БЕЗОПАСНОСТЬ ЦИФРОВЫХ ТЕХНОЛОГИЙ. – 2023. – № 4 (111). – 64–81

МЕТОДЫ И СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ, ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК 004.056 DOI: 10.17212/2782-2230-2023-4-64-81

ФОРМИРОВАНИЕ МЕТОДИКИ БЕЗОПАСНОГО ПРОГРАММИРОВАНИЯ ДЛЯ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ*

А.Б. АРХИПОВА[1], Р.Е. ЛИСТАРОВ[2]

[1] *630073, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, доцент кафедры защиты информации. E-mail: arhipova@corp.nstu.ru*
[2] *630073, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, лаборант кафедры защиты информации. E-mail: kaf_zi@corp.nstu.ru*

В настоящее время киберфизические системы нередко становятся объектами целевых кибератак, заключающихся в том числе в эксплуатации уязвимостей программного обеспечения, функционирующего в таких системах. Уязвимое программное обеспечение может быть использовано для распространения вредоносных программных средств, кражи или разглашения конфиденциальных данных, воздействия на защищаемую информацию с нарушением установленных прав, приводящих к разрушению, уничтожению, искажению или сбою в работе автоматизированных систем, в которых данное программное обеспечение функционирует. Определение методики безопасной разработки веб-приложений является важнейшим этапом в процессе разработки системы комплексной защиты объекта информатизации. В статье предложены технологии по улучшению уровня защищенности, предоставляемого методикой безопасного программирования BSIMM, с помощью которых можно обеспечить должный уровень защиты для каждого веб-приложения в зависимости от чувствительности обрабатываемой информации.

Ключевые слова: информационная безопасность, защита информации, безопасное программное обеспечение, требования безопасности, жизненный цикл разработки, тестирование защищенности, угрозы безопасности

ВВЕДЕНИЕ

В области информационной безопасности одной из наиболее насущных проблем является обеспечение защиты веб-приложений. Взлом веб-приложений может причинить значительный ущерб системам, которые хранят конфиденциальные и критически важные данные. Статистика показывает, что

* Статья получена 07 ноября 2023 г.

в настоящее время более 80 % кибератак осуществляются через API-интерфейсы, что увеличивает риск для компаний и государственных учреждений [1].

Поэтому формирование методики безопасного программирования, которая поможет снизить риски утечки конфиденциальных данных в веб-приложениях и будет способствовать созданию тестирования для обучения персонала данной методике, является актуальной задачей.

1. КЛАССИФИКАЦИЯ ОСНОВНЫХ УЯЗВИМОСТЕЙ ВЕБ-ПРИЛОЖЕНИЙ

Основные уязвимости веб-приложений классифицируют согласно OWASP Top 10. Open Worldwide Application Security Project (OWASP) – это некоммерческая организация, работающая над повышением безопасности программного обеспечения. OWASP имеет более 250 отделений по всему миру, десятки тысяч участников, проводит отраслевые, образовательные и обучающие конференции [10].

OWASP Top 10 – это стандартный документ для разработчиков и специалистов по безопасности веб-приложений, обновляющийся каждые 3-4 года и представляющий список наиболее опасных уязвимостей для веб-приложений на данный момент.

Рассмотрим основные группы уязвимостей более подробно.

Нарушение контроля доступа

Контроль доступа обеспечивает соблюдение политики безопасности. Нарушения, как правило, приводят к несанкционированному раскрытию, изменению или уничтожению информации, а также выполнению бизнес-функций, выходящих за пределы прав пользователя.

Перечень основных уязвимостей нарушения контроля доступа представлен ниже.

- Нарушение принципа минимальных привилегий или принципа запрета по умолчанию, когда доступ должен быть предоставлен только для определенных возможностей, ролей или пользователей, но доступен для всех.
- Обход контроля доступа путем изменения URL (манипуляции параметрами или принудительного просмотра), внутреннего состояния приложения или HTML-страницы, а также с использованием инструмента атаки для изменения API-запросов.
- Разрешение просмотра или редактирования учетных записей другого человека путем предоставления ее уникального идентификатора (небезопасные прямые ссылки на объекты).

- Доступ к API без наличия контроля доступа для операций POST, PUT и DELETE.
- Повышение привилегий. Действие в качестве пользователя без авторизации или действие в качестве администратора при входе в систему как обычный пользователь.
- Манипулирование метаданными, такое как повторное использование или изменение токена управления доступом JSON Web Token (JWT), а также изменение cookie или скрытого поля с целью повышения привилегий или злоупотребление отменой JWT.
- Неправильная конфигурация CORS (Cross-Origin Resource Sharing) позволяет получить доступ к API из неподтвержденных / ненадежных источников.
- Принудительный просмотр страниц, требующих аутентификации, в качестве неаутентифицированного пользователя, или принудительный доступ к привилегированным страницам в качестве обычного пользователя.

Контроль доступа эффективен только в надежном серверном коде или в API без сервера, где злоумышленник не может изменить проверку контроля доступа или метаданные.

Криптографические сбои

Криптографические сбои часто приводят к утечке конфиденциальных данных или к нарушению целостности системы. Элементами противодействия криптографическим сбоям являются меры в виде определения требований к защите данных в процессе передачи и хранения, использования современных криптографических функций и схем, анализа эффективности конфигурации и настроек системы.

Инъекции

Приложение становится уязвимым для атак данного типа в следующих случаях:

- входные данные, предоставляемые пользователем, не проходят проверку, фильтрацию или очистку со стороны приложения;
- динамические запросы или непараметризованные вызовы без контекстно осознанной экранировки используются напрямую в интерпретаторе;
- поисковые данные используются в параметрах поиска объектно-реляционного отображения (ORM) для извлечения дополнительных конфиденциальных записей;
- поисковые данные непосредственно используются или конкатенируются. SQL-запрос или команда содержат структуру и вредоносные данные в динамических запросах, командах или хранимых процедурах.

Небезопасный дизайн

Небезопасный дизайн – широкая категория уязвимостей, несущих риск безопасности. Он отличается от небезопасной реализации и имеет свои причины. Отсутствие профилирования бизнес-риска является фактором, способствующим небезопасному дизайну. Поэтому интеграция принципов безопасного программирования на всех жизненных циклах этапов разработки является необходимой составляющей бизнес-процессов организации.

Безопасное программное обеспечение требует применения цикла безопасной разработки, некоторой формы безопасного дизайн-паттерна, методологии, библиотеки безопасных компонентов, инструментария и моделирования угроз. Так, например, следует следить за распределением уровней и слоев систем и сетей, потребление ресурсов для каждого пользователя или сервиса должно быть ограничено.

Неправильная конфигурация

Приложение становится уязвимым для атак данного типа в следующих случаях:

- при отсутствии соответствующей защиты в любой части стека приложения или неправильной настройки разрешения в облачных сервисах;
- включении или установке лишних средств и функций (порты, сервисы, страницы, учетные записи или привилегии) в рамках определенной задачи;
- использовании стандартных учетных записей с неизмененными паролями;
- наличии устаревшего или уязвимого программного обеспечения и других.

Уязвимые и устаревшие компоненты

Приложение становится уязвимым для атак данного типа в следующих случаях:

- если не известны версии всех используемых компонентов (клиентских и серверных);
- программное обеспечение уязвимо, не поддерживается или устарело (включая операционную систему, веб-сервер, сервер приложений, систему управления базами данных, приложения, API, среды выполнения и библиотеки);
- не исправляется или своевременно не обновляется базовая платформа, фреймворки и зависимости. Это часто происходит в средах, где обновления происходят ежемесячно или ежеквартально в рамках процедуры управления изменениями, что оставляет организации открытыми для известных уязвимостей в течение дней или месяцев;
- разработчики программного обеспечения не проверяют совместимость обновленных, улучшенных или исправленных библиотек.

Соответственно для устранения уязвимости «Уязвимые и устаревшие компоненты» должен существовать процесс управления обновлениями. Каждая организация должна обеспечить постоянный мониторинг и применение обновлений или изменений конфигурации на протяжении всего срока эксплуатации приложения или портфолио.

Ошибки идентификации и аутентификации

Подтверждение личности пользователя, аутентификация и управление сеансами являются критическими мерами для защиты от аутентификационных атак.

Могут существовать уязвимости аутентификации в следующих случаях:

- если приложение разрешает автоматизированные атаки, такие как наполнение учетных данных (credential stuffing), когда злоумышленник располагает списком действительных имен пользователей и паролей;
- разрешает атаки методом перебора (brute force) или другие автоматизированные атаки;
- разрешает использование паролей по умолчанию, слабых или широко известных, таких как «Password1» или «admin/admin»;
- использует слабые или неэффективные процессы восстановления учетных данных и восстановления забытого пароля, которые не обеспечивают должную безопасность;
- использует хранение паролей в открытом тексте, шифрованных или слабо хешированных данных;
- отсутствует или неэффективна многофакторная аутентификация;
- показывает идентификатор сеанса в URL;
- повторно использует идентификатор сеанса после успешной аутентификации;
- некорректно отменяет действие идентификатора сеанса.

В качестве противодействия данной уязвимости рекомендуется при возможности реализовать многофакторную аутентификацию для предотвращения автоматического наполнения учетных данных, атак методом перебора и повторного использования украденных учетных данных.

Нарушение целостности данных и программного обеспечения

Сбои программного обеспечения и целостности данных связаны с кодом и инфраструктурой, которые не обеспечивают защиту от нарушений целостности. Примером является ситуация, когда приложение полагается на плагины, библиотеки или модули из ненадежных источников, репозиториев и сетей доставки контента. Небезопасный процесс непрерывной поставки может привести к возможности несанкционированного доступа, внедрения вредоносного кода

или компрометации системы. Кроме того, многие приложения сейчас включают функцию автоматического обновления, при которой обновления загружаются без должной проверки целостности и применяются к ранее доверенному приложению. Злоумышленники могут потенциально загружать свои собственные обновления для распространения и выполнения на всех установках. Еще одним примером является ситуация, когда объекты / данные кодируются или сериализуются в структуру, которую злоумышленник может видеть и изменять, что делает такую систему уязвимой для небезопасной десериализации.

Журнал безопасности и сбои мониторинга

Без ведения журналов и мониторинга невозможно обнаружить нарушения. Недостаточный уровень журналирования, обнаружения, мониторинга и активного реагирования проявляется в следующих случаях:

- события, которые могут быть подвержены аудиту, такие как вход в систему, неудачные попытки входа и транзакции высокой ценности, не регистрируются;
- предупреждения и ошибки не создают или создают недостаточно информативные или понятные журнальные сообщения;
- тестирование на проникновение и сканирование с использованием инструментов динамического тестирования безопасности приложений (DAST) не инициируют оповещений и другие.

Подделка запросов со стороны сервера

Уязвимости данного вида возникают, когда веб-приложение получает удаленный ресурс без проверки предоставленного пользователем URL. Это позволяет злоумышленнику принудить приложение отправить сформированный запрос на неожиданный адрес, даже если он защищен брандмауэром, VPN или другим типом списка управления доступом к сети.

Поскольку современные веб-приложения предоставляют конечным пользователям удобные функции, получение URL становится обычной ситуацией. В результате данная уязвимость становится всё более распространенной [3].

2. СУЩЕСТВУЮЩИЕ МЕТОДИКИ БЕЗОПАСНОГО ВЕБ-ПРОГРАММИРОВАНИЯ

2.1. МЕТОДОЛОГИЯ SDL

Методология разработки программного обеспечения Security Development Lifecycle (SDL) является формальным подходом, который учитывает меры безопасности и конфиденциальности на всех этапах разработки. Она помогает

разработчикам создавать программное обеспечение высокой степени защищенности, обеспечивает соблюдение требований безопасности и конфиденциальности, а также позволяет снизить затраты на разработку.

Методология включает следующие стадии разработки безопасности программного обеспечения.

1. Обеспечение обучения. Эффективное обучение должно дополнять и укреплять политики безопасности, практики разработки безопасного программного обеспечения, стандарты и требования к безопасности на основе данных или новых технических возможностей. Важно отметить, что безопасность – это задача каждого участника процесса, но необязательно каждый должен быть экспертом по безопасности или стремиться стать опытным тестером на проникновение. Однако каждый должен понимать перспективы злоумышленника и его цели, чтобы повысить осведомленность о важности безопасности для организации.

2. Определение требований безопасности. Неотъемлемой частью разработки высокозащищенных приложений и систем является учет безопасности и конфиденциальности. Независимо от выбранной методологии разработки требования безопасности должны постоянно обновляться, чтобы отразить изменения в функциональности и смену окружения угроз. Наиболее подходящим временем для определения требований безопасности является начальный этап проектирования и планирования, поскольку это позволяет интегрировать безопасность и минимизировать возможные нарушения. Факторы, влияющие на требования безопасности, включают юридические и отраслевые стандарты, внутренние политики и практики программирования, анализ предыдущих инцидентов и известные угрозы. Для отслеживания этих требований необходимо использовать системы учета задач или анализировать данные, полученные в процессе разработки.

3. Определение метрик и отчетов о соответствии. Важно установить минимально допустимые уровни безопасности и обязать команды разработчиков их соблюдать. Раннее определение этих уровней помогает команде понять риски, связанные с проблемами безопасности, обнаруживать и исправлять дефекты безопасности на этапе разработки и применять соответствующие стандарты на протяжении всего проекта. Определение метрик для исправления ошибок включает четкое разграничение уровней серьезности уязвимостей безопасности.

4. Создание модели угроз. В средах с высоким уровнем риска для безопасности следует применять модель угроз. Анализ угроз может быть проведен на уровне компонентов, приложений или системы в целом. Это практика, которая позволяет командам разработчиков тщательно рассмотреть, задокументировать и, что важно, структурированно обсудить возможные послед-

ствия для безопасности в контексте запланированной операционной среды. Применение структурированного подхода к сценариям угроз помогает команде более эффективно и экономично выявлять уязвимости безопасности, оценивать риски от этих угроз, а затем выбирать соответствующие меры безопасности и предпринимать необходимые шаги для устранения угроз.

5. Установка требований к дизайну. Обычно методология SDL рассматривается как набор мер, направленных на обеспечение уверенности и помогающих инженерам внедрять «безопасные функции», то есть функции, разработанные с учетом безопасности. Для достижения этой цели инженеры обычно полагаются на такие функции безопасности, как криптография, аутентификация, журналирование и другие. Но во многих случаях их выбор или реализация оказываются сложными, из-за чего дизайнерские или реализационные решения могут привести к появлению уязвимостей. Поэтому крайне важно последовательно применять эти функции и иметь понимание о том, какую защиту они обеспечивают.

6. Определение и использование криптографических стандартов. С увеличением популярности мобильных и облачных вычислений важно гарантировать защиту всех данных, включая конфиденциальную информацию, а также управляющие и контрольные данные, от несанкционированного раскрытия или изменения при передаче или хранении. Обычно для этой цели используется шифрование. Неправильный выбор в использовании любого аспекта криптографии может иметь серьезные последствия, поэтому рекомендуется разработать четкие стандарты шифрования, которые учитывают детали каждого элемента реализации шифрования. Важно доверить эту задачу экспертам. Хорошим общим правилом является использование только проверенных отраслевых библиотек шифрования и обеспечение их реализации таким образом, чтобы их можно было легко заменить при необходимости.

7. Управление рисками безопасности, связанными с использованием стороннего ПО. В настоящее время большинство программных проектов в значительной степени зависит от использования сторонних компонентов – как коммерческих, так и с открытым исходным кодом. При выборе таких компонентов важно учитывать, какая уязвимость в них может повлиять на безопасность более обширной системы, в которую они интегрируются. Для снижения этого риска необходимо иметь точный список сторонних компонентов и план реагирования на обнаружение новых уязвимостей. Однако стоит также рассмотреть возможность проведения дополнительной проверки в соответствии с рисковой политикой организации, типом используемого компонента и потенциальным влиянием уязвимости на системы безопасности.

8. Использование проверенного инструментария. Определите и опубликуйте список инструментов, которые прошли проверку и были утверждены

для использования, включая соответствующие проверки безопасности, такие как параметры компилятора или компоновщика и предупреждения. Инженеры должны стремиться использовать самую последнюю версию утвержденных инструментов (например, утвержденные версии компилятора) и пользоваться преимуществами новых функций анализа безопасности и механизмов защиты.

9. Использование статических анализаторов кода. Анализ исходного кода перед компиляцией предоставляет высокомасштабируемый метод проверки безопасности кода и помогает гарантировать соблюдение политик безопасного программирования. Статический анализ исходного кода (SAST) обычно интегрируется в процесс фиксации изменений, чтобы обнаруживать уязвимости при каждой сборке или упаковке программного обеспечения. Однако некоторые инструменты внедряются непосредственно в среду разработки, чтобы обнаруживать определенные недостатки, такие как использование небезопасных или запрещенных функций, и заменять их на более безопасные альтернативы в процессе активной разработки. Нет универсального решения, и команды разработчиков должны определить оптимальную частоту проведения SAST и, возможно, использовать несколько подходов, направленных на достижение баланса между производительностью и надлежащим обеспечением безопасности [20].

10. Использование динамических анализаторов кода. Проверка работоспособности полностью скомпилированного или упакованного программного обеспечения во время его работы позволяет проверить функциональность, которая проявляется только при интеграции и запуске всех компонентов. Для этого обычно используется набор предварительно созданных атак или инструментов, специально разработанных для мониторинга поведения приложения и выявления проблем, связанных с повреждением памяти, с привилегиями пользователей и с другими критическими проблемами безопасности. Как и в случае с SAST, здесь нет универсального решения, и хотя некоторые инструменты, такие как инструменты сканирования веб-приложений, могут быть легче интегрированы в процесс непрерывной интеграции и доставки (CI/CD), а другие методы тестирования DAST (например, фаззинг) требуют особого подхода [20, 21].

11. Использование тестирования на проникновение. Penetration-тестирование – это процесс анализа безопасности программной системы, выполняемый квалифицированными специалистами по информационной безопасности, которые имитируют действия хакера. Главная цель теста на проникновение состоит в выявлении потенциальных уязвимостей, вызванных ошибками в написании кода, неправильными настройками системы или другими слабостями, связанными с развертыванием приложения. В результате такого тести-

рования обычно обнаруживается широкий спектр уязвимостей. Проведение Penetration-тестов часто сопровождается автоматическими и ручными проверками кода, чтобы создать условия для обеспечения более глубокого уровня анализа по сравнению с обычными возможностями разработчиков.

12. Установка стандарта реагирования на инциденты. Подготовка стандарта реагирования на инциденты имеет решающее значение для противодействия новым угрозам, которые могут возникнуть со временем. Он должен быть разработан совместно командой реагирования на инциденты информационной безопасности продукта (PSIRT) вашей организации. В рамках стандарта должно быть определено, к кому следует обращаться в случае чрезвычайной ситуации в области безопасности и как устанавливать протоколы для обслуживания безопасности, включая стандарты для кода, полученного от сторонних поставщиков и других групп внутри организации. Стандарт реагирования на инциденты должен быть протестирован до момента его фактического использования, чтобы убедиться в его эффективности [5].

Методология SDL была подвергнута тестированию, которое продемонстрировало ее высокую эффективность, в результате чего она была внедрена в широкомасштабное использование. Эта концепция принесла значительную пользу, поскольку существенно снизила частоту возникновения уязвимостей. С течением времени и благодаря усилиям идеологов и опытных специалистов методология регулярно обновляется и совершенствуется.

2.2. МЕТОДОЛОГИЯ BSIMM

Позднее, с широким распространением модели SDL и ее популярностью на рынке, стало ясно, что необходимо систематизировать накопленные знания о безопасной разработке. Кроме того, требовалось найти способ оценки и измерения эффективности этой концепции. В это время возникла и развилась BSIMM (Building Security In Maturity Model), которая предлагает более структурированное описание методологии SSDL (Secure Software Development Lifecycle). BSIMM помогает организациям планировать, осуществлять и измерять различные инициативы в области обеспечения безопасности [4].

«В 2008 году стало ясно, что организации выбирают разные пути для защиты своего программного обеспечения. Эксперты, входящие в состав нынешней группы Synopsys Software Integrity Group, приступили к сбору данных об этих различных путях с целью изучения организаций, которые были высокоэффективны в области безопасности программного обеспечения, провели личные интервью со специалистами по безопасности в организациях и опубликовали свои выводы» [6].

Методология разделена на 4 крупных домена.

- Управление (Governance) – набор практик, которые отвечают за организацию, управление и оценку эффективности безопасной разработки программного обеспечения (SSDL).
- База знаний (Intelligence) – практики, связанные со сбором и систематизацией информации об информационной безопасности внутри организации. Они направлены на распространение и усвоение практик разработки безопасного ПО в широком масштабе.
- Точки соприкосновения с жизненным циклом разработки ПО (SSDL Touchpoints) – практики, связанные с анализом и оценкой конкретных артефактов и процессов, входящих в жизненный цикл производства программного обеспечения.
- Развертывание и эксплуатация (Deployment) – набор практик, отвечающих за взаимодействие с отделами сетевой и инфраструктурной безопасности и службами технической поддержки.

Каждый из этих доменов, в свою очередь, подразделяется на 3 практики.

В общей сложности в BSIMM13 описано 125 активностей. В рамках практик и уровней зрелости нет строгих правил относительно количества активностей. Каждая активность имеет уникальный идентификатор и последовательную нумерацию.

Уровень зрелости отражает популярность активности среди участников и ее частоту использования. Уровень активности зависит от ее сложности и важности. В зависимости от версии BSIMM уровень активности также может изменяться. Фреймворк безопасной разработки (Software Security Framework, SSF) представляет собой набор руководящих принципов, на которых основывается структура BSIMM.

3. ФОРМИРОВАНИЕ МЕТОДИКИ БЕЗОПАСНОГО ПРОГРАММИРОВАНИЯ

В качестве основы формирования методики безопасного программирования возможно предложить внесение дополнительных разделов в методику BSIMM 13.

1. Искусственный интеллект

В сценариях защиты от несанкционированного доступа искусственный интеллект (ИИ) может быть более эффективным, чем человек, в обнаружении аномалий в корпоративных информационных системах. Например, система ИИ может быстро определить, что пользователь пытается войти в систему с необычного рабочего места, которое не принадлежит этому пользователю.

В таком случае ответом со стороны информационной безопасности может быть идентификация этого события сотрудником службы ИБ и принятие соответствующих мер для защиты системы.

В случаях потери данных ИИ способен предсказывать отказы оборудования, на котором обрабатывается информация, более быстро и точно, чем человек. Кроме того, нейросети могут предпринимать превентивные меры, такие как миграция данных на резервные мощности, для минимизации рисков потери данных.

Также ИИ может использоваться для обнаружения вредоносного контента. В контексте вредоносного контента понимается вредоносный код, нацеленный на кражу, блокировку санкционированного доступа или уничтожение значимой информации, а также спам или фишинг. В этом случае использование искусственного интеллекта позволяет проводить эвристическое сканирование и блокировать источники вредоносной информации, получаемой через Интернет.

Одним из важных аспектов работы ИИ в информационной безопасности является появление подобия «гонки вооружений» между специалистами ИБ и злоумышленниками. С распространением технологий искусственного интеллекта и машинного обучения злоумышленники также начинают применять нейросети в своих целях. Учитывая эти проблемы, можно сделать вывод, что использование ИИ в области информационной безопасности требует применения передовых разработок в этой сфере.

Таким образом, можно сделать вывод о том, что применение искусственного интеллекта в ИБ позволяет повысить ее эффективность и решить множество проблем, с которыми человеку сложно справиться из-за больших объемов и скорости обработки информации. Однако при разработке систем с использованием ИИ в информационной безопасности необходимо учитывать особенности этой предметной области, включая противодействие со стороны злоумышленников [11].

2. Big Data

В информационной безопасности концепция Big Data означает использование технологии обработки и анализа больших объемов данных с целью обеспечения безопасности. Данная методика предусматривает сбор и анализ данных, полученных из различных источников, включая структурированные и неструктурированные данные, с высокой скоростью обновления. Информация может поступать из информационных систем, бизнес-платформ, систем управления и связи, а также с устройств и датчиков.

Применение больших данных в обеспечении информационной безопасности имеет огромный потенциал, особенно в контексте развития облачных

сервисов и интернета вещей. Рост объемов данных при условии их своевременного и точного анализа позволяет получить более полное представление о процессах информационной безопасности, обнаруживать и анализировать угрозы и принимать эффективные меры по их предотвращению. Использование методов анализа больших данных позволяет повысить осведомленность о наличии или отсутствии угроз, их характере и в результате принимать более эффективные меры по защите информации [8].

При использовании больших данных в стратегии кибербезопасности появляются новые преимущества, но следует помнить, что такая технология также имеет риск быть атакованной. В то время как анализ объемных данных улучшает обнаружение вредоносной активности и предотвращение утечки данных, сама эта новая форма данных может стать источником потенциальных рисков, связанных с утечкой информации. Это может привести к серьезным последствиям для компании, которая использует такую технологию, поэтому ее, в свою очередь, тоже нужно защищать [9].

В результате можно сделать вывод о том, что технологии искусственного интеллекта и больших данных при совместном использовании с BSIMM смогут создать согласованную информационную систему, в которой методология обеспечит надежную защиту технологии больших данных, а Big Data позволит эффективнее выявлять аномалии, вызванные атаками злоумышленников. Таким образом, можно сделать вывод о том, что применение искусственного интеллекта в ИБ позволяет повысить ее эффективность и решить множество проблем, с которыми человеку сложно справиться из-за больших объемов и скорости обработки информации. Однако при разработке систем с использованием ИИ в информационной безопасности необходимо учитывать особенности этой предметной области, включая противодействие со стороны злоумышленников.

ЗАКЛЮЧЕНИЕ

В статье рассмотрены существующие методики безопасного веб-программирования. Предложены технологии по улучшению уровня защищенности, предоставляемого методикой BSIMM. Благодаря методике безопасного программирования можно обеспечить должный уровень защиты для каждого веб-приложения в зависимости от чувствительности обрабатываемой информации.

СПИСОК ЛИТЕРАТУРЫ

1. *Алекперов З.А.* Обзор популярных уязвимостей веб-приложений и их решений // Аллея науки. – 2019. – № 5 (32), т. 2. – С. 1098–1113. – URL: https://www.elibrary.ru/item.asp?id=38626043 (дата обращения: 06.12.2023).

2. Что такое CI/CD? Разбираемся с непрерывной интеграцией и непрерывной поставкой // Блог компании OTUS. – URL: https://habr.com/ru/companies/otus/articles/515078/ (дата обращения: 06.12.2023).

3. OWASP Top 10:2021. – URL: https://owasp.org/Top10/ (accessed: 06.12.2023).

4. BSIMM: вдумчиво о плюсах и минусах // Блог компании Swordfish Security. – URL: https://habr.com/ru/companies/swordfish_security/articles/680616/ (дата обращения: 06.12.2023).

5. What are the Microsoft SDL practices? – URL: https://www.microsoft.com/en-us/securityengineering/sdl/ (accessed: 06.12.2023).

6. BSIMM 13 foundations report 2022. – URL: https://www.synopsys.com/software-integrity/engage/bsimm-web/bsimm13-foundations#BSIMM13%20Foundations.indd%3A.67583%3A2668 (accessed: 06.12.2023).

7. OWASP. Стандарт верификации требований к безопасности приложений 4.0.3. – URL: https://github.com/OWASP/ASVS/blob/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-ru.pdf (дата обращения: 06.12.2023).

8. Security Vision. Big Data. – URL: https://www.securityvision.ru/info/big_data/ (дата обращения: 06.12.2023).

9. Почему Большие данные (Big Data). – URL: https://club.cnews.ru/blogs/entry/pochemu_bolshie_dannye_big_data_/ (дата обращения: 02.11.2023).

10. OWASP Application Security Verification Standard. – URL: https://owasp.org/www-project-application-security-verification-standard/ (accessed: 06.12.2023).

11. *Антюфеев А.Б.* Применение и проблемы искусственного интеллекта в информационной безопасности при защите бизнеса // Наука и бизнес: пути развития. – 2021. – № 12 (126). – С. 26–28. – URL: https://www.elibrary.ru/item.asp?id=48095280 (дата обращения: 06.12.2023).

12. *Пителинский К.В, Цапин Д.М.* Управление безопасностью информационных потоков методами технологии Big Data // Международный журнал социогуманитарных исследований. – 2021. – № 4 (4). – С. 11–20. – URL: https://www.elibrary.ru/item.asp?id=50198550 (дата обращения: 06.12.2023).

13. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud / M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca, R. Casallas, S. Gil // 2015 10th Computing Colombian

Conference (10CCC). – IEEE, 2015. – P. 583–590. – DOI: 10.1109/ColumbianCC.2015.7333476.

14. *Mikowski M., Powell J.* Single page web applications: JavaScript end-to-end. – Simon and Schuster, 2013. – 432 p.

15. HTTP метод GET / Э.Ф. Насиров, Д.С. Кириллов, М.В. Чернова, Г.Р. Мертинс // Актуальные вопросы современной науки и образования: сборник статей VII Международной научно-практической конференции. – Пенза, 2021. – С. 33–35. – URL: https://naukaip.ru/wp-content/uploads/2021/01/MK-984-1.pdf#page=33 – (дата обращения: 06.12.2023).

16. CI/CD Pipelines evolution and restructuring: a qualitative and quantitative study / F. Zampetti, S. Geremia, G. Bavota, M. Di Penta // 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME). – Luxembourg, 2021. – P. 471–482. DOI: 10.1109/ICSME52107.2021.00048.

17. *Thakur P.* Evaluation and implementation of progressive web application: thesis. – Helsinki Metropolia University of Applied Sciences, 2018. – URL: https://www.theseus.fi/bitstream/handle/10024/142997/PWA%20thesis.pdf (accessed: 06.12.2023).

18. A survey and comparison of relational and non-relational database / N. Jatana, S. Puri, M. Ahuja, I. Kathuria, D. Gosain // International Journal of Engineering Research & Technology. – 2012. – Vol. 1 (6). – P. 1–5. – DOI: 10.1142/9781848168701_0002.

19. *Boonkrong S., Somboonpattanakit C.* Dynamic salt generation and placement for secure password storing // IAENG International Journal of Computer Science. – 2016. – Vol. 43 (1). – P. 27–36. URL: https://www.iaeng.org/IJCS/issues_v43/issue_1/IJCS_43_1_04.pdf (accessed: 06.12.2023).

20. *Li J.* Vulnerabilities mapping based on OWASP-SANS: a survey for static application security testing (SAST) // Annals of Emerging Technologies in Computing (AETiC). – 2020. – Vol. 4 (3). – URL: https://arxiv.org/ftp/arxiv/papers/2004/2004.03216.pdf (accessed: 06.12.2023).

Архипова Анастасия Борисовна, доцент кафедры защиты информации Новосибирского государственного технического университета. Основное направление научных исследований – математическое моделирование в информационной безопасности, оценка качества социально значимой деятельности. E-mail: arhipova@corp.nstu.ru

Листаров Роман Евгеньевич, лаборант кафедры защиты информации Новосибирского государственного технического университета. Направления научных исследований – информационная безопасность, информационные технологии. E-mail: kaf_zi@corp.nstu.ru

DOI: 10.17212/2782-2230-2023-4-64-81

Formation of secure programming methods for development web applications*

A.B. Arkhipova[1], R.E. Listarov[2]

[1] *Novosibirsk State Technical University, 20 Karl Marx Prospekt, Novosibirsk, 630073, Russian Federation, Associate Professor of the Department of Information Security. E-mail: arhipova@corp.nstu.ru*

[2] *Novosibirsk State Technical University, 20 Karl Marx Prospekt, Novosibirsk, 630073, Russian Federation, laboratory assistant of the security department. E-mail: kaf_zi@corp.nstu.ru*

Today, cyberphysical systems often become targets of targeted cyberattacks, including the exploitation of vulnerabilities in software operating in such systems. Vulnerable software can be used to distribute malicious software, steal or disclose confidential data, affect protected information in violation of established rights, leading to destruction, destruction, distortion or malfunction of automated systems in which this software operates. Determining the methodology for the secure development of web applications is the most important stage in the development of a comprehensive protection system for the object of informatization. This article proposes technologies to improve the level of security provided by the BSIMM method of secure programming, with the help of which it is possible to ensure the proper level of protection for each web application, depending on the sensitivity of the information processed.

Keywords: information security, information security, secure software, security requirements, development lifecycle, security testing, security threats

REFERENCES

1. Alekperov Z.A. Obzor populyarnykh uyazvimostei veb-prilozhenii i ikh reshenii [Overview of popular vulnerabilities in web applications and their solutions]. *Alleya nauki = Alley of Science*, 2019, no. 5 (32), vol. 2, pp. 1098–1113. Available at: https://www.elibrary.ru/item.asp?id=38626043 (accessed 06.12.2023).

2. Chto takoe CI/CD? Razbiraemsya s nepreryvnoi integratsiei i nepreryvnoi postavkoi [What is CI/CD? We deal with continuous integration and non-continuous delivery]. Available at: https://habr.com/ru/companies/otus/articles/515078/ (accessed 06.12.2023).

3. OWASP Top 10:2021. Available at: https://owasp.org/Top10/ (accessed 06.12.2023).

4. BSIMM: vdumchivo o plyusakh i minusakh [BSIMM: Thoughtful about pros and cons]. Available at: https://habr.com/ru/companies/swordfish_security/articles/680616/ (accessed 06.12.2023).

* *Received 07 November 2023.*

5. What are the Microsoft SDL practices? Available at: https://www.microsoft.com/en-us/securityengineering/sdl/ (accessed 06.12.2023).

6. BSIMM 13 foundations report 2022. Available at: https://www.synopsys.com/software-integrity/engage/bsimm-web/bsimm13-foundations#BSIMM13%20Foundations.indd%3A.67583%3A2668 (accessed 06.12.2023).

7. OWASP. *Standart verifikatsii trebovanii k bezopasnosti prilozhenii 4.0.3* [Standard for verification of safety requirements of appendices 4.0.3]. Available at: https://github.com/OWASP/ASVS/blob/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-ru.pdf (accessed 06.12.2023).

8. Security Vision. *Big Data*. Available at: https://www.securityvision.ru/info/big_data/ - (accessed 06.12.2023).

9. Why Big Data is the new focus for information security. (In Russian). Available at: https://club.cnews.ru/blogs/entry/pochemu_bolshie_dannye_big_data_/ (accessed 02.11.2023).

10. OWASP Application Security Verification Standard. Available at: https://owasp.org/www-project-application-security-verification-standard/ (accessed 06.12.2023).

11. Antiufeev A.B. Primenenie i problemy iskusstvennogo intellekta v informatsionnoi bezopasnosti pri zashchite biznesa [Application and problems of artificial intelligence in information security in protecting business]. *Nauka i biznes: puti razvitiya = Science and business: development ways*, 2021, no. 12 (126), pp. 26–28. Available at: https://www.elibrary.ru/item.asp?id=48095280 (accessed 06.12.2023).

12. Pitelinskiy K.V, Tsapin D.M. Upravlenie bezopasnost'yu informatsionnykh potokov metodami tekhnologii Big Data [Information flows security management using Big Data technology methods]. *Mezhdunarodnyi zhurnal cotsiogumanitarnykh issledovanii = International Journal of Socio-Humanitarian Research*, 2021, no. 4 (4), pp. 11–20. Available at: https://www.elibrary.ru/item.asp?id=50198550 (accessed 06.12.2023).

13. Villamizar M., Garcés O., Castro H., Verano M., Salamanca L., Casallas R., Gil S. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. *2015 10th Computing Colombian Conference (10CCC)*. IEEE, 2015, pp. 583–590. DOI: 10.1109/ColumbianCC.2015.7333476.

14. Mikowski M., Powell J. *Single page web applications: JavaScript end-to-end*. Simon and Schuster, 2013. 432 p.

15. Nasirov E.F., Kirillov D.S., Chernova M.V., Mertins G.R. [HTTP GET method]. *Aktual'nye voprosy sovremennoi nauki i obrazovaniya* [Current issues of modern science and education]. Collection of articles of the VII International Scientific and Practical Conference. Penza, 2021, pp. 33–35. (In Russian). Available at: https://naukaip.ru/wp-content/uploads/2021/01/MK-984-1.pdf#page=33 (accessed 06.12.2023).

16. Zampetti F., Geremia S., Bavota G., Di Penta M. CI/CD Pipelines evolution and restructuring: a qualitative and quantitative study. *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Luxembourg, 2021, pp. 471–482. DOI: 10.1109/ICSME52107.2021.00048.

17. Thakur P. *Evaluation and implementation of progressive web applic*ation: thesis. Helsinki Metropolia University of Applied Sciences, 2018. Available at: https://www.theseus.fi/bitstream/handle/10024/142997/PWA%20thesis.pdf (accessed 06.12.2023).

18. Jatana N., Puri S., Ahuja M., Kathuria I., Gosain D. A survey and comparison of relational and non-relational database. *International Journal of Engineering Research & Technology*, 2012, vol. 1 (6), pp. 1–5. DOI: 10.1142/9781848168701_0002.

19. Boonkrong S., Somboonpattanakit C. Dynamic salt generation and placement for secure password storing. *IAENG International Journal of Computer Science*, 2016, vol. 43 (1), pp. 27–36. Available at: https://www.iaeng.org/IJCS/issues_v43/issue_1/IJCS_43_1_04.pdf (accessed 06.12.2023).

20. Li J. Vulnerabilities mapping based on OWASP-SANS: a survey for static application security testing (SAST). *Annals of Emerging Technologies in Computing (AETiC)*, 2020, vol. 4 (3). Available at: https://arxiv.org/ftp/arxiv/papers/2004/2004.03216.pdf (accessed 06.12.2023).

Для цитирования:

Архипова А.Б., Листаров Р.Е. Формирование методики безопасного программирования для разработки веб-приложений // Безопасность цифровых технологий. – 2023. – № 4 (111). – С. 64–81. – DOI: 10.17212/2782-2230-2023-4-64-81.

For citation:

Arkhipova A.B., Listarov R.E. Formirovanie metodiki bezopasnogo programmirovaniya dlya razrabotki veb-prilozhenii [Formation of secure programming methods for development web ap-plications]. *Bezopasnost' tsifrovykh tekhnologii = Digital Technology Security*, 2023, no. 4 (111), pp. 64–81. DOI: 10.17212/2782-2230-2023-4-64-81.