УДК 004.052.42:004.056.53

Метод оценки качества web-приложений, основанный на обнаружении уязвимостей

Д. Е. Оношко, магистр технических наук, ассистент кафедры ПОИТ
E-mail: onoshko@bsuir.by

Белорусский государственный университет информатики и радиоэлектроники, ул. П. Бровки, д. 6, 220013, г. Минск, Республика Беларусь

В. В. Бахтизин, к. т. н., профессор кафедры ПОИТ

Белорусский государственный университет информатики и радиоэлектроники, ул. П. Бровки, д. 6, 220013, г. Минск, Республика Беларусь

Аннотация. В статье предлагается метод оценки качества web-приложений, основанный на обнаружении уязвимостей путем статического анализа исходных кодов. Приводится описание модели обнаружения уязвимостей к SQL-инъекциям, а также модели качества web-приложений, основанной на результатах обнаружения уязвимостей и являющейся расширением модели качества ISO/IEC 25010.
Ключевые слова: web-приложение; статический анализ; SQL-инъекция; уязвимость
Для цитирования: Оношко, Д. Е. Метод оценки качества web-приложений, основанный на обнаружении уязвимостей / Д. Е. Оношко, В. В. Бахтизин // Цифровая трансформация. – 2018. – № 1 (2). – С. 58–65.

A Web-application Quality Evaluation Method Based on Vulnerability Detection

D. E. Onoshko, Master of Technical sciences, Assistant Lecturer of Software for Information Technologies Department
E-mail: onoshko@bsuir.by

Belarusian State University of Informatics and Radioelectronics, 6 P. Brovki Str., 220013 Minsk, Republic of Belarus

V. V. Bakhtizin, Candidate of Sciences (Technology), Professor of Software for Information Technologies Department

Belarusian State University of Informatics and Radioelectronics, 6 P. Brovki Str., 220013 Minsk, Republic of Belarus

Abstract. This article introduces a method of web-application quality evaluation based on static analysis of the source code. A model for SQL-injection vulnerability detection and a web-application quality model based on the results of vulnerability detection that extends the ISO/IEC 25010 quality model are described.
Keywords: web-application; static analysis; SQL-injection; vulnerability
For citation: Bakhtizin V. V., Onoshko D. E. A Web-application Quality Evaluation Method Based on Vulnerability Detection. Cifrovaja transformacija [Digital transformation], 2018, 1 (2), pp. 58–65 (in Russian).

Введение. Наблюдающаяся в настоящее время популярность web-приложений, обусловленная, среди прочего, такими их свойствами, как доступность из любой точки мира и относительная независимость от программно-аппаратной платформы, используемой конечными пользователями, существенно повышает значимость проблемы обеспечения их качества.

Одним из аспектов качества современных web-приложений является их устойчивость к атакам методом внедрения кода (инъекциям). В соответствии с данными отчета Открытого проекта обеспечения безопасности web-приложений (OWASP)

в настоящее время наиболее распространенной угрозой для различных типов приложений являются SQL-инъекции [1]. Уязвимость web-приложений к данной угрозе возникает при некорректной обработке данных, поступающих извне. В зависимости от характера использования этих данных web-приложением «сырые» данные (raw data) могут требовать различного набора преобразований, которые должны выполняться на различных этапах обработки raw data. Неправильная фильтрация поступающих от пользователя данных позволяет злоумышленнику встроить в них фрагменты, которые при обработке web-приложением ошибочно распознаются как управляющие, что делает возможным несанкционированное изменение логики работы web-приложения.

Существующие технические решения, например, подготовленные выражения (prepared statements), при правильном использовании позволяют избежать подобных ошибок. Однако контроль правильности их использования — трудоемкая рутинная задача, требующая не только повышенной концентрации внимания, но и хорошего знания архитектуры разрабатываемого web-приложения, характера взаимосвязей отдельных его модулей и др. Важно также понимать, что любое изменение в коде самого web-приложения или программного обеспечения, входящего в состав платформы (например, обновление web-сервера), потенциально способно внести новые уязвимости, а также заставить проявиться уже имеющиеся, но не замеченные ранее. Следовательно, контроль web-приложения на наличие в нем уязвимостей необходим на протяжении всего жизненного цикла с момента появления первого прототипа.

Для снижения трудозатрат на осуществление такого контроля целесообразна разработка программного средства (ПС) обнаружения уязвимостей к SQL-инъекциям. Помимо собственно ответа на вопрос о их наличии такое ПС, в отличие от экспертного контроля, позволяет собрать сопутствующую информацию, которая может упростить принятие управленческих решений, связанных с обеспечением качества web-приложения.

Модель обнаружения уязвимостей. Как показано в одной из наших предыдущих статей [2], с точки зрения уязвимости к SQL-инъекциям web-приложение следует рассматривать как набор слоев, обеспечивающих преобразование запросов пользователей в запросы к системе управления базами данных (СУБД). При этом действия, выполняемые кодом web-приложения, сводятся к двум видам преобразований:

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

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

Модель web-приложения в контексте обнаружения уязвимостей к SQL-инъекциям (см. рис. 1) рассматривает все web-приложение в целом как конечный автомат Мили, входным сигналом которого является HTTP-запрос, поступающий от клиента, а выходным — ответ на этот запрос. Ответ зависит как от параметров, переданных с запросом, так и от данных, полученных из их хранилища. Таким образом, в web-приложении имеется внутренняя память. Это делает затруднительным его функциональное тестирование: одни и те же исходные данные в разное время могут приводить к получению различных результатов, бесконечно увеличивая множество тестов. Тем не менее, основываясь на предложенной модели web-приложения в контексте обнаружения уязвимостей, этот автомат может быть разделен на две части: хранилище данных и собственно код web-приложения. В этом случае код (т. е. web-приложение без частей, отвечающих за хранение данных между запросами) может рассматриваться как сложное преобразование HTTP-запроса в запросы к СУБД.

Преимущество такого разделения заключается в том, что вся необходимая для обнаружения уязвимостей информация содержится в коде web-приложения, а значит становится возможным его статический анализ, т. е. анализ без запуска web-приложения.

В рамках предлагаемой модели обнаружения уязвимостей для формальной верификации кода web-приложения используется абстрактная интерпретация [3]. При этом web-приложение рассматривается как множество процедур, связь между которыми устанавливается путем их вызова друг другом. Такой подход к рассмотрению кода web-приложения оказывается оправданным, поскольку работа с хранилищем данных, так или иначе, сводится к вызову стандартных процедур языка программирования или процедур стандартной библиотеки. Операции языка программирования также рассматриваются как процедуры.

В соответствии с принципами восходящего проектирования точка входа (или главный блок) web-приложения также является процедурой, параметры и возвращаемое значение которой совпадают с входными и выходными данными web-приложения, а логика реализована с использованием иерархии вызовов более простых

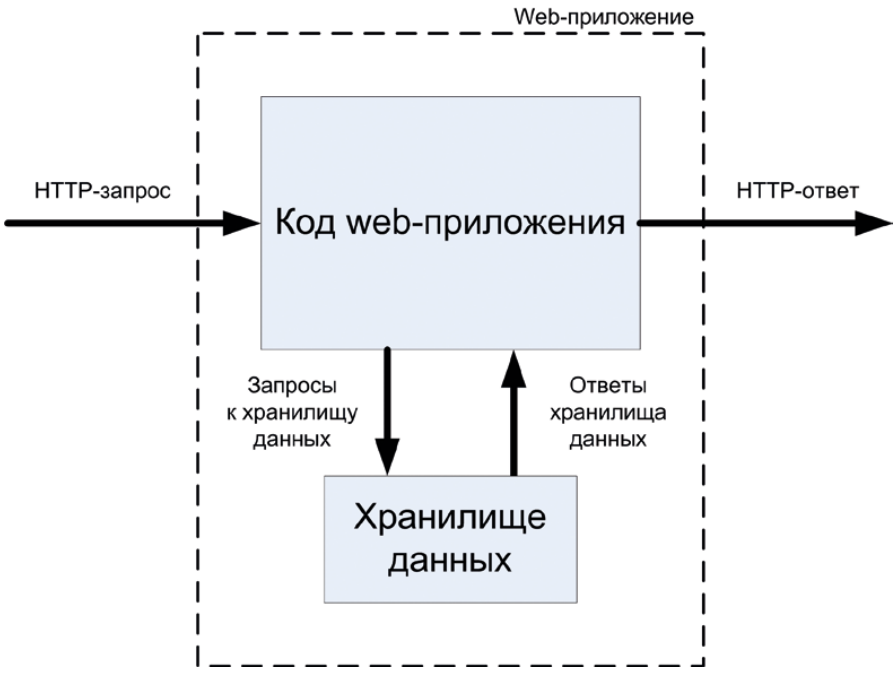

Рис. 1. Представление web-приложения как автомата Мили

процедур, на нижнем уровне которой находятся стандартные процедуры.

Во многих языках программирования помимо параметров, передаваемых по значению, существует возможность передавать параметры по указателю (ссылке), что позволяет вызываемой процедуре изменять значение переменной, использованной в качестве фактического параметра. Также в С-подобных языках программирования существуют операции с побочными эффектами: например, операции присваивания (=, +=, –= и т. д.), инкремента и декремента. Для того чтобы обеспечить применимость модели ко всем перечисленным случаям, предлагается выделить два вида параметров процедур:

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

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

Возвращаемое значение функции рассматривается как особый случай out-параметра.

В рамках предлагаемой модели in- и out-параметры используются для обозначения передачи данных только в одном направлении (либо в процедуру, либо из нее). Параметры, позволяющие выполнять передачу в обоих направлениях, при анализе заменяются сочетанием in- и out-параметров.

В статье Джоэла Спольски [4] рассматривается подход, заключающийся в использовании венгерской нотации при написании кода web-приложений для обозначения того, является ли содержимое той или иной переменной потенциально опасным при подстановке в SQL-запрос. Применение данного подхода позволяет разрабатывать web-приложения, не содержащие уязвимостей к SQL-инъекциям, однако при этом требуется достаточная квалификация разработчика для правильного выбора префиксов. Кроме того, изменения в коде web-приложения могут потребовать смены префикса, отслеживание чего в больших проектах является проблематичным. Наконец, данный подход является лишь рекомендацией и не позволяет дать ответ на вопрос о наличии или отсутствии уязвимостей в web-приложении. Для получения такого ответа необходимо выполнить переименование идентификаторов с правильным выбором префиксов, что может оказаться трудоемкой задачей.

Предлагаемая в статье модель обнаружения уязвимостей в отличие от указанного подхода [4] основана на автоматизированном назначении оценок отдельным элементам web-приложений. В рамках этой модели выделяются оценки двух видов — оценки для данных и оценки для параметров процедур. Такое разделение, во-первых, позволяет исключить повторный анализ кода процедур за счет сохранения промежуточных результатов, а во-вторых, учитывает существование процедур (операции языка и библиотечные процедуры), вместо исходного кода которых доступны только соответствующие оценки.

Система оценок в простейшем случае является бинарной и строится так, чтобы ответ на вопрос о правильности обработки данных web-приложением можно было получить сравнением оценок in-пара-

метров (формальных) с оценками фактически передаваемых значений. В дальнейшем при реализации программных средств, основанных на предлагаемой модели, возможно расширение системы оценок.

Для данных (переменных, констант) предлагается использовать следующие оценки:

– оценку *S* (safe) получают данные, которые могут быть подставлены в текст SQL-запроса и при этом не приведут к возникновению уязвимостей;

– оценку *U* (unsafe) получают данные, которые при подстановке в текст SQL-запроса могут изменить его логику, т. е. привести к SQL-инъекции.

Наилучшей оценкой считается оценка *S*, наихудшей — оценка *U*.

Оценка in-параметра равна наихудшей возможной оценке фактического параметра (данных), при которой его передача в процедуру не приведет к возникновению уязвимостей:

– оценку *S* (safe) получают in-параметры, для которых передаваемое значение должно иметь оценку не ниже *S*, т. е. быть надлежащим образом обработано;

– оценку *U* (unsafe) получают in-параметры, для которых передаваемое значение должно иметь оценку не ниже *U*, т. е. может быть любым.

Оценки для out-параметров получаются в ходе абстрактной интерпретации кода процедуры. Значение оценки out-параметра — это значение наихудшей оценки для данных, которые процедура может возвращать через out-параметр. Эта оценка при анализе вызывающей процедуры становится оценкой переменной, в которую записываются соответствующие данные.

Оценки для out-параметров:

– оценку *S* (safe) получают out-параметры, через которые возвращаются данные с оценкой не ниже *S*;

– оценку *U* (unsafe) получают out-параметры, через которые возвращаются данные с оценкой не ниже *U*, т. е. которые следует в дальнейшем считать потенциально опасными.

В связи со спецификой задачи по обнаружению уязвимостей, а также из практических соображений система оценок может быть расширена. Одним из способов расширения системы оценок является снижение числа ложноположительных срабатываний. При использовании бинарной системы оценок они могут возникать в ряде случаев, например:

– web-приложение в настоящее время не имеет уязвимости, однако она может возникнуть в результате внесения изменений в код приложения;

– алгоритмы экранирования данных, созданные разработчиками web-приложения, оказываются слишком сложными для автоматического распознавания моделью обнаружения уязвимостей.

Следует понимать, что модель предполагает обнаружение не только уязвимостей, позволяющих успешно провести атаку, но и потенциальных уязвимостей, которые могут стать эксплуатируемыми при изменении кода web-приложения в ходе его дальнейшей разработки и/или сопровождения. Наличие ложноположительных срабатываний, таким образом, говорит в первую очередь об общем качестве кода приложения, включая его надежность и сопровождаемость, а также ряд других характеристик.

Тем не менее, для практических целей систему оценки имеет смысл дополнить третьим значением — UDS (user-defined safe — безопасное по определению пользователя), позволяющим пользователю программного средства обнаружения уязвимостей указать, что некоторые параметры процедур, несмотря на полученную при автоматическом анализе оценку *U*, на самом деле должны иметь оценку *S*.

В первом приближении web-приложение может быть представлено в виде ориентированного графа, вершины которого соответствуют процедурам, а дуги отражают выполняемые этими процедурами вызовы (направление — от вызывающей к вызываемой). На рис. 2 представлен упрощенный пример такого графа, причем процедура является точкой входа.

Очевидно, что анализ кода web-приложения следует начинать с процедур, для которых в графе отсутствуют исходящие дуги к процедурам, не имеющим оценок. Для приведенного примера такими процедурами являются p_7 и p_9.

Особое внимание следует уделить случаям рекурсии, которые в графе представлены циклами (вершина p_8 и множество вершин $\{p_2, p_4, p_5\}$). Анализ таких случаев затруднен тем, что процедуры, участвующие в рекурсии, взаимно зависимы. Простейшее решение данной проблемы — назначить in-параметрам таких процедур оценки *S*, а out-параметрам — оценки *U*. Такое решение может привести к росту числа ложноположительных срабатываний, что, однако, не является проблемой и может быть устранено путем выборочного назначения оценок UDS.

Метод оценки качества web-приложений. В настоящее время при обсуждении вопросов обеспечения качества программных средств (и, в частности, web-приложений) целесообразно опираться на международный стандарт ISO/IEC 25010:2011 [5]. Модель качества продукта, приводимая в нем, является трехуровневой: оценка состоит из характеристик, подхарактеристик и мер. Стандарт регламентирует состав характеристик и подха-

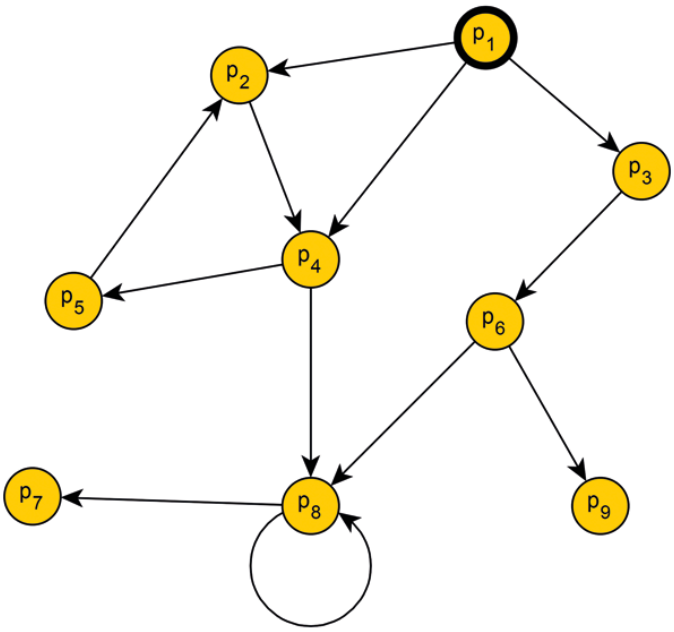

Рис. 2. Пример представления кода web-приложения в виде графа

рактеристик, расширение модели допускается только путем добавления мер к уже существующим подхарактеристикам.

Побочным продуктом статического анализа исходных кодов web-приложения являются сведения, которые могут быть использованы для оценки его качества. В статье «Модель оценки качества web-приложений, основанная на обнаружении уязвимостей к SQL-инъекциям» [6] вводится несколько понятий, позволяющих более кратко и точно описывать способ вычисления мер и их физический смысл. На основе этой информации при оценке качества web-приложений в контексте обнаружения уязвимостей предлагается использовать за основу модель качества web-приложений, структура которой приведена на рисунке 3.

Предлагаемые для использования в рамках модели качества web-приложения меры можно рассчитать на основании собранных при анализе

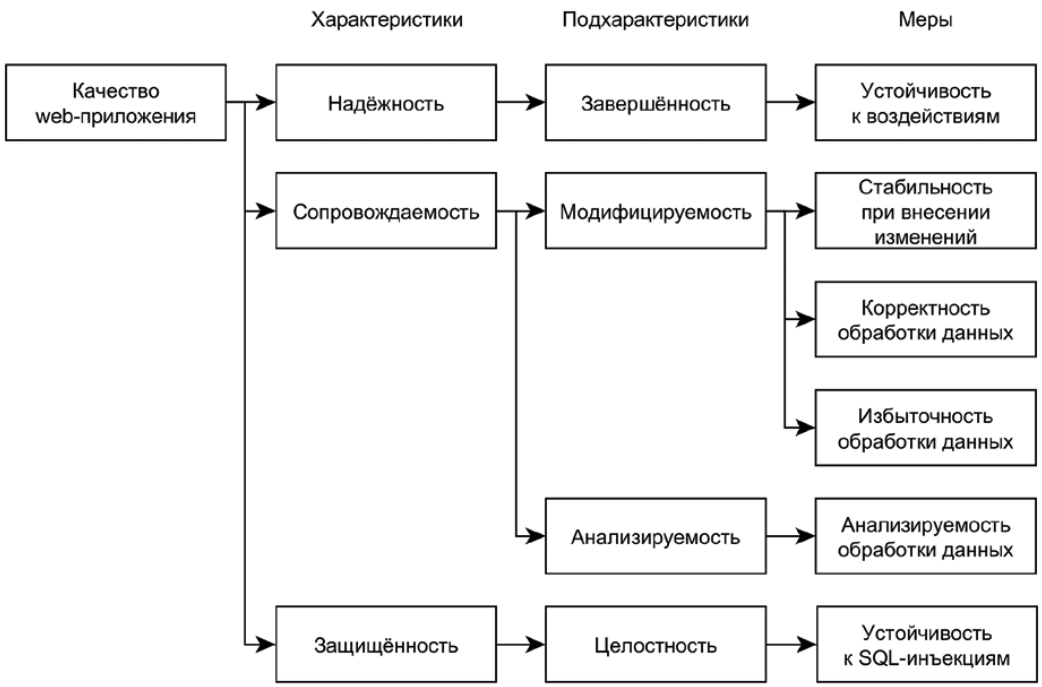

Рис. 3. Модель качества web-приложений в контексте обнаружения уязвимостей

кода web-приложения данных по одной из двух формул:

$$X = \frac{A}{B}, \quad (1)$$

$$X = 1 - \frac{A}{B}, \quad (2)$$

где X — рассчитываемое значение меры;
A — абсолютная величина, характеризующая то или иное свойство ПС;
B — абсолютная величина, характеризующая максимально возможное значение A ($B > 0$).

Порядок расчета значений предлагаемых мер представлен в таблице 1. Для всех предложенных мер выполняются следующие условия: $A \le B$, $0 \le X_i \le 1$ ($i = 1...6$). При этом большему значению соответствует более высокий уровень качества, т. е. меры удовлетворяют критериям обоснованности — трассируемости и непротиворечивости [7].

Получение исходных данных для вычисления значения меры «Устойчивость к SQL-инъекциям» возможно в автоматизированном режиме при условии расширения объема исходных данных (имеющихся у анализирующего ПС) сведениями о назначении стандартных процедур.

Для получения интегральной оценки качества web-приложения, основанной на предложенных мерах, рекомендуется осуществить выбор весовых коэффициентов в зависимости от значимости каждой из мер в рамках конкретного web-приложения, подлежащего оценке. После этого значения подхарактеристик, характеристик и интегральной оценки качества могут быть получены по аналогии с формулами, предложенными в ГОСТ 28195-99 [8]:

$$S_{ij} = \sum_{k=1}^{m} \left(M_{jk} \cdot V_{jk}^{M} \right), \quad (3)$$

$$C_i = \sum_{j=1}^{s} \left(S_j \cdot V_j^{S} \right), \quad (4)$$

$$Q = \sum_{i=1}^{c} \left(C_i \cdot V_i^{C} \right), \quad (5)$$

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

Таблица 1. Меры качества web-приложений в контексте обнаружения уязвимостей

Мера	Формула	Описание
1. Достаточность обработки данных	(2)	A — количество точек входа данных, потенциально допускающих проведение атак B — общее количество точек входа данных
2. Стабильность при внесении изменений	(1)	A — количество формальных in-параметров с оценкой U B — общее количество формальных in-параметров *При этом стандартные процедуры не учитываются.*
3. Корректность обработки данных	(1)	A — количество in-параметров, при передаче которых соблюдаются правила применения оценок B — общее количество in-параметров, передаваемых при вызовах процедур
4. Избыточность обработки данных	(1)	A — количество in-параметров, при передаче которых оценка фактического параметра выше (лучше) оценки формального параметра B — общее количество in-параметров, передаваемых при вызовах процедур *При этом стандартные процедуры не учитываются.*
5. Анализируемость обработки данных	(2)	A — количество in-параметров с оценкой UDS B — общее количество параметров с оценкой S или UDS
6. Устойчивость к SQL-инъекциям	(2)	A — количество in-параметров процедур взаимодействия с СУБД, имеющих оценку S и получающих параметры с оценкой U B — общее количество in-параметров процедур взаимодействия с СУБД *При этом учитываются только стандартные механизмы (процедуры) языка / библиотеки для взаимодействия с СУБД.*

вычислении значения подхарактеристики;
S_{ij} — значение j-й подхарактеристики i-й характеристики качества;
V_{ij}^{S} — значение весового коэффициента подхарактеристики S_{ij};
s — общее количество подхарактеристик, используемых при вычислении значения характеристики;
C_i — значение i-й характеристики качества;
V_i^C — значение весового коэффициента характеристики C_i;
c — 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 таким образом, чтобы выполнялись следующие условия:

$$\sum_{k=1}^{m} V_{jk}^{M} = 1 \quad (6)$$

$$\sum_{j=1}^{s} V_{ij}^{S} = 1 \quad (7)$$

$$\sum_{i=1}^{c} V_i^C = 1 \quad (8)$$

Предлагаемая модель может дополняться другими мерами в соответствии со стандартом ISO / IEC 25010:2011 для получения оценки качества, учитывающей те или иные свойства web-приложения, не относящиеся к вопросам его уязвимости к SQL-инъекциям. В этом случае рекомендуется придерживаться тех же принципов выбора мер и их весовых коэффициентов.

Заключение. Рассмотренные в статье модель обнаружения уязвимостей к SQL-инъекциям, модель качества web-приложений и основанный на них метод оценки качества web-приложений предоставляют возможность автоматизации трудоемкого процесса анализа исходных кодов web-приложений на наличие уязвимостей к SQL-инъекциям, а также обеспечивают качество web-приложений путем отслеживания динамики изменений уровня их качества на различных этапах разработки и сопровождения.

Список литературы

1. OWASP Top 10 2017. The Ten Most Critical Web Application Security Risks. [Electronic resource]. – Mode of access: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf. – Date of access: 27.11.2017.
2. Бахтизин, В. В. Модель обнаружения уязвимостей в web-приложениях / В. В. Бахтизин, Д. Е. Оношко // Докл. БГУИР. – 2016. – №1 (95). – С. 5–11.
3. Cousot P. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints / P. Cousot, R. Cousot // Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, January 1977. – Los Angeles: ACM, 1977. – Pp. 238–252.
4. Spolsky, J. Making Wrong Code Look Wrong – Joel on Software / J. Spolsky // JOEL ON SOFTWARE [Electronic resource]. – Mode of access: http://www.joelonsoftware.com/articles/Wrong.html. – Date of access: 21.12.2014.
5. ISO/IEC 25010:2011. Системная и программная инженерия — Требования к качеству и оценка программного продукта (SQuaRE) — Модели качества систем и программных средств. – Введ. 01.03.2011. – Женева: ISO/IEC, 2011.
6. Оношко, Д. Е. Модель оценки качества web-приложений, основанная на обнаружении уязвимостей к SQL-инъекциям / Д. Е. Оношко, В. В. Бахтизин // Докл. БГИР. – 2016. – №3 (97). – С. 37–43.
7. Бахтизин, В. В. Метрология, стандартизация и сертификация в информационных технологиях: учеб. пособие: в 2 ч. / В. В. Бахтизин, Л. А. Глухова. – Минск: БГУИР, 2016. – Ч. 2. – 343 с.
8. Оценка качества программных средств. Общие положения: ГОСТ 28195-99. Введ. 2000-01-03. – М., 1998. – 49 с.

References

1. OWASP Top 10 2017. The Ten Most Critical Web Application Security Risks. Available at: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf (accessed 27.11.2017).
2. Bakhtizin V. V. A web-application vulnerability detection model. Doklady BGUIR. 2016, no. 1, pp.5–11 (In Russian).
3. Cousot P., Cousot R. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, 1977, pp. 238–252.
4. Spolsky J. Making Wrong Code Look Wrong – Joel on Software. Available at: http://www.joelonsoftware.com/articles/Wrong.html (accessed 21.12.2014).
5. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Geneva, ISO/IEC, 2011.

6. Onoshko D. E. A web-application quality assessment model based on SQL-injection vulnerability detection. Doklady BGUIR, 2016, no. 3, pp. 37–43 (In Russian).
7. Bakhtizin V. V., Glukhova L. A. *Metrologija, standartizacija i sertifikacija v informacionnyh tehnologijah* [Metrology, standardization and certification in information technologies]. Minsk, BSUIR, 2016. 343 p. (In Russian).
8. GOST 28195-99. *Ocenka kachestva programmnyh sredstv. Obshhie polozhenija* [State Standard 28195-99. Quality control of software system. General principles]. Moscow, Standartinform Publ., 1998. 49 p. (In Russian).

Received: 02.03.2018 *Поступила: 02.03.2018*