УДК 004

Алекперов З.А.,
магистр 2 курс,
факультет «программная инженерия и компьютерная техника» Национальный Исследовательский Университет Информационных Технологий Механики и Оптики
Россия, г. Санкт-Петербург

АНАЛИЗ УГРОЗ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ

Аннотация: *Статья посвящена анализу существующих угроз и методами борьбы с ними. В статье рассматривается существующие проблемы безопасности, топ 10 угроз и методы решения этих проблем. В статье также рассмотрена также статистика атак на веб-приложения.*

Ключевые слова: *Безопасность, угрозы, веб-приложения, атака, OWASP.*

Annotation*: This article is devoted to an analysis of existing threats and methods of dealing with them. This article discusses the existing security problems. The article has been reviewed by the statistics of attacks on web applications.*

Key words: *Security, threats, web applications, attack, OWASP.*

1. ОСНОВНАЯ ЧАСТЬ

1.1 Анализ угроз и методов борьбы с ними

Веб-приложения — это клиент-серверные приложения, основанные на протоколе HTTP [10], которые обычно управляют данными содержимого HTML, JavaScript, JSON и XML.

При разработке веб-приложений разработчики больше внимания уделяют обычно на обеспечение требуемой функциональности. А вопросам безопасности и качества программного кода уделяется недостаточно внимания. В результате подавляющее большинство веб-приложений содержит уязвимости различной степени критичности.

1.1.1 Угрозы

Open Web Application Security Project — это открытый проект обеспечения безопасности веб-приложений. На документ OWASP Top 10 часто ссылаются при упоминании наиболее распространенных недостатков безопасности веб-приложений. В документе описываются 10 наиболее распространенных недостатков, основанных на данных, собранных от сотен организаций и более 50 000 приложений и API, и он классифицирует их на основе их оценок распространенности и консенсуса в отношении возможности использования, обнаруживаемости и воздействия. По мнению OWASP, на эти уязвимости стоит обратить самое пристальное внимание как государственным, так и коммерческим организациям, желающим обезопасить себя и своих клиентов от злоумышленников. Все указанные уязвимости достаточно широко распространены, а использовать их под силу даже малоквалифицированным злоумышленникам, поскольку соответствующие средства взлома легко найти в сети Интернет.

Ниже представлен список десяти наиболее опасных и распространённых уязвимостей по результатам исследования OWASP [18]:

- A1 Внедрение кода

Инъекционные недостатки как SQL, NoSQL, OS и LDAP, возникают, когда ненадежные данные отправляются интерпретатору как часть команды или запроса. Враждебные данные атакующего могут обмануть интерпретатора в выполнение непреднамеренных команд или доступ к данным без надлежащей авторизации[16].

- A2 Некорректная аутентификация и управление сессией

Функции приложения, связанные с аутентификацией и управлением сеансами, часто основываются неправильно, что разрешает атакующим нарушать пароли, ключи или токены сеанса, или использовать другие недостатки реализации, чтобы временно или постоянно использовать идентификаторы других пользователей.

- A3 Межсайтовый скриптинг

Многие веб-приложения и API не защищают конфиденциальные данные, такие как финансовые, медицинские и PII. Атакующие могут красть или модифицировать такие слабо защищенные данные для совершения мошенничества с кредитными картами, кражи личных данных или других преступлений. Чувствительные данные могут быть скомпрометированы без дополнительной защиты, например, шифрование в состоянии покоя или в пути, и требует особых мер предосторожности при обмене с браузером.

- A4 Нарушение контроля доступа

Многие старые или слабо сконфигурированные XML-процессоры оценивают ссылки на внешние сущности в документах XML. Внешние объекты могут быть использованы для раскрытия внутренних файлов с использованием обработчика URI файла, внутренних общих файлов, внутреннего сканирования портов, удаленного выполнения кода и атак типа отказ в обслуживании.

- A5 Небезопасная конфигурация

Ограничения на то, что разрешено пользователям, прошедшим проверку подлинности, часто не выполняются должным образом. Атакующие могут использовать эти недостатки для доступа к несанкционированным функциям и/или данным, таким как доступ к учетным записям других пользователей, просмотр чувствительных файлов, изменение данных других пользователей, изменение прав доступа и т.д.

- A6 Утечка чувствительных данных

Наиболее часто встречающаяся проблема – это неправильная конфигурация системы. Обычно это результат небезопасных конфигураций по умолчанию, неполных или специальных конфигураций, открытого облачного хранилища, неправильно сконфигурированных заголовков HTTP и подробных сообщений об ошибках, содержащих конфиденциальную информацию. Мало того, что все операционные системы, фреймворки, библиотеки и приложения должны быть надежно настроены, но они должны быть исправлены / обновлены своевременно.

- A7 Недостаточная защита от атак (NEW)

Ошибки XSS возникают, когда приложение включает не доверенные данные на новой веб-странице без правильной проверки или экранирования, или обновляет существующую веб-страницу с предоставленными пользователем данными с помощью API-интерфейса браузера, который может создавать HTML или JavaScript. XSS позволяет атакующим выполнять сценарии в браузере жертвы, которые могут захватывать пользовательские сеансы, деактивировать веб-сайты или перенаправлять пользователя на вредоносные сайты.

- A8 Подделка межсайтовых запросов

Небезопасная десериализация часто приводит к удаленному выполнению кода. Даже если недостатки десериализации не приводят к удаленному выполнению кода, они могут использоваться для выполнения атак, включая атаки повторного использования, атаки на атаки и атаки на эскалацию привилегий.

- A9 Использование компонентов с известными уязвимостями

Компоненты как библиотеки, фреймворки и другие программные модули, работают с теми же привилегиями, как и приложение. Если уязвимый компонент используется, такая атака может облегчить серьезную потерю данных или захват серверов. Приложения и API, использующие компоненты с известными уязвимостями, могут подорвать защиту приложений и активировать различные атаки и удары.

- A10 Недостаточное журналирование и мониторинг

Недостаточная регистрация и мониторинг в сочетании с отсутствующей или неэффективной интеграцией с реагированием на инциденты позволяют атакующим продолжать атаковать системы, поддерживать постоянство, склоняться к большему количеству систем и подделывать, извлекать или удалять данные. Многие исследования с нарушением показывают, что время обнаружения нарушения составляет более 200 дней, обычно обнаруживаемых внешними сторонами, а не внутренними процессами или мониторингом.

1.1.2 Виды атак

Существует множество различных атак на веб-приложения, и с каждым годом их становится всё больше и больше. В данной главе рассмотрены некоторые виды сетевых атак. Проведено детальное рассмотрение каждой из атак и описаны способы защиты.

SQL-инъекция — это атака, направленная на веб-приложение, в ходе которой конструируется SQL-выражение из пользовательского ввода путем простой конкатенации (например, $query="SELECT * FROM users WHERE id=".$_REQUEST["id"]). В случае успеха атакующий может изменить логику выполнения SQL-запроса так, как это ему нужно. Чаще всего он выполняет простой fingerprinting СУБД, а также извлекает таблицы с наиболее «интересными» именами (например, «users»). После этого, в зависимости от привилегий, с которыми запущено уязвимое приложение, он может обратиться к защищенным частям бэк-энда веб-приложения (например, прочитать файлы на стороне хоста или выполнить произвольные команды).

Есть пять основных классов SQL-инъекций, и все их поддерживает sqlmap:

- UNION query SQL injection. Классический вариант внедрения SQL-кода, когда в уязвимый параметр передается выражение, начинающееся с «UNION ALL SELECT». Эта техника работает, когда веб-приложения напрямую возвращают результат вывода команды SELECT на страницу: с использованием цикла for или похожим способом, так что каждая запись полученной из БД выборки последовательно выводится на страницу. Sqlmap может также эксплуатировать ситуацию, когда возвращается только первая запись из выборки (Partial UNION query SQL injection).
- Error-based SQL injection. В случае этой атаки сканер заменяет или добавляет в уязвимый параметр синтаксически неправильное выражение, после чего парсит HTTP-ответ (заголовки и тело) в поиске ошибок DBMS, в которых содержалась бы заранее известная инъецированная последовательность символов и где-то «рядом» вывод на интересующий подзапрос. Эта техника

работает только тогда, когда веб-приложение по каким-то причинам (чаще всего в целях отладки) раскрывает ошибки DBMS.

- Stacked queries SQL injection. Сканер проверяет, поддерживает ли веб-приложение последовательные запросы, и, если они выполняются, добавляет в уязвимый параметр HTTP-запроса точку с запятой (;) и следом внедряемый SQL-запрос. Этот прием в основном используется для внедрения SQL-команд, отличных от SELECT, например, для манипуляции данными (с помощью INSERT или DELETE). Примечательно, что техника потенциально может привести к возможности чтения/записи из файловой системы, а также выполнению команд в ОС. Правда, в зависимости от используемой в качестве бэк-энда системы управления базами данных, а также пользовательских привилегий.
- Boolean-based blind SQL injection. Реализация так называемой слепой инъекции: данные из БД в «чистом» виде уязвимым веб-приложением нигде не возвращаются. Прием также называется дедуктивным. Sqlmap добавляет в уязвимый параметр HTTP-запроса синтаксически правильно составленное выражение, содержащее подзапрос SELECT (или любую другую команду для получения выборки из базы данных). Для каждого полученного HTTP-ответа выполняется сравнение headers/body страницы с ответом на изначальный запрос — таким образом, утилита может символ за символом определить вывод внедренного SQL-выражения. В качестве альтернативы пользователь может предоставить строку или регулярное выражение для определения «true»-страниц (отсюда и название атаки). Алгоритм бинарного поиска, реализованный в sqlmap для выполнения этой техники, способен извлечь каждый символ вывода максимум семью HTTP-запросами. В том случае, когда вывод состоит не только из обычных символов, сканер подстраивает алгоритм для работы с более широким диапазоном символов (например, для unicode’a).
- Time-based blind SQL injection. Полностью слепая инъекция. Точно так же как и в предыдущем случае, сканер «играет» с уязвимым параметром. Но в этом случае добавляет подзапрос, который приводит к паузе работы DBMS на

определенное количество секунд (например, с помощью команд SLEEP () или BENCHMARK ()). Используя эту особенность, сканер может посимвольно извлечь данные из БД, сравнивая время ответа на оригинальный запрос и на запрос с внедренным кодом. Здесь также используется алгоритм двоичного поиска. Кроме того, применяется специальный метод для верификации данных, чтобы уменьшить вероятность неправильного извлечения символа из-за нестабильного соединения.

XSS (Cross-Site Scripting — «межсайтовый скриптинг») — тип атаки на веб-системы, заключающийся во внедрении в выдаваемую веб-системой страницу вредоносного кода (который будет выполнен на компьютере пользователя при открытии им этой страницы) и взаимодействии этого кода с веб-сервером злоумышленника. Является разновидностью атаки «внедрение кода».

Специфика подобных атак заключается в том, что вредоносный код может использовать авторизацию пользователя в веб-системе для получения к ней расширенного доступа или для получения данных пользователя. Вредоносный код может быть вставлен в страницу как через уязвимость в веб-сервере, так и через уязвимость на компьютере пользователя [17].

Для термина используют сокращение «XSS», чтобы не было путаницы с каскадными таблицами стилей, использующими сокращение «CSS».

Cross-Site Request Forgery (CSRF) (Подделка межсайтовых запросов). Атака CSRF может происходить, когда злоумышленник находит воспроизводимую ссылку, которая выполняет определенное действие на сервере, когда жертва находится там.

Злоумышленник может встроить такую ссылку, как ссылки на изображения или ссылки на спам по электронной почте, и заставить жертву открыть ее, заставив выполнять определенные действия без согласия пользователя.

Пример: злоумышленник находит следующую ссылку, которая переводит средства учетной записи пользователя, когда пользователь правильно вошел в систему

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

Злоумышленник может встроить ссылку в вредоносные сайты или спам-сообщения и обманным путем заставить законного пользователя получить доступ к этим ссылкам, пока жертва остается в системе. Злонамеренное действие будет выполнено без согласия пользователя:

1.1.3 Статистика атак на веб-приложения

Positive Technologies — международная компания, специализирующаяся на разработке программного обеспечения в области информационной безопасности в проводимой исследовании, представил статистику атак на веб-приложения за IV квартал 2017 года. Исходные данные были получены в ходе пилотных проектов по внедрению межсетевого экрана уровня приложений PT Application Firewall, а также по итогам работы PT AF для защиты веб-приложений компании Positive Technologies. В отчете рассмотрены наиболее распространенные типы атак, цели атак, их источники, а также интенсивность и распределение во времени. Исследование атак позволяет оценить текущие тенденции в области безопасности веб-приложений, выявить актуальные угрозы и выделить факторы, на которые прежде всего следует обратить внимание при разработке веб-приложения и построении системы защиты. Для получения более достоверных результатов, автоматизированный поиск уязвимостей с помощью специализированного ПО для сканирования веб-приложений (например, Acunetix) был исключен из исходных данных. Приведенные в отчете примеры атак были проверены вручную на предмет ложных срабатываний и являются достоверными. Ниже приведены результаты исследования [5]:

Рисунок 1. Самые распространенные атаки

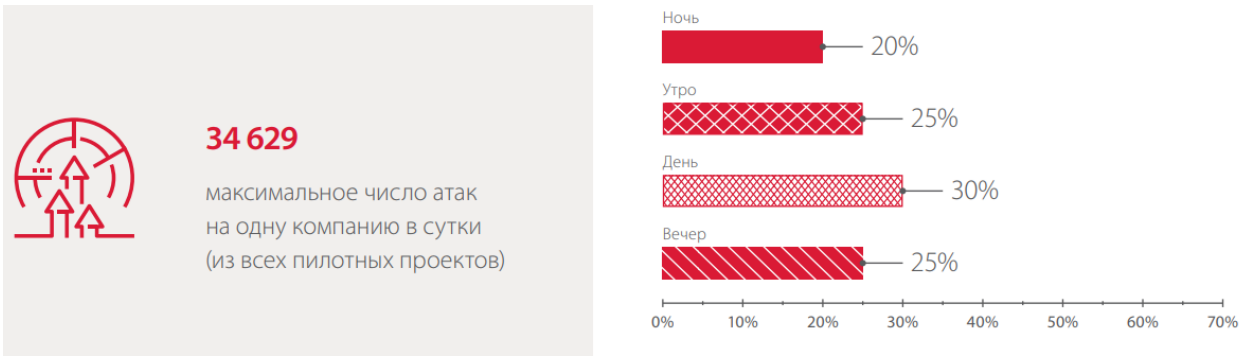

Рисунок 2. Распределение атак по времени суток (по местному времени исследуемых организаций)

Рисунок 3. Среднее количество атак в день на одну компанию

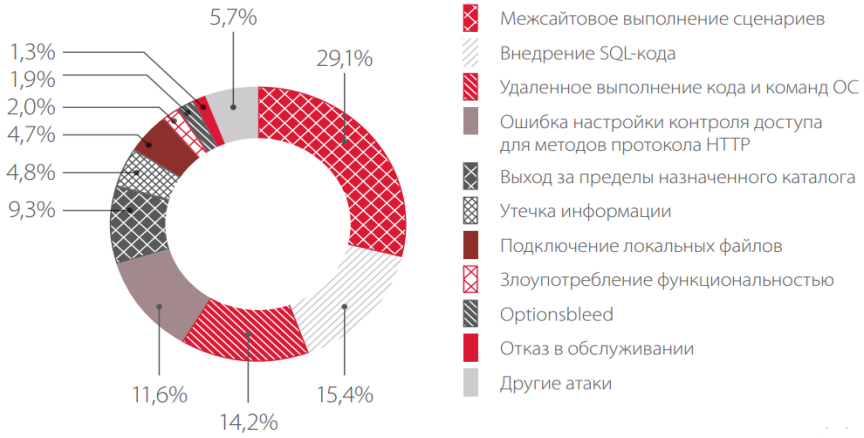

Рисунок 4. Типы атак на веб-приложения

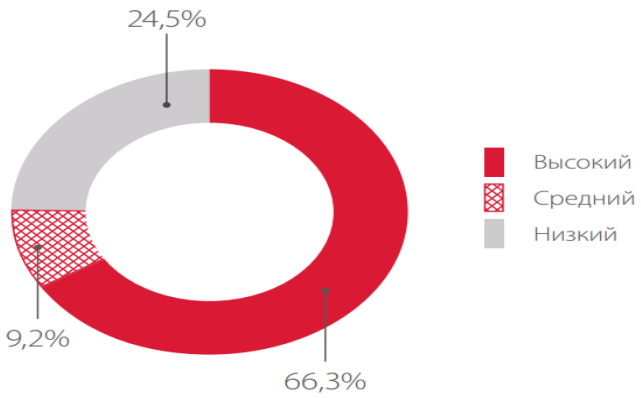

Рисунок 5. Распределение атак на веб-приложения по уровню риска

При анализе результатов пилотного проекта для одной IT-компании был обнаружен ряд атак, направленных на эксплуатацию недавно опубликованных уязвимостей в CMS платформе WordPress. Дальнейшее исследование показало, что данные атаки, вероятнее всего, производились при помощи ботнета из более чем 300 устройств. В течение суток было отправлено около 400 HTTP-запросов, при этом злоумышленники старались максимально скрыть свои действия, и с одного узла, входящего в ботнет, отправлялось не более двух запросов.

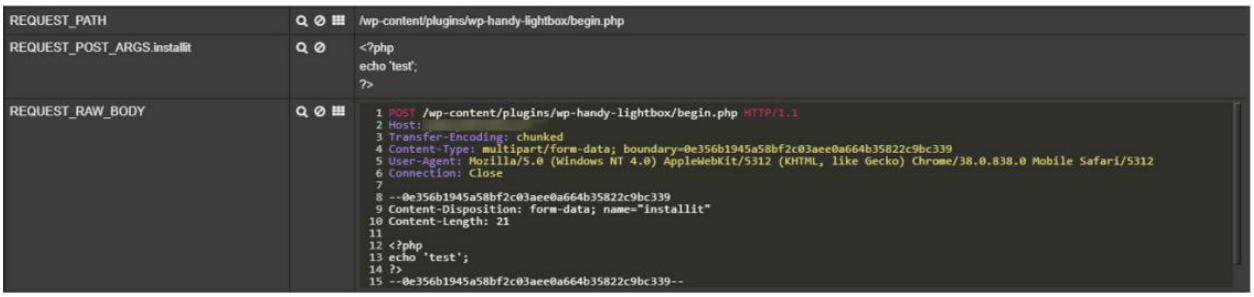

Рисунок 6. Запрос для эксплуатации уязвимости в CMS WordPress (интерфейс PT AF)

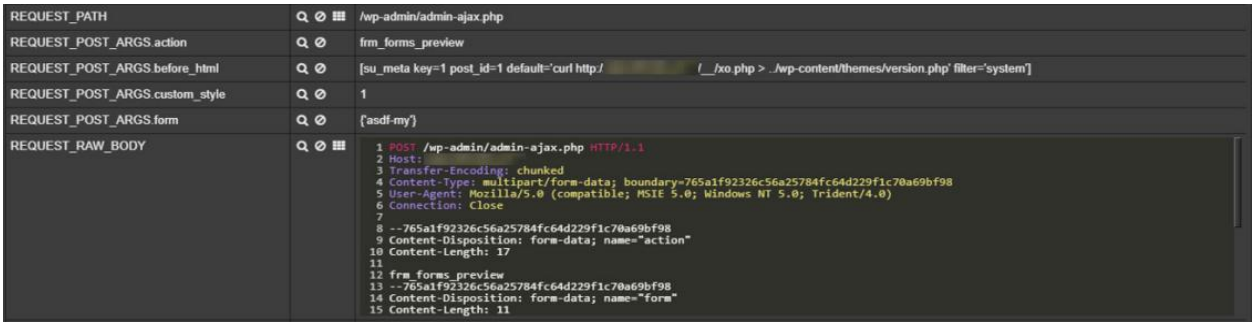

Рисунок 7. Запрос для эксплуатации уязвимости в CMS WordPress (интерфейс PT AF)

Выводы:

- абсолютно любое веб-приложение вне зависимости от функциональных особенностей может стать мишенью для злоумышленников;
- большинство атак направлено на доступ к чувствительной информации и на пользователей веб-приложений;
- у злоумышленников нет выходных, праздников, отпусков и фиксированного рабочего дня, атаки на веб-приложения производятся в любой день недели в любое время суток;
- после публикации информации о новой уязвимости злоумышленники в кратчайшие сроки разрабатывают эксплойты и начинают тестировать их на веб-приложениях;
- для автоматизации атак нарушители могут использовать не только общедоступные готовые эксплойты и утилиты, но и целые ботнеты [13].

1.1.4 Защита от атак

Существует различные способы защиты атак. Например, инструменты для анализа защищенности. Прежде чем искать уязвимости вручную желательно проверить приложение автоматизированными средствами. Они выполняют тесты на проникновение, пытаются его взломать, например, при помощи SQL-инъекции.

Ниже приведена подборка бесплатных инструментов.

Приложения и фреймворки

- OpenVAS сканирует узлы сети на наличие уязвимостей и позволяет управлять уязвимостями.
- OWASP Xenotix XSS Exploit Framework сканирует ресурс на возможность эксплуатации XSS-уязвимостей.
- Approof от Positive Technologies проверяет конфигурацию веб-приложения, сканирует на наличие уязвимых компонентов, незащищенных чувствительных данных и вредоносного кода.

Онлайн-сервисы

- SecurityHeaders.io проверяет на наличие и корректность заголовков ответа сервера, отвечающих за безопасность веб-приложения [6].
- Observatory by Mozilla сканирует ресурс на наличие проблем безопасности. Кроме своих результатов, при выборе соответствующей опции, собирает и добавляет к отчету аналитику со сторонних сервисов анализа защищённости [7].
- One button scan сканирует на наличие уязвимостей компоненты ресурса: DNS, HTTP-заголовки, SSL, чувствительные данные, используемые сервисы [8].
- CSP Evaluator проверяет правильность составления политики безопасности содержимого (CSP) и устойчивость к XSS [9].
- SSL Server Test выполняет анализ SSL-конфигурации веб-сервера [10].
- ASafaWeb проверяет на наличие распространённых уязвимостей конфигурации сайтов, написанных на ASP.NET [11].
- Snyk сканирует JavaScript, Ruby и Java-приложения на наличие уязвимостей и, при необходимости, исправляет проблемы безопасности. Интегрируется с GitHub репозиторием для проведения автоматической проверки и оповещает о найденных уязвимостях [12].

1.1.5 Существующие методы борьбы с уязвимостями веб-приложений

На сегодняшний день по исследованиям OWASP [15] наиболее эффективным способом выявления уязвимостей является экспертный анализ исходного кода (code review).

Обзор кода (Code Review) – один из методов анализа кода. Суть обзора кода заключается в совместном внимательном чтении всего кода приложения. В процессе обзора кода тестировщики дают советы в улучшении кода. Автор кода при этом не должен давать объяснений, как работает та или иная часть кода, так как считается что алгоритм кода должен быть понятен из кода или комментариев. При отсутствии этих правил код должен быть доработан.

Этот способ трудоемкий и требует высокой квалификации эксперта и не защищен от ошибок эксперта. Поэтому активно развиваются методы автоматического обнаружения уязвимостей веб–приложений [4].

Методы автоматического обнаружения уязвимостей приложений делится на две основные группы:

1) методы, анализирующие работу веб–приложения без обращения к исходным кодам веб–приложения;

a) Метод получения идентифицирующей информации о веб–приложении и выявлении его уязвимостей с помощью бюллетеней безопасности (security advisory).

b) Метод тестирования на проникновение.

Эта группа методов рассматривает веб–приложение с точки зрения потенциального злоумышленника.

2) методы, анализирующие исходные коды веб–приложения и конфигурационные настройки.

a) Метод статического анализа исходных кодов веб–приложения.

Статический анализ кода — анализ программного обеспечения, производимый без реального выполнения исследуемых программ. В большинстве случаев анализ производится над какой-либо версией исходного кода, хотя иногда анализу подвергается какой-нибудь вид объектного кода, например P-код или код на MSIL. Статический анализ можно рассматривать как автоматизированный процесс обзора кода.

b) Метод динамического анализа исходных кодов веб–приложения

Динамический анализ – это тестирование и оценка программы путем выполнения данных в режиме реального времени.

Динамический анализ можно подразделить на следующие пункты:

- подготовка исходных данных
- проведение тестового запуска программы и сбор необходимых параметров
- анализ полученных данных

Тестовый запуск исполнения программы возможно воспроизводить как на реальном, так и на виртуальном процессоре. Поэтому для анализа нужно из исходного кода получить исполняемый файл, то есть нельзя проанализировать код, содержащий ошибки компиляции или сборки.

Цель динамического анализа – найти ошибки безопасности в программе во время ее работы. В отличие от статического анализа, программа динамического анализа не имеет доступа к исходному коду и поэтому обнаруживает уязвимости, фактически выполняя атаки.

1.1.6 Брандмауэры веб-приложений (WAF)

Брандмауэры веб-приложений - это особый тип системы обнаружения и предотвращения вторжений, предназначенный для защиты веб-приложений [1]. Они способны фильтровать и отслеживать трафик на уровне приложений (уровень 7 OSI) и обнаруживать или блокировать атаки на уровне приложений, такие как SQL-инъекция или межсайтовый скриптинг. Они также имеют дело с распространенными неправильными настройками безопасности веб-сервера и уязвимостями, которые могут быть использованы злоумышленниками. Примером популярного WAF с открытым исходным кодом является ModSecurity

1.1.7 Метод тестирования на проникновение

Метод тестирования на проникновение (penetration testing) рассматривает веб–приложение с точки зрения внешнего пользователя, то есть потенциального злоумышленника. При этом считается, что злоумышленник обладает такими же возможностями, как и обычный пользователь, т. е. не имеет доступа к исходным кодам и доступа к серверу, где находится веб–приложение. Этот метод тестирует приложение посылая запросы, которые имитируют пользовательскую активность, включающую «корректные» запросы, соответствующие нормальным действиям пользователя и некорректные запросы, соответствующие действиям злоумышленника. При поиске уязвимостей в веб–приложении методом тестирования на проникновение возникают три основные задачи [3]:

1. Задача получения и анализа структуры веб–приложения;

Суть этой задачи состоит в том, чтобы построить полный список URL веб–приложения, методов доступа к ним и списков их параметров, выделить URL,

защищённые аутентификацией. Данная информация необходима для построения набора тестовых запросов к веб–приложению.

2. Задача построения набора тестовых HTTP-запросов на основе построенной структуры веб–приложения;

Суть этой задачи состоит в том, чтобы по исходным данным (список URI приложения, методы доступа к каждому URI и набор параметров к данному URI) подобрать запросы так, чтобы было обнаружено как можно больше уязвимостей [18]. Ниже представлен список способов построения таких запросов:

- ***Запрос ресурсов по базе имён***

Этот способ подразумевает, что существует некая база ресурсов, состоящая из имён файлов, которые потенциально могут встретиться в структуре веб–приложения. Наличие в ней того или иного ресурса свидетельствует об уязвимости, связанной с возможным доступом к этому ресурсу. Например, присутствуют имена известных уязвимых CGI-сценариев и имена конфигурационных файлов веб–серверов

- ***Генерация запросов по шаблону с типизированными параметрами***

В этом способе для каждого URI задаётся шаблон, параметры которого типизированы. Далее происходит автоматическая генерация запросов по заданному шаблону со случайным выбором значений конкретных параметров. Значения параметров могут задаваться регулярными выражениями, что позволяет эмулировать атаки XSS и SQL injection на параметрах запроса

- ***Анализ настроек каталогов Веб–приложения***

По иерархии веб–приложения проверяются настройки веб–сервера, которые могут выявить уязвимости веб–приложения, связанные с неправильным конфигурированием веб–приложения и веб–сервера. Сюда относятся проверка возможности автоматического построения индекса каталога, выполнения HTTP-методов PUT и DELETE, возможность обращения к ресурсам из областей аутентификации напрямую, возможность получения исходных кодов веб–приложения.

3. Задача прогона тестового набора с анализом ответов веб–приложения для выявления уязвимостей.

Эта задача ставит перед собой цель сделать правильный вывод о том, демонстрирует ли данный HTTP-запрос наличие уязвимости в веб–приложении или нет. Данная задача тесно связана с задачей построения тестового набора, а основная проблема состоит в определении критериев наличия уязвимости.

При использовании метода тестирования на проникновение могут использоваться принципы “черного”, “белого” и “серого ящика”.

Принцип «черного ящика» (black-box).

Black Box Testing - это метод тестирования программного обеспечения, при котором внутренняя структура / дизайн / реализация тестируемого элемента НЕ известны тестировщику.

Это полезно, когда необходимо оценить защищенность с позиций злоумышленника, обычно располагающего минимальными знаниями об исследуемой системе. Все исследования могут проходить как с предупреждением обслуживающего персонала о планируемых работах, так и без него. Во втором случае существует возможность оценить, за какое время после начала исследования персонал зафиксирует инцидент, а также какова адекватность предпринимаемых действий по минимизации его воздействия или предотвращения.

Принцип «серого ящика» (gray-box).

Тестирование серого ящика – это методология тестирования программного обеспечения, которая включает в себя сочетание тестирования белого ящика и черного ящика. Он использует простую технику тестирования черного ящика, а также подход для систем с целевым кодом, как в случае тестирования белого ящика. Тестировщикам, которые используют тестирование «серого ящика», необходима высокоуровневая прикладная документация для завершения тестов. Тестирование «серого ящика» направлено на поиск дефектов, основанных на неправильной структуре или использовании приложения.

Обычно исполнителю предоставляются следующие данные: структура каталогов приложения, данные для авторизованного подключения в пространстве Веб–приложения (например, имя пользователя, пароль и набор одноразовых паролей для проводки транзакций), исходный код некоторых файлов или функций и пр.

Принцип «белого ящика» (white-box).

White Box Testing - это метод тестирования программного обеспечения, при котором внутренняя структура / дизайн / реализация тестируемого элемента известны тестировщику.

Данный принцип подразумевает передачу исполнителю всего приложения с его последующим развертыванием на площадке консультанта, выполняющего работу по его анализу, либо организацию аналогичной копии приложения в собственной информационной системе с предоставлением исполнителю полного доступа к этому ресурсу. В данном случае имеется возможность отследить, каким образом приложение реагирует на любой передаваемый к нему запрос. Это наиболее продуктивный метод проведения анализа защищенности Веб–приложений, позволяющий выявить наибольшее число уязвимостей. Однако стоит заметить, что данный метод лишен возможности взглянуть на приложение с позиций атакующего.

ЗАКЛЮЧЕНИЕ

В данной научно-исследовательской работе проделан тщательный анализ литературных источников как зарубежных, так и отечественных авторов.

В работе был проведен анализ существующих угроз и средств безопасности веб-приложений.

На основании проведенного анализа можно сделать вывод, что дальнейшее развитие методов обнаружения уязвимостей веб–приложений пойдет по пути интеграции возможностей различных методов с тем, чтобы было возможно контролировать полноту обнаружения уязвимостей и обнаружение уязвимостей

разных классов с тем, чтобы по итогам автоматического анализа можно было бы дать гарантию отсутствия уязвимостей заданных классов.

СПИСОК ЛИТЕРАТУРЫ

1. Баранов А.П. Актуальные проблемы в сфере обеспечения информационной безопасности программного обеспечения, Вопрос кибербезопасности [Cybersecurity issues], 2015, No 1 (9), pp.2-5.
2. Будников Е.А., Борисова С.Н. УЯЗВИМОСТЬ Web-ПРИЛОЖЕНИЙ // Международный студенческий научный вестник. – 2015. – № 3-2.; [Электронный ресурс]. – Режим доступа: https://www.eduherald.ru/ru/article/view?id=12471 (дата обращения: 16.02.2018).
3. Калашников А.О., Ермилов Е.В., Чопоров О.Н., Разинкин К.А. Баранников Н.И. Атаки на информационно-технологическую инфраструктуру критически важных объектов: оценка и регулирование рисков. Под ред. чл.-корр. РАН Д.А Новикова. – Воронеж: Издательство «Научная книга», 2013. – 160 с.
4. Козлов Д. Д., Петухов А. А. "Методы обнаружения уязвимостей в web - приложениях" / Программные системы и инструменты: тематический сборник ф-та ВМиК МГУ им. Ломоносова N 7. П/р Л.Н. Королева. М: Издательский отдел ВМиК МГУ. Изд-во МАКС Пресс, 2006 г.
5. Королев О.Л. Безопасность веб-приложений / О.Л. Королев, М.А. Лукьянова // Сборник трудов Международной научно-практической конференции «Проблемы информационной безопасности». – Симферополь, 2016. – С. 166-168.
6. Онлайн сервис SecurityHeaders.io. [Электронный ресурс]. – Режим доступа: https://securityheaders.io (дата обращения: 19.02.2018).
7. Онлайн сервис – Observatory by Mozilla [Электронный ресурс]. – Режим доступа: https://observatory.mozilla.org (дата обращения: 19.02.2018).
8. Онлайн сервис – One button scan [Электронный ресурс]. – Режим доступа: https://sergeybelove.ru/one-button-scan (дата обращения: 19.02.2018).

9. Онлайн сервис – CSP Evaluator [Электронный ресурс]. – Режим доступа: https://csp-evaluator.withgoogle.com (дата обращения: 19.02.2018).
10. Онлайн сервис – SSL Server Test [Электронный ресурс]. – Режим доступа: https://www.ssllabs.com/ssltest/ (дата обращения: 19.02.2018).
11. Онлайн сервис – ASafaWeb [Электронный ресурс]. – Режим доступа: https://asafaweb.com (дата обращения: 19.02.2018).
12. Онлайн сервис – Snyk [Электронный ресурс]. – Режим доступа: https://snyk.io (дата обращения: 19.02.2018).
13. Статистика атак на веб – приложения IV квартал 2017 года. [Электронный ресурс]. – Режим доступа: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/WebApp-Vulnerabilities-2017-Q4-rus.pdf (дата обращения: 21.02.2018).
14. Уязвимость CSRF. Введение [Электронный ресурс]. – Режим доступа: http://intsystem.org/768/learn-about-csrf-intro/ (дата обращения: 24.11.2017).
15. Curphey M., Wiesman A., Van der Stock A., Stirbei R. A Guide to Building Secure Web Applications and Web Services. OWASP, 2005.
16. Fonseca J, Vieira M, Madeira H: Testing and Comparing Web vulnerability scanning tools for SQL injections and XSS attacks. In Proc. 2007 IEEE Symposium Pacific Rim Dependable Computing (PRDC 2007).
17. OWASP Top 10 Application Security Risks – 2017. [Электронный ресурс]. – Режим доступа: https://www.owasp.org/index.php/Top_10-2017_Top_10 (дата обращения: 04.03.2018).
18. Yao-Wen Huang, Shih-Kun Huang, Tsung-Po Lin, Chung-Hung Tsai. Web Application Security Assessment by Fault Injection and Behavior Monitoring. Proceedings of 12-th WWW Conference, 2003.