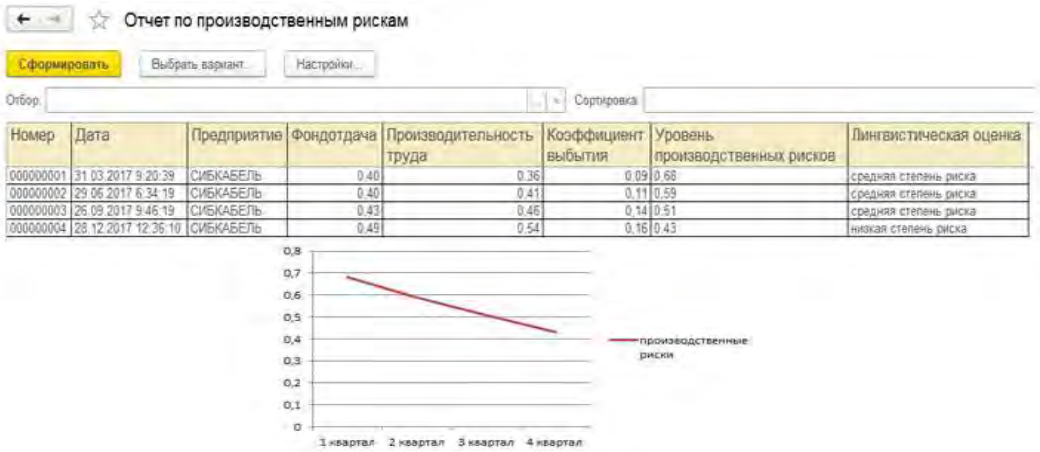

Рис. 3. Отчет по производственным рискам

Список литературы:

1. Джамансариев, Н. Б. Программное обеспечение для оценки риска банкротства предприятий [Электронный ресурс] / Н. Б. Джамансариев, Е. В. Телипенко // Современные технологии поддержки принятия решений в экономике: сборник трудов III Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых, 24-25 ноября 2016 г., г. Юрга / НИ ТПУ, Юргинский технологический институт (ЮТИ); под ред. А. А. Захаровой. – Томск: Изд-во ТПУ, 2016. – [С. 133-135].
2. Джамансариев Н. Б. Использование метода деревьев для оценки финансового состояния предприятия / Н. Б. Джамансариев; науч. рук. Е. В. Телипенко // Современные технологии поддержки принятия решений в экономике: сборник трудов Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых, 19-20 ноября 2015 г., г. Юрга. – Томск: Изд-во ТПУ, 2015. – [С. 78-79].
3. Джамансариев Н.Б. Оценка рисков машиностроительного предприятия, способных привести к банкротству / Н. Б. Джамансариев ; науч. рук. Е. В. Телипенко // Прогрессивные технологии и экономика в машиностроении: сборник трудов VIII Всероссийской научно-практической конференции для студентов и учащейся молодежи, 6-8 апреля 2017 г., Юрга. – Томск: Изд-во ТПУ, 2017. – [С. 152-154].

ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ СОВРЕМЕННЫХ WEB-ПРИЛОЖЕНИЙ

А.О. Ерёменко, студент, научный руководитель: Захарова А.А., д.т.н.
Юргинский технологический институт (филиал) Национального исследовательского Томского политехнического университета
652055, Кемеровская обл., г. Юрга, ул. Ленинградская, 26
E-mail: eao1@mail.ru, тел. +7(905)995-39-49

WEB-приложение независимо от его назначения является наиболее простым и, соответственно, самым распространенным способом доставки различного рода услуг до конечного пользователя. Повсеместное использование обусловливает высокие требования к надежности и бесперебойности работы WEB-приложений. Их удовлетворение возможно лишь путем применения широкого перечня защитных мер, одной из них является тестирование безопасности.

Хотя WEB-технологии зародились еще в начале 1990-х годов и прошли немалый путь в своем развитии, они остались все той же комбинацией транспорта (HTTP) и представления (HTML), обросшей множеством «фишек» для удовлетворения современных потребностей в броском внешнем виде и удобстве управления. Безопасность же до сих пор является лишь опцией в общей схеме, ввиду того что она не была заложена как необходимое условие на начальном этапе. В компаниях постоянно имеют место события информационной безопасности в результате проблем в WEB-приложениях, часто это приводит к прямым финансовым потерям. Тем не менее многие разработчики, ставя во главу угла коммерческий успех, стремятся сокращать сроки проектов, часто в ущерб безопасности. Например, для упрощения своей работы они могут использовать методы и конструкции, не совместимые с принципами разработки безопасного кода.

В данной статье будут рассмотрены основные способы обеспечения безопасности WEB-приложения, которыми разработчики должны пользоваться в процессе разработки.

В ноябре 2017 года специалисты, входящие в состав проекта OWASP (Open Web Application Security Project) [1] выпустили обновленную версию рейтинга уязвимостей, которые являются наиболее критичными для безопасности WEB-приложений. Данный рейтинг получил название OWASP Top 10 2017. Этот проект ссылается на множество стандартов безопасности и является наиболее признанной методологией оценки уязвимости сайтов. Согласно этому рейтингу, наиболее опасными уязвимостями считаются:

1. Внедрение кода;
2. Некорректная аутентификация и управление сессией;
3. Утечка чувствительных данных;
4. Внедрение внешних XML-сущностей (XXE);
5. Нарушение контроля доступа;
6. Небезопасная конфигурация;
7. Межсайтовый скриптинг;
8. Небезопасная десериализация;
9. Использование компонентов с известными уязвимостями;
10. Отсутствие журналирования и мониторинга.

Далее будут рассмотрены меры борьбы с некоторыми из этих уязвимостей.

Тщательная проверка всех форм на сайте и HTML разметки. HTML формы могут создавать иллюзию контроля над вводимыми данными, потому что эти формы самостоятельно ограничивают типы значений, которые вводятся со стороны клиента. Однако это абсолютно не гарантирует безопасность. Пользователь WEB-приложения может легко изменить разметку страницы для отправки данных или использовать специальные утилиты (например, *curl)*. Для минимизации риска ввода потенциально опасных данных необходимо осуществлять проверку ввода. Необработанные данные, вводимые со стороны клиента, могут привести к неожиданным результатам, таким как нарушение бизнес логики, срабатыванию ошибок или даже позволить взять приложение под контроль злоумышленника. Если пользователь введет в форму код, то сервер может принять его за исполнимый. К таковым относятся запросы к базе данных или код на языке программирования JavaScript. Проверка гарантирует, что пользователь ввел именно те значения, которые WEB-приложение от него ожидает увидеть, например, определенного типа или из какого-либо диапазона. К примеру, разработчики делают проверку на то, что введенные пользователем значения не равны нулю, или что они относятся к целочисленному типу данных. Проверка вводимых данных более эффективна для тех данных, которые можно ограничить определенным набором значений. Например, нет смысла пользователю переводить отрицательное количество денег или добавлять несколько тысяч продуктов в корзину на сайте. Практика ограничения вводимых данных по определенным и допустимым на стороне сервера типам называется *whitelistening* (с англ. «добавление в белый список») [2]. Функционал проверки вводимых данных по умолчанию встроен в современных языках программирования и Фреймворках (таблица 1). Также существуют специальные библиотеки для тех языков программирования, где этот функционал не поддерживается.

Таблица 1

Фреймворки, содержащие встроенную проверку вводимых данных.

Фреймворк (язык программирования)	Название встроенного функционала
Java	Hibernate (Bean Validation)
Spring	Controller, Validator interface
Ruby on Rails	Active Record Validators
ASP.NET	Validation (BaseValidator)
Play	Validator
Generic JavaScript	Xss-filters
NodeJS	Validator-js

Разработчики WEB-приложений должны думать не только о том, какую информацию вводит пользователь, но и о том в каком виде клиент получает данные. Современные WEB-приложения обычно содержат HTML разметку, CSS стили, JavaScript для логики приложения и сам контент [3]. HTML являет-

ся не строгим форматом – браузеры отображают элементы документа даже если есть ошибки в синтаксисе страницы, например отсутствие закрывающих тегов. Это порождает разного рода уязвимости. У злоумышленника в таком случае появляется возможность встраивания собственного кода. Большинство современных WEB-фреймворков имеют механизмы безопасного хранения содержимого и экранирования зарезервированных символов, однако желательно не допускать ошибок при написании программного кода во избежание возникновения сбоев в работе WEB-приложения.

Защита данных при их передаче. При использовании обычного HTTP-соединения пользователи подвергаются многим рискам из-за того, что данные передаются в виде простого текста. Злоумышленник может перехватить сетевой трафик, находясь как бы между браузером и сервером, способен подслушивать и даже изменять данные. Нет ограничений в том, на что способен злоумышленник в такой ситуации – украсть сведения о пользовательском сеансе, завладеть личной информацией, встроить вредоносный код, который будет исполнен браузером на сайте, или изменить данные, отправляемые пользователем на сервер.

К сожалению, разработчики не могут контролировать то, какую сеть пользователи будут использовать. Они могут выходить в интернет через сети, в которых любой может легко наблюдать за трафиком, например, открытая беспроводная сеть в кафе или аэропорту. Многие пользователи даже не подозревают, что такие открытые сети могут быть специально настроены злоумышленниками в общественных местах. Также стоит отметить, что интернет-провайдеры могут внедрять на сайт рекламу, а во многих странах правительство регулярно следит за своими гражданами в интернете. К счастью, от многих из этих рисков можно защититься с помощью HTTPS.

Первоначально HTTPS использовался главным образом для защиты конфиденциального вебтрафика (например, финансовые транзакции), но теперь рекомендуется использовать данный протокол на многих сайтах, которыми мы пользуемся каждый день. К таким можно отнести социальные сети, поисковые системы и почтовые сервисы. Данные в протоколе HTTPS передаются поверх криптографических протоколов SSL (англ. Secure Sockets Layer – Уровень защищенных сокетов) или TLS (англ. Transports Layer Security – Протокол защиты транспортного уровня). При правильной настройке он обеспечивает защиту от подслушивания и подмены трафика, то есть сохраняет конфиденциальность и целостность данных.

С учетом многих рисков, с которыми мы сталкиваемся, все более разумно рассматривать весь сетевой трафик как конфиденциальный и шифровать его. При работе с веб-трафиком это делается с использованием HTTPS. Многие разработчики браузеров уже стали помечать сайты, на которых не используется HTTPS, как ненадежные.

Безопасное хранение паролей в базе данных. При разработке приложений нужно думать о том, как защитить данные своих пользователей. Небезопасное хранение паролей создает риск того, что человек, который имеет доступ к базе данных, может узнать пароль любого пользователя системы. Даже если у пользователя нет никакой важной информации на сайте, это не значит, что можно пренебречь безопасностью. К сожалению, многие пользователи не создают разные пароли для разных сайтов, а используют тот, к которому они привыкли на всех сервисах. То есть в таком случае злоумышленник зная только логин и пароль, может получить практически любую информацию о человеке.

Для того чтобы по максимуму обеспечить безопасность нужно, во-первых, не хранить пароли в базе данных в чистом виде, а хранить его хэш. Применяя хеширующий алгоритм к пользовательским паролям перед сохранением их в своей базе данных, вы делаете невозможным разгадывание оригинального пароля для атакующего вашу базу данных, в то же время сохраняя возможность сравнения полученного хеша с оригинальным паролем. Необходимо также добавлять к хэшу так называемую «Криптографическую соль». Криптографическая соль представляет собой данные, которые применяются в процессе хеширования для предотвращения возможности разгадать оригинальный ввод с помощью поиска результата хеширования в списке заранее вычисленных пар ввод-хеш, известном также как "радужная" таблица [4]. Более простыми словами, соль - это кусочек дополнительных данных, которые делают ваши хеши намного более устойчивыми к взлому. Существует много онлайн-сервисов, предоставляющих обширные списки заранее вычисленных хешей вместе с их оригинальным вводом. Использование соли делает поиск результирующего хеша в таком списке маловероятным или даже невозможным.

Успешные атаки злоумышленников на WEB-приложение могут нанести прямой финансовый и репутационный ущерб, а также использоваться в качестве плацдарма для вторжения, например, в корпоративную сеть предприятия. Именно поэтому вопросы защиты WEB-приложения должны быть

обязательно отражены в общей стратегии обеспечения информационной безопасности. Кроме этого, средства защиты приложения, некоторые из которых были рассмотрены в данной статье, должны обязательно интегрироваться с другими системами защиты, применяемыми в корпоративной сети. Такой подход позволит обеспечить более высокий уровень информационной безопасности в целом.

Действительно эффективным средством обеспечения безопасности корпоративных WEB-ресурсов остается периодическая проверка со стороны. Проводя аудит архитектуры, процессов, настроек и кода приложения, а также осуществляя моделирование действий потенциальных злоумышленников, можно получить картину, наиболее четко отражающую текущие проблемы в WEB-приложении. Кроме того, это позволяет определить приоритетные направления деятельности разработчиков, подразделений ИТ и ИБ в части защиты корпоративных ресурсов.

Список литературы:

1. OWASP Top 10 2017, The Ten Most Critical Web Application Security Risks [Электронный ресурс]. – Режим доступа: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf
2. Пьюривал С. Основы разработки веб-приложений / С. Пьюривал – СПб.: Питер, 2015, – 272с.
3. Елисеев Н.А., Федоров С.А., Антонов О.Д. ОБЗОР УГРОЗ БЕЗОПАСНОСТИ WEB-ПРИЛОЖЕНИЙ // Вопросы технических наук в свете современных исследований: сб. ст. по матер. V-VI междунар. науч.-практ. конф. № 1(4). – Новосибирск: СибАК, 2018. – С. 18-23
4. PHP: Хэширование паролей // Официальная документация языка программирования PHP [Электронный ресурс]. Режим доступа: http://php.net/manual/ru/faq.passwords.php

АЛГОРИТМЫ ОБУЧЕНИЯ НЕЙРОННОЙ СЕТИ ДЛЯ ИДЕНТИФИКАЦИИ ПОТЕНЦИАЛЬНЫХ ПОКУПАТЕЛЕЙ

А.В. Законов, студент
Томский государственный университет систем управления и радиоэлектроники (ТУСУР)
634050, г. Томск, пр. Ленина 40, тел. (3822) 51-05-30
E-mail: anton155578@mail.ru

Статья посвящена исследованию алгоритмов обучения нейронной сети для идентификации потенциальных покупателей. Для решения задачи рассмотрены два алгоритма: стохастический и дельта-метод.

Важным фактором успешной деятельности компании является выявление целевой аудитории. Определение характеристик клиентов может быть выполнено путем использования статистики о прошлых продажах: пол, возраст, семейное положение и т.д. Данное исследование посвящено выявлению в сообществах потенциальных покупателей на основе статистических данных о покупках за предыдущие периоды. Информация о продажах и характеристиках покупателей (пол, возраст) была собрана из сообщества по вейпингу.

Для решения задачи классификации (является ли человек потенциальным клиентом) была разработана нейронная сеть, представленная на рис.1 (a – весовые коэффициенты; x2,1, x2,2, x3,1, x3,2 – нейроны сети; x1,1, x1,2 - входные данные о поле и возрасте человека). Нейроны x2.1, x2.2 имеют линейную функцию активации, для активации нейронов x3,1 и x3,2 используется функция Хэвисайда. Результирующее значение формируется на основе правила: если выходы нейронов x3,1 и x3,2 равны 0 либо 1, то результат будет положительным, т.е. модель отнесет человека к потенциальному покупателю. Во всех других случаях (если выходы x3,1 и x3,2 равны (1,0) или (0,1)) результат будет отрицательным.

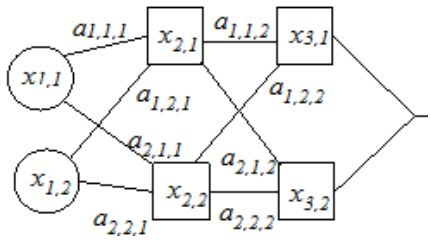

Рис.1. Вид сети в форме графа