Автор:

Коровин Кирилл Сергеевич

студент

Московский институт электроники и математики

им. А.Н. Тихонова ФГАОУ ВО «Национальный исследовательский

университет «Высшая школа экономики»

г. Москва

DOI 10.21661/r-462946

РАЗРАБОТКА ОНЛАЙН-АНАЛИЗАТОРА ВЕБ-САЙТОВ НА НАЛИЧИЕ СЕТЕВЫХ УЯЗВИМОСТЕЙ

Аннотация*: объектом разработки в данной статье является онлайн-сервис, выполняющий комплексное сканирование web-сайтов на сетевые уязвимости. Целью данной работы является предоставление инструмента, основанного на технологиях с открытым исходным кодом, способного снять часть задач по ручному тестированию, который может стать достойной альтернативой коммерческим онлайн-сканерам. В результате сервис был создан и вся заявленная функциональность реализована. Автор рекомендует использовать этот продукт пока только для своего пользования, но все дальнейшие силы пойдут на подготовку для его выхода в публичный доступ.*

Ключевые слова*: информационная безопасность, автоматизирование тестирования, уязвимость, web-приложения, онлайн-сервисы, CSRF, XSS, SQL инъекции, Python, Django, Nginx, MySQL.*

Интернет является очень важным инструментом для решения колоссального количества задач повседневной жизни что привело в итоге к тому, что очень большие объемы данных хранятся в интернете, в том числе, персональные и конфиденциальные данные пользователей. Как сказал Марсель Низамутдинов в своей книге «Тактики защиты и нападения на web-приложения» – «Интернет – это враждебная среда». Для работы в такой среде нужны свои методы защиты и

персональные либо конфиденциальные данные в интернете должны быть защищены должным образом. К сожалению, далеко не каждая компания может позволить себе иметь собственный отдел безопасности, и, к еще большему сожалению, далеко не все разработчики имеют даже базовые представления о том, как спроектировать безопасную систему. Так как порог вхождения в сферу информационной безопасности достаточно высок, то рядовому разработчику приходится выбирать чему посвятить свою время – усовершенствованию знаний в своей области разработки или углублению в защиту и безопасность систем. Существует большое количество сторонних инструментов, позволяющих провести тестирование на безопасность, но для того чтобы их освоить так же нужно время, так как большинство их них имеют не самый интуитивный интерфейс. Так же нельзя не обратить внимание на то, что в данный момент времени имеет место тенденция стремления к облачным технологиям, что позволяет пользователям сэкономить время на установке того или иного продукта, доверяя все производителю. В связи с вышеобозначенной проблемой, многие компании начали предоставлять свои продукты, которые проводят тестирование безопасности узлов посредством облачных сервисов.

В процессе изучения этого вопроса было найдено несколько продуктов, рассчитанных на разную аудиторию, но большинство из них были ориентированы на бизнес и имеют достаточно большую цену для рядовых пользователей или маленьких стартапов. Среди них Acunetix Online, Cloud Penetrator, Qualys и Secapps.

Проводя тестирование на уязвимость, нужно быть уверенным в используемых методах и инструментах, поэтому мной была предпринята попытка поиска существующих продуктов, основанных на open-source решениях, так как их исходный код можно изучить и удостовериться в безопасности их использования. К сожалению, продукты, упомянутые выше либо полагаются на собственные разработки, либо скрывают информацию о технологиях, лежащих в основе продукта.

В связи со всем вышеперечисленным, было принято решение разработать свой комплексный сканер либо основываясь на open-source решениях, либо с помощью своих разработок, чей исходный код позже также будет опубликован. Целью данной работы является предоставление полностью открытой альтернативы с проприетарным крупным продуктам, для обеспечения разработчиков инструментами, которые помогут им проверить свои web-сайты на уязвимости и сэкономить время.

Был поставлен ряд задач, которые нужно решить в процессе разработки. Для того, чтобы выглядеть выгодно на фоне конкурентов, интерфейс приложения должен быть максимально интуитивным и отзывчивым, поэтому было решено сделать приложение, которое располагалось бы на одной странице (ввиду его небольшого разнообразия видимой функциональности). Так как на серверной стороне работает большое количество скриптов и процессов, которые не всегда являются ресурсоёмкими, то была поставлена задача использовать технологии, которые впоследствии при росте пользователей могут масштабироваться с минимальными затратами. И последней задачей, исходя из цели, было использование сторонних инструментов только с открытым исходным кодом.

Для решения первой задачи, то есть, реализации принципа Single Page Application и функциональности приложения в режиме реального времени (без перезагрузки страницы) были использованы Javascript, посредством которого реализован интерфейс, работающий с web-сокетами) и jQuery, а также некоторые функции собственной jQuery библиотеки фреймворка Bootstrap. С помощью этого фреймворка можно достигнуть простых эффектов, как выпадающие списки или модальные окна, используя только атрибуты HTML объектов. Интерфейс web-сокетов в свою очередь может принимать сообщения из потока, преобразовывать их в объекты и передавать jQuery, который, в свою очередь, в реальном времени отрисует их на странице без перезагрузки страницы. Это добавляет интерфейсу плавности и интерактивности.

К реализации следующей задачи подход был более ответственным, и были взвешены все варианты, которые были доступны. Поставленная задача гласила о

том, что все технологии на серверной стороне должны быть максимально легко масштабируемы, и исходя из этого был сделан выбор. При выборе web-сервера между Nginx и Apache был выбран первый ввиду его большой производительности при малых объемах ресурсов. Также мы используем Django, и поэтому нам не нужен интерпретатор, встроенный в Web сервер, а лишь обработчик запросов, который будет отправлять их web-фреймворку. Из-за принципа своей работы (event-driven loop), этот web-сервер максимально просто и широко масштабируется в условиях ограниченного количества ресурсов.

Для управления базами данных была выбрана система MySQL, которая может масштабироваться как вертикально, так и горизонтально. Существует несколько методик горизонтального масштабирования MySQL, в зависимости от природы проблемы. При нагрузках, вызванных большим количеством запросов, используется репликация Master-Slave, позволяющая легко взаимозаменять вышедшие из строя серверы баз данных. При нагрузках, вызванных большим количеством данных, используется «шардинг», то есть разбиение таблицы на несколько таблиц, хранящихся на разных серверах.

Само приложение было решено написать на Django, и одним из факторов является то, что в этом фреймворке удобно реализован интерфейс управления соединениями с базами данных, что позволит при необходимости переключать их в случае репликации или шардинга.

Для решения последней задачи я решил использовать проверенные open-source сканеры, рекомендованные к использованию OWASP (Online Web Application Security Project, некоммерческая организация изучающая безопасность web-приложений). Поэтому, я использовал такие сканеры, как nmap, nikto, cmsmap, xsser, и sqlmap. Для тестирования CSRF уязвимостей такой сканер найден не был, поэтому было принято решение написать свой.

В качестве результата этой работы предполагается работающий сайт с формой для создания задач на сканирование и позволяющий просматривать прогресс

в реальном времени и результаты созданных задач. С исходным кодом результата можно ознакомиться по следующей ссылке: *https://github.com/skvoter/OVWAScanner/*

Список литературы

1. Низамутдинов М.Ф. Тактика защиты и нападения на web-приложения. СПб.: БХВ-Петербург, 2005.

2. Kali Linux Tools Listing [Электронный ресурс]. – Режим доступа: http://tools.kali.org/tools-listing (дата обращения: 02.05.2017).

3. Single page apps in depth [Электронный ресурс]. – Режим доступа: http://singlepageappbook.com/goal.html (дата обращения: 02.05.2017).

4. Репликация данных [Электронный ресурс]. – Режим доступа: https://ruhighload.com/post/Репликация+данных#masterslave (дата обращения: 03.05.2017).

5. Горизонтальный шардинг [Электронный ресурс]. – Режим доступа: https://ruhighload.com/post/Горизонтальный+шардинг (дата обращения: 03.05.2017).

6. Open Web Application Security Project (OWASP) [Электронный ресурс]. – Режим доступа: https://www.owasp.org (дата обращения: 29.04.2017).

7. Apache vs Nginx: практический взгляд [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/267721/ (дата обращения 06.05.2017).

8. Django Official Documentation [Электронный ресурс]. – Режим доступа: https://docs.djangoproject.com/en/1.10/ (дата обращения: 20.04.2017).

9. Django Channels [Электронный ресурс]. – Режим доступа: https://channels.readthedocs.io/en/stable/ (дата обращения: 21.04.2017).

10. XSSer: Cross Site «Scripter» [Электронный ресурс]. – Режим доступа: https://xsser.03c8.net/ (дата обращения: 23.04.2017).

11. Sqlmap – Automatic SQL injection and database takeover tool [Электронный ресурс]. – Режим доступа: http://sqlmap.org/ (дата обращения: 23.04.2017).

12. Nikto2 [Электронный ресурс]. – Режим доступа: https://cirt.net/Nikto2 (дата обращения: 23.04.2017).

13. JavaScript HTML DOM [Электронный ресурс]. – Режим доступа: https://www.w3schools.com/js/js_htmldom.asp (дата обращения: 10.05.2017).

14. jQuery API Documentation [Электронный ресурс]. – Режим доступа: https://api.jquery.com/ (дата обращения: 11.05.2017).