МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

DOI 10.17223/20710410/23/6

УДК 519.7

МЕТОД ОБНАРУЖЕНИЯ АТАК ТИПА «ОТКАЗ В ОБСЛУЖИВАНИИ» НА WEB-ПРИЛОЖЕНИЯ

С. Н. Сорокин

Московский институт электроники и математики Национального исследовательского университета «Высшая школа экономики», г. Москва, Россия

E-mail: sergey-dcm@yandex.ru

Рассматривается метод обнаружения атак типа «отказ в обслуживании» на web-приложения, основанный на использовании многослойного персептрона. Описаны основные вопросы, связанные с использованием нейронной сети: процесс сбора и подготовки статистических данных, особенности формирования показателей, описывающих состояние web-приложения, а также вопросы обучения и оценки качества работы нейронной сети.

Ключевые слова: *обнаружение атак, отказ в обслуживании, DoS, web-приложение.*

Введение

За сравнительно небольшой промежуток времени в сети Интернет возникло большое количество web-приложений, предоставляющих пользователям различную информацию или оказывающих услуги. В Интернет постепенно переносятся традиционные приложения, что обеспечивает возможность удобного взаимодействия с ними с помощью различных устройств и практически в любом месте. В связи с этим происходит постоянный рост числа пользователей web-приложений и возникает проблема обеспечения надежного функционирования таких систем.

Современные web-приложения являются достаточно сложным программным обеспечением, которое может обрабатывать большие объёмы данных и формировать ответы на запросы пользователей. Сложность функционирования web-приложений приводит к возникновению высоких нагрузок на аппаратные ресурсы, а большое количество пользователей делает такие приложения интересной мишенью для компьютерных атак. Атаки на web-приложения могут проводиться с разными целями, например с целью кражи персональной информации или информации о кредитных картах пользователей.

Одним из популярных типов атак на web-приложения является «отказ в обслуживании». Количество атак данного типа в 2013 г. увеличилось на 32,4 % по сравнению с 2012 г. [1]. Цель атак типа «отказ в обслуживании» — полный или частичный вывод из строя web-приложения, что лишает пользователей возможности в полной мере его использовать. «Отказ в обслуживании» web-приложения может достигаться с помощью атак, направленных на его инфраструктуру, либо может проводиться на прикладном уровне, при этом осуществляется взаимодействие с приложением по протоколу HTTP.

Атаки, направленные на инфраструктуру web-приложения, достаточно хорошо изучены; существует большое количество методов обнаружения таких атак:

— отклонение параметров данных;
— изменение вероятностных параметров данных;
— обнаружение с помощью извлечения данных;
— восстановление пути по сигнатурам;
— метод разметки пакетов;
— генерация служебных пакетов и др.

Данные методы плохо применимы для анализа атак на web-приложения, проводимых на прикладном уровне. Это связано с тем, что различные компоненты современных web-приложений сильно отличаются частотой использования и потреблением аппаратных ресурсов, а также с тем, что в современных web-приложениях происходит постоянный рост количества доступных для просмотра страниц.

На прикладном уровне к «отказу в обслуживании» web-приложения могут приводить эксплуатация различных уязвимостей в web-приложении и превышение допустимого количества запросов, которые могут быть одновременно обработаны web-приложением.

Увеличение количества запросов, находящихся одновременно на обработке в web-приложении, может происходить:

— за счёт увеличения интенсивности потока запросов;
— в процессе атак, использующих уязвимости в протоколе HTTP (например, `slow HTTP POST` [2]).

В работе рассматриваются атаки «отказ в обслуживании», возникающие вследствие увеличения интенсивности потока поступающих запросов, вызванного путем имитации действий пользователей автоматическими программами (ботами). Данный вид атак в 2013 г. составил 86 % от общего количества атак типа «отказ в обслуживании» на web-приложения, проводимых на прикладном уровне [1].

В настоящее время не существует общепринятых классификаций и эффективных методов обнаружения данного вида атак. На прикладном уровне известен метод обнаружения атак типа «отказ в обслуживании» на основе оценки вероятности потерь заявок, однако он применим только в случае низкоактивных атак [3]. В связи с этим задача создания метода обнаружения атак типа «отказ в обслуживании» на web-приложения, функционирующего на прикладном уровне, является актуальной.

1. Общая схема метода обнаружения атак типа «отказ в обслуживании» на web-приложения

Существует два основных подхода к обнаружению атак: обнаружение аномалий и обнаружение злоупотреблений [4]. При обнаружении аномалий составляется профиль нормального состояния системы, отклонение текущего состояния системы от профиля считается аномалией и делается заключение о возможной атаке. При обнаружении злоупотреблений задача состоит в том, чтобы описать профили системы в состоянии различных атак, чтобы в дальнейшем распознавать различные виды атакующих воздействий.

Для построения метода использован подход, связанный с обнаружением злоупотреблений с помощью нейронных сетей. Данный подход позволяет гибко адаптировать средство обнаружения атак к особенностям конкретного web-приложения (производить автоматическую настройку допустимых значений параметров), он успешно

используется для решения ряда смежных задач и обнаружения некоторых классов атак [5 – 7].

В качестве используемой нейронной сети целесообразно использовать сеть многослойного персептрона ввиду следующих особенностей [8]:

— эффективность решения задач аппроксимации;
— возможность работы с нелинейно-разделимыми входными показателями;
— простота реализации;
— высокая скорость работы сети — это делает возможным использование данного типа нейронной сети для обнаружения атак типа «отказ в обслуживании» в реальном времени.

Общая схема метода обнаружения атак типа «отказ в обслуживании» представлена на рис. 1.

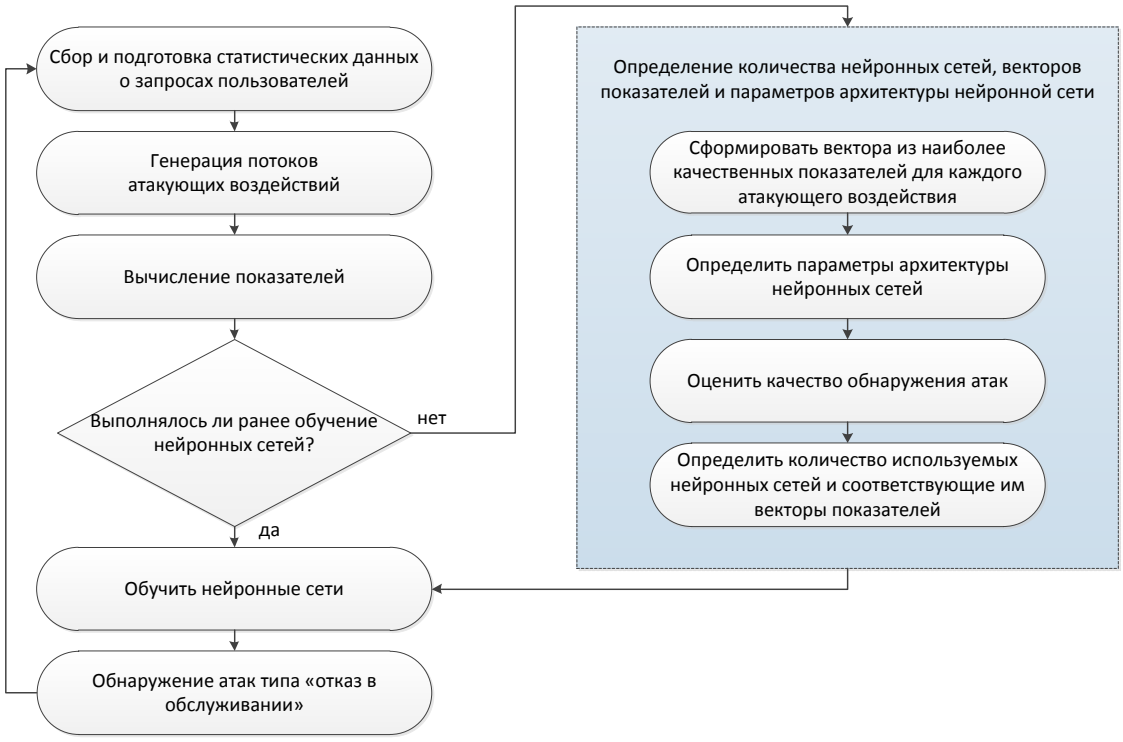

Рис. 1. Общая схема функционирования метода обнаружения атак типа «отказ в обслуживании» на web-приложения

Метод заключается в следующем:

1) подготовить статистическую информацию о поведении пользователей;
2) произвести генерацию статистической информации о различных видах атакующих воздействий, приводящих к «отказу в обслуживании»;
3) вычислить в рассматриваемом временном периоде (не меньше недели) значения показателей (и соответствующих показателям статистических структур);
4) провести оценку применимости показателей и сформировать множества показателей, наиболее подходящих для определения различных видов атак;

5) определить наилучшие параметры архитектуры нейронных сетей для различных видов атак и качество обнаружения атак;
6) сгруппировать различные виды атак в классы атакующих воздействий с целью уменьшения количества используемых нейронных сетей;
7) произвести обучение нейронных сетей;
8) использовать нейронные сети для обнаружения атак типа «отказ в обслуживании»;
9) по необходимости производить переобучение нейронных сетей на новых статистических данных.

2. Сбор статистических данных

Для описания запроса пользователя к web-приложению разработан следующий формат векторов данных:

$$(Time_i, SessionID_i, IsSessionStart_i, URL_i, Parameters_i, IP_i, Referer_i,$$
$$UserAgent_i, Latency_i, DocSize_i, Memory_i, CpuTime_i),$$

где

— i — порядковый номер запроса к web-приложению;
— $Time_i$ — время запроса, выраженное в формате `timestamp`;
— $SessionID_i$ — идентификатор сеанса;
— $IsSessionStart_i$ — индикатор начала сеанса;
— URL_i — URL-адрес запроса;
— $Parameters_i$ — GET-параметры запроса;
— IP_i — IP-адрес, с которого был сделан запрос;
— $Referer_i$ — адрес предыдущей посещенной страницы;
— $UserAgent_i$ — идентификационная строка клиентской программы пользователя;
— $Latency_i$ — время отклика сервера при обработке данного запроса;
— $DocSize_i$ — объём данных, которые были загружены пользователем в ответ на запрос;
— $Memory_i$ — количество оперативной памяти сервера, потраченное на обработку запроса;
— $CpuTime_i$ — процессорное время, потраченное на выполнение запроса.

Порядок подготовки статистических данных о запросах пользователей представлен на рис. 2.

После сбора статистики в предложенном формате необходимо произвести преобразование URL-запросов к каноническому виду с помощью правил преобразования адреса, удаления и задания порядка параметров запросов.

Введём следующие обозначения:

— URL — строка URL [9];
— $ScriptURL$ — строка URL без строки запроса (начальный фрагмент URL до символа «?»);
— n — максимально возможное количество параметров в запросе;
— s — количество всех возможных параметров в web-системе;
— $\{parameter_1, \ldots, parameter_s\}$ — множество всех возможных параметров в web-системе;
— $parameter_i^j$ — некоторый j-й параметр в запросе, $j \in \{1, \ldots, n\}$, $i \in \{1, \ldots, s\}$;
— $parameters = parameter_{i_1}^1 \& \ldots \& parameter_{i_k}^k$ — строка параметров, содержащая $k \in \{1, \ldots, n\}$ параметров, $i_k \in \{1, \ldots, s\}$;

Рис. 2. Порядок подготовки статистических данных

— $URLFilter(ScriptURL)$ — функция, описывающая алгоритм формирования пути к ресурсу на сервере, не содержащего специальных конструкций;
— $ParametersSequence(parameter_{i_1}, \ldots, parameter_{i_k})$ — функция, описывающая применение правил формирования последовательности параметров к некоторой последовательности параметров $parameter_{i_1}, \ldots, parameter_{i_k}$, $i_j \in \{1, \ldots, s\}$, $j \in \{1, \ldots, k\}$, $k \in \{1, \ldots, n\}$, результат работы функции не зависит от порядка следования аргументов;
— $ParameterPresence(parameter_i)$ — функция, описывающая применение правил удаления параметров к некоторому параметру $parameter_i$, $i \in \{1, \ldots, s\}$;
— . — операция «склейки» строк.

Во введённых обозначениях URL можно представить [9] как

$$URL = ScriptURL.?.parameters.$$

Канонической формой запроса для некоторого запроса URL называется запрос следующего вида:

$$CanonicalURL(URL) = URLFilter(ScriptURL).?.$$
$$.ParametersSequence \begin{pmatrix} ParameterPresence(parameter^1_{i_1}), \\ \ldots \\ ParameterPresence(parameter^k_{i_k}) \end{pmatrix}.$$

Алгоритм формирования пути $URLFilter(ScriptURL)$ может быть описан следующим образом: удалить из части строки $ScriptURL$, следующей за названием домена и портом, в следующей последовательности:

1) конструкции вида «//» из строки $ScriptURL$;

2) конструкции вида «/./» из строки $ScriptURL$;
3) конструкции вида «/$text$/../», где $text$ — произвольная строка, не содержащая символов «/».

Каноническое представление идентификационной строки необходимо для анализа строки $User - Agent$. Анализ исходных данных $User - Agent$ затруднён, поскольку существует большое количество браузеров, а эти строки могут существенно различаться даже у различных версий одного браузера. Поэтому необходимо выделить из строк $User - Agent$ значимые параметры и привести их строковую запись к единому виду. Этот процесс аналогичен получению канонического вида строк URL-запросов: необходимо сформировать правила удаления и упорядочивания параметров строки $User - Agent$.

В дальнейшем необходимо разбить все запросы по логическому и нагрузочному типу страниц.

В основе **кластеризации по логическому типу страницы** лежат следующие предположения:

— пользователям свойственно совершение некоторых последовательностей действий, которые имеют схожую смысловую нагрузку: например, просмотр новостей обычно подразумевает просмотр ленты новостей, чередующийся с просмотром страниц конкретных новостей;
— схожие действия пользователей (например, просмотр новости или профиля) приводят к возникновению схожих нагрузок на сервере.

Разбиение по логическому типу страниц производится в два этапа:

— выделение крупных логических разделов на множестве страниц web-приложения (например, форум, чат, новостная лента и т. п.);
— выделение различных типовых действий пользователей внутри логических разделов (например, для форума: чтение списка форумов, чтение списка тем, чтение темы и т. п.).

При **кластеризации по нагрузочным показателям** в один класс помещаются запросы, создающие на сервере сходные нагрузки на различные ресурсы. Кластеризация по нагрузочным характеристикам неинформативна для описания поведения пользователей, поскольку сходные по нагрузке запросы могут быть совершенно различными, не связанными между собой действиями пользователя web-приложения. Поэтому целесообразно использовать комбинированный подход, при котором сначала производится разбиение запросов по логическому типу, а потом запросы из одного логического класса ещё раз разбиваются на классы по нагрузочным показателям.

Для разбиения по нагрузочным характеристикам можно использовать алгоритм построения минимального остовного дерева [10].

Для формирования показателей, учитывающих зависимости между запросами пользователей, необходимо сформировать дополнительные статистические структуры, описывающие различные свойства поведения пользователей. В рамках исследования автором выделены следующие направления анализа зависимостей:

— зависимость появления запроса от предыдущего запроса;
— зависимость появления запроса от его расположения в пользовательских сеансах;
— зависимость появления запроса от последовательности предыдущих запросов.

Для учёта каждой из этих зависимостей предложено использовать следующие статистические структуры:

— матрица зависимостей от предыдущих запросов;

— матрица зависимостей от номера запроса в сеансе;
— лес сеансовых деревьев.

3. Формирование множества показателей, описывающих состояние web-приложения

Предлагается подход к формированию показателей для обучения нейронной сети, которые учитывают изменения посещаемости web-приложения в различное время суток, в разные дни недели. Данный подход позволяет также учесть естественные изменения популярности web-приложения.

Пусть:

— u — количество показателей, подаваемых на вход нейронной сети;
— K_g — некоторый показатель, подаваемый на вход нейронной сети, $g \in \{1, \ldots, u\}$;
— $K_{g_{i,t,h}}$ — некоторый показатель, подаваемый на вход нейронной сети, вычисленный i дней назад во время t за временной интервал h, $g \in \{1, \ldots, u\}$;
— $\overline{K} = (K_1, \ldots, K_u)$ — вектор длины u, характеризующий набор показателей, подающихся на вход нейронной сети.

В зависимости от значения длины временного интервала h сутки можно разбить на различное количество временных интервалов. Обозначим N_h — количество временных интервалов при выбранном значении h. Тогда $N_h = 86400/h$; 86400 — это длина суток, выраженная в секундах.

3.1. Учёт времени суток при работе с нейронной сетью

Пусть натуральное число $n_{t,h} \in \{1, \ldots, N_h\}$ — порядковый номер временного интервала, соответствующего времени t в текущих сутках, вычисляемый по формуле $n_{t,h} = [t \bmod 86400/h]$.

Для учёта изменений поведения пользователей в различное время суток необходимо добавить в векторы входных показателей нейронной сети $(K_1, \ldots, K_u)$ специальный показатель $n_{t,h}$.

3.2. Учёт дня недели и изменения популярности web-приложения

Для учёта данных изменений при расчёте показателей необходимо использовать результаты вычисления показателей в прошлом и специальный поправочный множитель (множитель популярности), характеризующий тенденцию изменения популярности web-приложения.

Схема вычисления прогнозного значения представлена на рис. 3.

Цифрами обозначены дни недели, отсчитанные от текущего дня в прошлое. Для прогнозирования текущего значения берётся прошлое значение и корректируется с помощью множителя популярности. Пример показателя, сформированного по данной схеме, представлен формулой (1).

Для формирования множества показателей можно использовать следующие классы показателей (классификация проведена по типу оцениваемых характеристик web-приложения):

— частотные показатели учитывают частотные свойства потока запросов к web-приложению;
— нагрузочные показатели основаны на изменениях нагрузочных характеристик запросов;
— структурные показатели учитывают структурные изменения в потоке запросов к web-приложению.

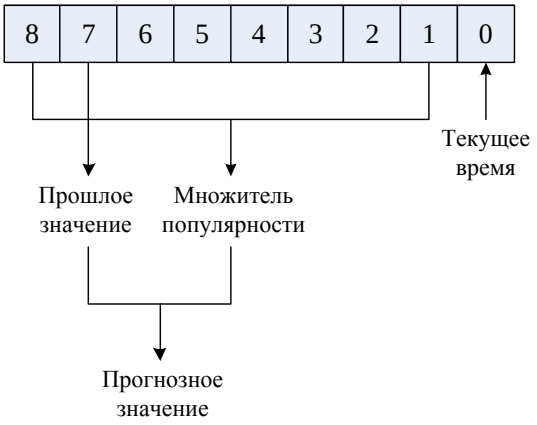

Рис. 3. Схема вычисления прогнозного значения показателя

Пример частотного показателя «Отношение посещаемости», отражающего, насколько текущая посещаемость отличается от прогнозной:

$$SessionsRatio_{0,t,h} = \frac{Sessions_{0,t,h}}{Sessions_{7,t,h} \cdot \dfrac{Sessions_{1,t,h}}{Sessions_{8,t,h}}}, \tag{1}$$

где $Sessions_{i,t,h}$ — количество сеансов пользователей, работавших с web-приложением i дней назад, рассчитанное во время t за временной интервал h.

4. Генерация атакующих воздействий

В процессе исследований рассмотрены модель поведения пользователей web-приложения и классификация основных типов автоматических программ, действия которых могут приводить к «отказу в обслуживании». Выделены следующие основные классы автоматических программ:

— поисковые;
— выгружающие содержимое страниц web-приложения;
— взаимодействующие с поисковой выдачей web-приложения;
— связанные с конкретными HTML-формами;
— собирающие информацию о web-приложении;
— атакующие.

На основании данной классификации можно сформировать статистические данные о запросах для различных типов атакующих воздействий. Более подробно вопросы классификации и имитации различных классов автоматических программ будут рассмотрены в других работах автора.

5. Обучение, определение параметров архитектуры и оценка качества работы нейронной сети

Обучение многослойного персептрона проводится с помощью алгоритма обратного распространения ошибки. Для обучения можно использовать следующий критерий останова [8]:

— ошибка обучения перестаёт уменьшаться (либо интенсивность уменьшения ошибки достигает порогового значения);

— ошибка достигает некоторого порогового значения;
— количество обучающих итераций достигает порогового значения (при этом не гарантируется правильность обучения многослойного персептрона).

Для определения параметров архитектуры необходимо провести последовательность тестов с увеличением количества слоёв нейронов и нейронов в одном слое. В результате тестов определяются параметры, когда минимальны

— средняя ошибка обучения на момент останова обучения;
— среднее количество обучающих эпох.

Для оценки качества работы нейронной сети используются следующие показатели:

— доля правильных ответов;
— доля ошибок первого рода;
— доля ошибок второго рода;
— доля неопределённых ответов.

6. Формирование векторов показателей для обучения нейронной сети

На первом этапе для формирования векторов показателей необходимо определить наиболее подходящие показатели для определения различных типов атакующих воздействий. Для этого используется методика оценки применимости показателей, основанная на вычислении следующих характеристик:

— амплитуды (разброса значений показателя);
— цикличности (степени различий между значениями показателя в различные дни недели);
— дифференциации (степени различий средних значений показателя на статистике нормального поведения пользователей и статистике атак).

На основании разработанной методики производится отбор наиболее подходящих для обнаружения различных видов атакующих воздействий показателей.

При таком подходе для обнаружения различных видов атакующих воздействий используются разные нейронные сети. Для ускорения работы метода предложена методика объединения различных типов атакующих воздействий в классы атак, что позволило использовать общий вектор показателей и одну нейронную сеть для атакующих воздействий, находящихся в одном классе атак.

Подробнее методики оценки качества показателей и формирования векторов показателей будут рассмотрены в других работах автора.

Заключение

Рассматривается метод, позволяющий производить обнаружение атак типа «отказ в обслуживании» на web-приложения на прикладном уровне. Данный метод был успешно протестирован и планируется к использованию при построении и внедрении систем обнаружения атак на web-приложения в органах государственной власти РФ.

Описанный подход к построению множества показателей позволяет обеспечить работоспособность метода с учётом изменения посещаемости web-приложения с течением времени суток, в различные дни недели и с учётом естественных изменений популярности web-приложений.

Методики оценки показателей и формирования векторов показателей и классов атак позволяют повысить производительность метода за счёт уменьшения количества используемых нейронных сетей и векторов показателей. Это позволяет использовать метод в реальном времени.

ЛИТЕРАТУРА

1. http://www.prolexic.com/knowledge-center-ddos-attack-report-2013-q4.html — Q4 2013 Global DDoS Attack Report. 2014.
2. https://community.qualys.com/blogs/securitylabs/2012/01/05/slow-read — Are you ready for slow reading? 2012.
3. *Щерба М. В.* Обнаружение низкоактивных распределенных атак типа «отказ в обслуживании» в компьютерных сетях: дис. ... канд. техн. наук. Омск, 2007. 130 с.
4. *Гамаюнов Д. Ю.* Обнаружение компьютерных атак на основе поведения сетевых объектов: дис. ... канд. физ.-мат. наук. М., 2007. 89 с.
5. *Жульков Е. В.* Построение нейронных сетей для обнаружения классов сетевых атак: дис. ... канд. техн. наук. СПб., 2007. 155 с.
6. *Александров И. С.* Разработка системы защиты web-приложений от автоматизированного копирования информации: дис. ... канд. техн. наук. М., 2003. 127 с.
7. *Хафизов А. Ф.* Нейросетевая система обнаружения атак на www-сервер: дис. ... канд. техн. наук. Уфа, 2004. 172 с.
8. *Хайкин С.* Нейронные сети: полный курс. 2-е изд., испр. М.: ООО «И.Д. Вильямс», 2006. 1104 с.
9. http://tools.ietf.org/html/rfc1738#section-3.3 — Uniform Resource Locators (URL). 1994.
10. *Menasce D. A. and Almeida V. A. F.* Capacity Planning for Web Services. Metrics, Models, and Methods. New Jersey: Prentice Hall PTR, 2001. 608 p.