинга. Разработанная модель качества включает предложенные меры, как показано на рисунке.

Модель качества web-приложения

Для получения интегральной оценки качества web-приложения необходимо осуществить выбор весовых коэффициентов в зависимости от значимости каждой из мер в рамках конкретного подлежащего оценке web-приложения. После этого значения подхарактеристик, характеристик и интегральной оценки качества могут быть получены по аналогии с формулами, предложенными в [4]:

$$S_{ij} = \sum_{k=1}^{m} \left(M_{jk} \cdot V_{jk}^{M} \right),$$

$$C_i = \sum_{j=1}^{n} \left(S_{ij} \cdot V_{ij}^{S} \right),$$

$$Q = \sum_{i=1}^{q} \left(C_i \cdot V_i^{C} \right),$$

где M_{jk} – значение k-й меры j-й подхарактеристики, V_{jk}^{M} – значение весового коэффициента меры M_{jk}, m – общее количество мер, используемых при вычислении значения подхарактеристики; S_{ij} – значение j-й подхарактеристики i-й характеристики качества, V_{ij}^{S} – значение весового коэффициента подхарактеристики S_{ij}, n – общее количество подхарактеристик, используемых при вычислении значения характеристики; C_i – значение i-й характеристики качества, V_i^{C} – значение весового коэффициента характеристики C_i, q – общее количество характеристик, используемых при вычислении интегральной оценки качества web-приложения; Q – интегральная оценка качества web-приложения с точки зрения уязвимости к SQL-инъекциям. При этом рекомендуется выбирать весовые коэффициенты V_{jk}^{M}, V_{ij}^{S} и V_i^{C} так, чтобы выполнялись условия $\sum_{k=1}^{m} V_{jk}^{M} = 1$, $\sum_{j=1}^{n} V_{ij}^{S} = 1$, $\sum_{i=1}^{q} V_i^{C} = 1$. Предлагаемая модель может дополняться другими мерами в соответствии с требованиями стандарта ISO/IEC 25010:2011 для получения оценки качества, учитывающей также свойства web-приложения, не относящиеся к вопросам его уязвимости к SQL-инъекциям. В этом случае рекомендуется придерживаться тех же принципов выбора мер и их весов.

Для экспериментальной проверки модели качества было разработано программное средство (ПС) анализа исходных кодов, поддерживающее подмножество языка PHP. Исходными данными для него выступали 10 тестов (web-приложений; пронумерованы от 1 до 10), разработанных вручную «с нуля» (№№ 4, 5, 7), а также на основе исходных кодов нескольких крупных проектов: LiveStreet (№ 1), WordPress (№ 2), Invision Power Board (№ 3), itERP (№№ 6, 9) и KidFeed (№№ 8, 10). При разработке тестов, основанных на существующих web-приложениях, проводилась адаптация их исходных кодов к возможностям ПС анализа исходных кодов. Исходный код web-приложений подвергался удалению второстепенной функциональности и приводился к поддерживаемому подмножеству языка PHP. В полученное web-приложение также искусственно вносились уязвимости к SQL-инъекциям.

Таблица 2. **Весовые коэффициенты, использованные для интегральной оценки качества**

Меры	V_{jk}^{M}	V_{ij}^{S}	V_{i}^{C}
1. Устойчивость к воздействиям	1	1	0,5
2. Устойчивость к изменениям	0,33	0,5	0,5
3. Корректность обработки данных	0,33		
4. Избыточность обработки данных	0,33		
5. Анализируемость обработки данных	1	0,5	

Коэффициенты, выбранные для получения интегральной оценки качества, приведены в табл. 2, результаты экспериментальной проверки – в табл. 3. Тестовые приложения оценивались с помощью ПС-анализа исходных кодов, а также группой экспертов, не участвовавших в подготовке тестовых кодов. Экспертные оценки приведены в виде среднего арифметического всех оценок, выставленных экспертами.

Для проверки предложенных мер на соответствие критериям обоснованности [5] был выбран критерий непротиворечивости: для любых u-го и v-го web-приложений и любой меры M при выполнении условия $Q^{(u)} > Q^{(v)}$ (где $Q^{(u)}$ и $Q^{(v)}$ – интегральные оценки качества u-го и v-го web-приложений) должно выполняться условие $M^{(u)} > M^{(v)}$ (где $M^{(u)}$ и $M^{(v)}$ – значения меры M для u-го и v-го web-приложений). Для результатов в табл. 3 это условие выполняется, т.е. меры предложенной модели качества являются обоснованными.

Таблица 3. **Результаты экспериментальной проверки предложенных мер**

Меры	Тип оценки	Номер теста									
		1	2	3	4	5	6	7	8	9	10
1. Устойчивость к воздействиям	эксперты	0,82	0,75	0,37	0,09	1,00	0,43	0,29	0,57	0,32	0,17
	ПС	0,79	0,73	0,34	0,08	1,00	0,41	0,26	0,52	0,29	0,15
2. Устойчивость к изменениям	эксперты	0,92	0,84	0,41	0,10	0,97	0,48	0,32	0,64	0,36	0,19
	ПС	0,94	0,87	0,43	0,14	0,98	0,51	0,35	0,69	0,39	0,22
3. Корректность обработки данных	эксперты	0,88	0,77	0,36	0,09	1,00	0,40	0,22	0,64	0,28	0,10
	ПС	0,87	0,80	0,38	0,11	1,00	0,45	0,29	0,61	0,32	0,17
4. Избыточность обработки данных	эксперты	0,69	0,57	0,19	0,05	0,81	0,25	0,16	0,34	0,16	0,11
	ПС	0,82	0,67	0,20	0,04	0,93	0,31	0,19	0,43	0,24	0,15
5. Анализируемость обработки данных	эксперты	0,95	0,93	0,86	0,23	1,00	0,89	0,65	0,71	0,68	0,32
	ПС	0,72	0,66	0,61	0,12	1,00	0,63	0,52	0,81	0,42	0,39
Интегральная оценка качества	эксперты	0,86	0,79	0,48	0,12	0,98	0,53	0,37	0,60	0,40	0,20
	ПС	0,79	0,73	0,41	0,09	0,99	0,47	0,33	0,61	0,33	0,22

Заключение

Предложенная в статье модель качества web-приложений, основанная на результатах оценки их исходных кодов, позволяет осуществлять оценку и отслеживать изменение качества web-приложений с точки зрения их уязвимости к SQL-инъекциям на различных этапах процессов разработки и сопровождения. Оценки качества, получаемые с использованием модели, подтверждают ее соответствие критериям обоснованности и могут использоваться для принятия управленческих решений, направленных на повышение качества web-приложений.

A WEB-APPLICATION QUALITY ASSESSMENT MODEL BASED ON SQL-INJECTION VULNERABILITY DETECTION

D.E. ONOSHKO, V.V. BAKHTIZIN

Abstract

A way of SQL-injection vulnerability detection based on static code analysis is discussed. A web-application internal quality model based on the results of automated detection of SQL-injection vulnerabilities is given. Practical results of applying the quality model to real-world web-applications are presented.

Keywords: quality, reliability, maintainability, web-application, static analysis, SQL-injection.

Список литературы

1. OWASP Top 10-2013. The Ten Most Critical Web Application Security Risks. [Электронный ресурс]. – Режим доступа: http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf. – Дата доступа: 31.10.2013.
2. ISO/IEC 25010:2011. Системная и программная инженерия. Требования к качеству и оценка программного продукта (SQuaRE). Модели качества систем и программных средств.
3. *Бахтизин В.В. Глухова Л.А., Неборский С.Н.* Метрология, стандартизация и сертификация в информационных технологиях. Минск, 2013.
4. ГОСТ 28195-99. Оценка качества программных средств. Общие положения.
5. *Бахтизин В.В., Глухова Л.А.* Стандартизация и сертификация программного обеспечения. Минск, 2006.