МОДЕЛЬ АВТОМАТИЗИРОВАННОГО ОБНАРУЖЕНИЯ УЯЗВИМОСТЕЙ В WEB-ПРИЛОЖЕНИЯХ

Д. Е. Оношко, В. В. Бахтизин
Кафедра программного обеспечения информационных технологий, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: {onoshko, bww}@bsuir.by

Статья посвящена проблеме автоматизации обнаружения уязвимостей в web-приложениях. Предлагается модель их автоматизированного обнаружения, основанная на статическом анализе исходных кодов web-приложения. Рассматриваются некоторые особенности применения предлагаемой модели на практике. Показано, что результаты, получаемые при применении модели, могут быть использованы для оценки качества web-приложений.

Введение

Тенденция к переходу от классических desktop-приложений к web-приложениям, предоставляющих пользователям значительно большую мобильность, в то же время привела к повышению значимости вопросов качества программных средств (ПС). Возможность получения доступа к web-приложению из любой точки земного шара, являясь полезным свойством, вместе с тем значительно повышает риски, связанные с возможным наличием в web-приложениях уязвимостей, эксплуатация которых злоумышленниками может иметь значительные негативные последствия.

По данным организации OWASP, по состоянию на 2013 год наиболее распространённым видом угроз, причём как для web-, так и для desktop-приложений, остаются уязвимости к инъекциям, в особенности к такому их подвиду, как SQL-инъекции [1]. Уязвимость web-приложения к SQL-инъекциям предполагает наличие в нём логических ошибок, связанных с недостаточной или некорректной обработкой данных, поступающих извне, т.е. от пользователей. Разработано значительное количество технических решений, позволяющих при правильном их использовании предотвратить возникновение таких ошибок, однако неправильное использование этих решений, чаще всего непредумышленное, по-прежнему оставляет возможность возникновения уязвимостей. При этом оказывается возможным формальное описание этих ошибок, а значит, и разработка методов их автоматизированного обнаружения.

I. Механизм эксплуатации уязвимостей

Для хранения данных в большинстве web-приложений используются системы управления базами данных (СУБД), взаимодействие с которыми осуществляется посредством отправки запросов на языке SQL, являющихся по своей сути фрагментами текста, сформированными по правилам этого языка. В качестве примера такого запроса можно привести следующий текст:

```
SELECT * FROM ‘Users‘
WHERE Login    = ’Username’
  AND Password = SHA1(’itsp@ssw0rdexample’);
```

В случае, если используемая СУБД или библиотека не предоставляет возможности использования подготовленных выражений (prepared statements), конкретные данные подставляются в текст запроса средствами самого языка программирования. Если эти данные не были должным образом обработаны перед подстановкой (в частности, не были экранированы специальные символы SQL), появляется возможность сформировать данные, которые изменят предусмотренную разработчиками логику работы запроса. Например, при вводе в поле имени пользователя текста `Admin’ OR 1 = 1; --` будет получен следующий запрос:

```
SELECT * FROM ‘Users‘
WHERE Login    = ’Admin’ OR 1 = 1; --’
  AND Password = SHA1(’itsp@ssw0rdexample’);
```

При этом запрос будет возвращать данные пользователя `Admin` без проверки пароля. Использование подготовленных выражений позволяет избежать этой проблемы за счёт раздельной передачи самого текста запроса и подставляемых данных, однако этот механизм имеет ряд ограничений, из-за которых в некоторых случаях запрос по-прежнему приходится формировать с помощью подстановок и конкатенации строк. Кроме того, поддержка подготовленных выражений не навязывает их повсеместное использование, а значит, в ходе рефакторинга или из-за невнимательности программиста необработанные данные по-прежнему могут попадать в текст запроса, внося в web-приложение уязвимости.

II. Модель обнаружения уязвимостей

Для автоматизации обнаружения уязвимостей предлагается использовать модель обнаружения уязвимостей, основанную на результатах статического анализа исходных кодов. В

рамках модели web-приложение рассматривается как множество обобщённых процедур $Q = \{P_1, P_2, \ldots, P_N\}$, где N – общее количество таких процедур, а P_N – главная процедура web-приложения (точка входа). В исходном коде web-приложения обобщённой процедуре могут соответствовать процедура/функция, метод, операция и другие языковые конструкции.

Каждая обобщённая процедура $P_i \in Q$ характеризуется набором in- и out-параметров. В рамках модели in-параметры соответствуют элементам данных, которые являются исходными для данной процедуры, out-параметры – элементам данных, которые вычисляются самой процедурой и, таким образом, являются результатами её работы.

Обнаружение уязвимостей с использованием предлагаемой модели основывается на назначении оценок in- и out-параметрам процедур, а также передаваемым между процедурами данным. В простейшем случае оценки носят бинарный характер, их значения условно обозначаются буквами U (unsafe) и S (safe), причём $U < S$. Для передаваемых между процедурами данных эти оценки имеют следующий смысл:

- U – данные не были должным образом обработаны, подстановка этого значения в текст SQL-запроса может стать причиной уязвимости web-приложения;
- S – данные были должным образом обработаны, подстановка этого значения в текст SQL-запроса не приведёт к уязвимости web-приложения.

Оценка in-параметра равна наихудшей (наименьшей) оценке, которую должны иметь принимаемые через этот параметр данные. Для бинарной системы оценок:

- U – можно передавать как U-, так и S-данные;
- S – можно передавать только S-данные.

Оценка out-параметра равна наихудшей (наименьшей) оценке, которую могут иметь возвращаемые через этот параметр данные. Для бинарной системы оценок:

- U – могут возвращаться необработанные данные;
- S – возвращаются обработанные данные.

Оценки для in- и out-параметров стандартных процедур (в том числе используемых для взаимодействия с СУБД, получения исходных данных от пользователя и т.д.), известны заранее. Пусть на i-м шаге анализа известны оценки для параметров процедур $Q_i = \{P_1, P_2, \ldots, P_{C(i)}\}, Q_i \subseteq Q$, где $C(i) \leq N$ – количество уже проанализированных процедур. Тогда, поскольку все процедуры принадлежат одному и тому же web-приложению, существует процедура $P_{C(i)+1}$, зависящая только от процедур из множества Q_i. Анализируя операторы процедуры $P_{C(i)+1}$, можно получить оценки для её параметров. Анализ продолжается до тех пор, пока не будут назначены оценки параметрам всех процедур из Q.

III. Особенности практического применения модели

При практическом применении модели для анализа реальных web-приложений необходимо учитывать ряд их особенностей. Так, в частности, при использовании программного средства, выполняющего автоматизированное обнаружение уязвимостей с использованием предлагаемой модели, возможны ситуации, когда преобразования поступивших извне данных, выполняемые в коде web-приложения, остаются нераспознанными (т.е. данные получают оценку U вместо S). В этом случае следует говорить о ложноположительном результате.

В качестве одного из способов минимизации количества ложноположительных срабатываний при автоматизированном обнаружении уязвимостей предлагается расширение системы оценок введением оценки UDS (user-defined safe). Предполагается, что используемое ПС предоставляет возможность пользователю явным образом указать, что некоторые параметры следует рассматривать, как имеющие оценку S.

Данные, полученные в результате анализа исходного кода web-приложений с использованием предлагаемой модели, могут быть использованы для оценки качества web-приложения. Так, например, наличие большого количества оценок UDS может говорить о недостаточном использовании возможностей, предоставляемых СУБД, либо высокой сложности анализа исходного кода, что может потребовать принятия организационных мер. Большое количество случаев, когда оценка фактически передаваемых данных выше, чем оценка соответствующего in-параметра, может характеризовать запас устойчивости web-приложения к SQL-инъекциям, что говорит о низкой вероятности возникновения уязвимостей в ходе дальнейшей разработки или сопровождения web-приложения.

Заключение

Предлагаемая модель обнаружения уязвимостей в web-приложениях, основанная на статическом анализе исходных кодов, позволяет обнаруживать как эксплуатируемые, так и потенциальные уязвимости, а также производить на основании полученных данных оценку качества web-приложения. Отслеживание динамики изменения результатов анализа исходных кодов с использованием предлагаемой модели позволяет принимать управленческие решения на ранних этапах жизненного цикла начиная с момента появления первого прототипа.

1. Top 10 2013 // Open Web Application Security Project (OWASP) [Electronic resource]. – 2013. – Mode of access: https://www.owasp.org/index.php/Top_10_2013-Top_10 – Date of access: 08.09.2016.