Важным моментом в применении данного принципа является то, что необходимо обеспечить порядок применения изменений кода инфраструктуры к самой инфраструктуре. Для минимизации рисков необходимо организовать полный цикл работы с кодом инфраструктуры:

- версионирование кода инфраструктуры – исходный код хранится в системе контроля версий;
- ревизия и обсуждение кода ключевыми членами команды перед помещением в репозиторий (код-ревью);
- модульное тестирование (где возможно) – позволяет убедиться, что отдельные модули и подмодули после изменения кода возвращают ожидаемые результаты;
- автоматизированное функциональное тестирование кода инфраструктуры;
- автоматизированное развертывание – инфраструктура разворачивает и тестирует саму себя;
- мониторинг развернутой инфраструктуры и контроль за процессом извне;

Таким образом, организация вышеописанного процесса позволяет добиться следующих преимуществ:

1. Быстрое внесение и применение изменений. Любое изменение – это изменение кода, которое, проходя через все системы автоматически тестируется и применяется.
2. Возможность быстрого развертывания всей инфраструктуры или ее части с нуля.
3. Самодокументируемость конфигурации инфраструктуры.
4. Версионность инфраструктуры.
5. Повышение надежности.

Следовательно, предложенный подход позволяет существенно упростить настройку и поддержку инфраструктуры жизненного цикла программного обеспечения, а также заметно снизить риски, связанные с ней.

Список использованных источников:

1. Morris K. Infrastructure as Code — O'Reilly, 2016.
2. Taylor M. Learning Chef. A Guide to Configuration Management and Automation — O'Reilly, 2014

МЕТОД ОБНАРУЖЕНИЯ УЯЗВИМОСТЕЙ К SQL-ИНЪЕКЦИЯМ В WEB-ПРИЛОЖЕНИЯХ

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Оношко Д.Е.

Бахтизин В.В. — к.т.н., профессор

Широкое использование web-приложений и повышение их сложности для решения новых задач поднимают значимость оценки их качества на новый уровень. Обнаружение уязвимостей к SQL-инъекциям, являющихся распространённым видом уязвимостей web-приложений, представляет собой важную часть такой оценки.

По состоянию на 2013 год по данным OWASP наиболее распространённым классом уязвимостей существующих приложений являются уязвимости-инъекции [1]. В web-приложениях наиболее частым видом инъекций являются SQL-инъекции. Ошибки, допущенные разработчиками при разработке алгоритмов обработки данных, позволяют злоумышленнику изменить поведение web-приложения путём формирования запросов, использующих эти ошибки, и являются причиной уязвимости web-приложений к SQL-инъекциям.

Большинство методов обнаружения уязвимостей к SQL-инъекциям основывается на анализе поведения web-приложения в условиях эксплуатации, т.е. на динамическом анализе. Однако такие методы по своей сути являются частным случаем функционального тестирования, и, следовательно, их эффективность определяется выбором тестовых данных, для которых производится анализ поведения, что в свою очередь означает невозможность обнаружения всех возможных ошибок данного класса. Статический анализ, т.е. анализ исходных кодов без запуска web-приложения, напротив, позволяет, последовательно анализируя отдельные части web-приложения, формально доказать корректность алгоритмов обработки данных либо выявить ошибки в них, а значит, получить ответ на вопрос о наличии в web-приложении эксплуатируемых и потенциальных уязвимостей к SQL-инъекциям. Вместе с тем, являясь рутинной процедурой, подобный анализ может быть автоматизирован посредством программного средства контроля качества кода, основанного на абстрактной интерпретации исходных кодов web-приложения.

Предлагаемый метод обнаружения уязвимостей основан на рассмотрении web-приложения как множества процедур $Q = \{P_1, P_2, \ldots, P_N\}$, вызывающих друг друга с некоторыми параметрами. Обмен данными между процедурами может реализовываться в различных языках программирования по-разному, поэтому при использовании метода предлагается приводить все возможные взаимодействия к одному из двух видов формальных параметров: in-параметры (данные, передаваемые в процедуру) и out-параметры (данные, возвращаемые из процедуры). Формальным параметрам процедур назначаются оценки, в простейшем случае — бинарного характера: «опасный» (U) или «безопасный» (S), причём U < S.

Первоначально анализатору известны оценки только для стандартных процедур. Пусть на i-м шаге известны оценки параметров и возвращаемых значений для процедур $Q_i = \{P_1, P_2, \ldots, P_{C_i}\}$, $Q_i \subseteq Q$, где $C_i < N$ — количество таких процедур на i-м шаге. Поскольку все процедуры принадлежат одному и тому же приложению, существует процедура P_{C_i+1}, зависящая только от процедур из Q_i. Получение оценок для её параметров может быть сведено к последовательному анализу её операторов с целью назначения оценок для используемых в ней переменных. Вычисленные оценки параметров процедуры P_N (им соответствуют поступающие от пользователя данные) должны иметь значение U. Параметры, для которых это не выполняется, являются потенциально уязвимыми. Анализируя путь, по которому была получена оценка, можно выявить причину возникновения уязвимости и предложить способы её устранения.

В общем случае можно сформулировать 5 правил, описывающих процесс назначения оценок и проверки web-приложения на наличие уязвимостей к SQL-инъекциям.

1. Оценка для переменной, передаваемой в качестве in-параметра, должна быть лучше (выше) оценки данного параметра.
2. Оценка переменной, принимающей значение из out-параметра, выбирается как худшая (меньшая) из её собственной оценки в данной точке программы и оценки out-параметра.
3. Всем литералам назначаются оценки S.
4. При наличии у переменной или out-параметра нескольких различных оценок выбирается наихудшая (наименьшая).
5. При наличии у in-параметра нескольких различных оценок выбирается наилучшая (наибольшая).

Предлагаемый метод позволяет обнаруживать как реальные (эксплуатируемые при нынешнем состоянии кода web-приложения) уязвимости, так и потенциальные. Собранные с ходе анализа web-приложения с помощью данного метода сведения могут использоваться в оценке качества web-приложения.

Список использованных источников

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: 01.03.2016.

СНИЖЕНИЕ РАЗМЕРНОСТИ ИССЛЕДУЕМЫХ ДАННЫХ НА ОСНОВЕ ЭВРИСТИЧЕСКОЙ ВОЗМОЖНОСТНОЙ КЛАСТЕРИЗАЦИИ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Петюкевич Н.С.

Вятченин Д. А. – канд. филос. наук, доцент

При необходимости обработки больших массивов данных на первой стадии возникает задача предварительного анализа имеющихся данных для:

– определения возможного числа классов, на которые «расслаивается» исследуемая совокупность;
– выделения в ней аномальных наблюдений;
– проецирования исследуемой совокупности на плоскость, к примеру, двух главных компонент или наиболее информативных признаков.

С этой целью может быть применен аппарат эвристической возможностной кластеризации.

Разведочный анализ данных традиционно используется, когда, с одной стороны, у аналитика имеется таблица многомерных данных, а с другой – информация о природе этих данных неполна или вовсе отсутствует. К задачам разведочного анализа данных традиционно [1] относят:

– проецирование данных в двумерное пространство;
– обнаружение аномальных наблюдений и очистка данных;
– обнаружение возможного числа кластеров в исследуемой совокупности.

При этом проецирование исследуемой совокупности в двумерное признаковое пространство предполагает вначале нахождение соответствующих признаков, на плоскость которых будет проецироваться исследуемая совокупность.

В последние 20-30 лет огромную популярность получили методы автоматической классификации, основанные на концепции теории нечетких множеств, предложенной выдающимся американским математиком Л.А. Заде [2], что обусловлено их высокой точностью и содержательной осмысленностью в сравнении с традиционными методами кластеризации. Одним из таких подходов стал эвристический метод возможностной кластеризации, предложенный в работе [3] и получивший дальнейшее развитие в задачах классификации и