УДК 004

БЕЗОПАСНАЯ РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Тарасова Ю.А.

ООО «Антифишинг», Тверь, e-mail: tarasovayuliya00@gmail.com

Информационная безопасность и защита данных являются одними из ключевых элементов эффективной работы современного программного обеспечения. Это повышает актуальность написания качественного кода и сведения к минимуму потенциальной возможности атак со стороны злоумышленников. В данной статье рассматриваются основные типы атак, такие как атаки на XML-парсеры (защита от XXE), DDoS (Distributed Denial-of-Service), SQL-инъекции, XSS (Cross-Site Scripting) и некоторые другие виды уязвимостей. Приводятся основные рекомендации и требования при написании безопасного кода, применяя которые на практике компании и разработчики смогут повысить стабильность и снизить уязвимость программного обеспечения. В данной статье была поставлена цель провести анализ основных аспектов, связанных с безопасным программированием. Были изучены и проанализированы принципы и методы, которые помогут разработчикам создавать программное обеспечение с повышенным уровнем безопасности. Безопасное программирование является неотъемлемой частью разработки, и анализ, представленный в данной статье, направлен на выявление наиболее эффективных подходов и решений в этой области. В статье рассмотрены такие важные аспекты, как защита от уязвимостей, обработка входных данных, а также выбор безопасных инструментов и технологий. Статья предоставит читателям ценную информацию и рекомендации, которые помогут создавать безопасное и надежное программное обеспечение. Научная ценность работы состоит в предпринимаемой попытке систематизации знаний относительно вопроса методики написания программ, устойчивых к кибератакам. Результаты работы могут быть полезны для специалистов, задачей которых является разработка высокоэффективных и безопасных программ.

Ключевые слова: защита данных, безопасный код, программирование, вредоносная программа, программное обеспечение, информационная безопасность, уязвимость, атака

SECURE SOFTWARE DEVELOPMENT

Tarasova Yu.A.

LLC Antiphishing, Tver, e-mail: tarasovayuliya00@gmail.com

Information security and data protection are key elements of effective modern software development. This increases the importance of writing quality code and minimizing the potential for attacks from malicious actors. This article examines the main types of attacks, such as XML parser attacks (protection against XXE), DDoS (Distributed Denial-of-Service), SQL injections, XSS (Cross-Site Scripting), and other vulnerabilities. It provides essential recommendations and requirements for writing secure code, which, when applied in practice, can enhance the stability and reduce the vulnerability of software. The objective of this article is to analyze key aspects related to secure programming. We aim to explore the principles and methods that help developers create software with a high level of security. Secure programming is an integral part of the development process, and our analysis focuses on identifying the most effective approaches and solutions in this field. We will examine important aspects such as vulnerability protection, input data handling, and the selection of secure tools and technologies. Our article will provide readers with valuable information and recommendations to aid in the creation of secure and reliable software. The scientific value of this work lies in the attempt to systematize knowledge regarding the methodology of writing programs resilient to cyber-attacks. The findings of this study can be beneficial to specialists involved in the development of high-performance and secure software.

Keywords: data protection, secure code, programming, malware, software, information security, vulnerability, attack

Обеспечение безопасности и работоспособности является неотъемлемой частью проектирования программного обеспечения. Важно отметить, что внедрение решений по обеспечению информационной безопасности после разработки является дорогостоящей задачей. В связи с этим особенно актуализируются вопросы, связанные с повышением уровня безопасности на этапе проектирования программ [1].

В результате этого особенную актуальность получает такое направление развития, как безопасный код (безопасное программирование). Написание безопасного кода представляет собой такой принцип разработки программного обеспечения, при котором разработчики пытаются учесть всевозможные ошибки и минимизировать вероятность их возникновения. Это значительно повышает стабильность и снижает уязвимость программного обеспечения. Безопасное программирование способно защитить данные пользователя и снизить вероятность их кражи или фальсификации [2].

Таким образом, рассматриваемое направление имеет значительный уровень актуальности на сегодняшний день. В связи с этим складывается необходимость более детального рассмотрения и анализа основных вопросов относительно принципов и правил написания безопасного кода. Представленные результаты исследования отражают комплексный подход к анализу по теме и могут стать полезным материалом для специалистов, ведущих свою деятельность в данной области [3].

Результаты исследования и их обсуждение

Ключевыми аспектами по рассматриваемой теме являются: защита от атак на XML-парсеры (защита от XXE); кодирование и экранирование при работе с браузерами; работа с Cookie и HTTP-заголовками; обработка входных данных и работа с SQL. Анализ и учет при разработке данных аспектов являются принципами безопасного программирования. В последующих материалах представлены основные результаты анализа по отдельности данных аспектов [4, 5].

Защита от атак на XML-парсеры является важной частью обеспечения безопасности веб-приложений и систем, которые обрабатывают XML-данные. XML (Extensible Markup Language) – это формат данных, который широко используется для обмена данными между различными приложениями. Однако неправильная обработка XML может привести к различным уязвимостям и атакам, таким как атаки на внедрение кода (XML Injection) или атаки на отказ в обслуживании (Denial of Service, DoS) [6, 7] (рис. 1).

Вследствие этого для защиты от атак на XML-парсеры можно рекомендовать выполнение следующих требований при написании кода [8, 9]:

– обеспечение валидации и фильтрации входных данных. Предотвращение внедрения вредоносного XML-кода начинается с валидации и фильтрации входных данных, поступающих от пользователей или внешних источников. Решение данных задач может включать в себя проверку на наличие недопустимых символов и структуру XML-документов;

– использование безопасных парсеров. Необходимо использовать безопасные XML-парсеры, которые обеспечивают защиту от атак. Примерами являются парсеры с поддержкой OWASP XML Security Gateway или другие инструменты, спроектированные с учетом безопасности;

– защита от DDoS атак. Для предотвращения атак данного типа необходимо ограничивать глубину и сложность парсинга XML-документов. Это позволяет избежать ресурсоемких атак на вычислительную систему;

– обеспечение возможности аутентификации и авторизации. Важно гарантировать возможность того, что только авторизованные пользователи имеют доступ к XML-документам и могут выполнять действия на основе XML-данных;

– регулярные обновления и обучение персонала. Это является дополнительным требованием помимо представленных ранее принципов безопасного программирования. В дополнение для обеспечения должного уровня информационной безопасности важно постоянно обновлять XML-парсеры и библиотеки для обеспечения безопасности. Также важно проводить обучение персонала с учетом современных угроз и методов защиты.

Следующим аспектом безопасного программирования является экранирование (Escaping) – это важный аспект обеспечения безопасности при работе с браузерами и обработке пользовательских данных [10, 11]. Экранирование используется для предотвращения внедрения вредоносного кода (как HTML, JavaScript, CSS или SQL-инъекции) через пользовательские данные, примером чего является ввод, который пользователи отправляют на сервер или отображается в браузере (рис. 2).

Важно отметить следующие принципы написания безопасного кода в рамках данной области:

– HTML экранирование (HTML Escaping). Когда пользовательские данные выводятся на веб-страницу, они должны быть экранированы. Это нужно для того, чтобы все HTML-теги и специальные символы (например, <, >, &) были представлены как текст, а не интерпретированы как код. Это предотвращает XSS (Cross-Site Scripting) атаки, при которых злоумышленники внедряют вредоносный JavaScript на страницу;

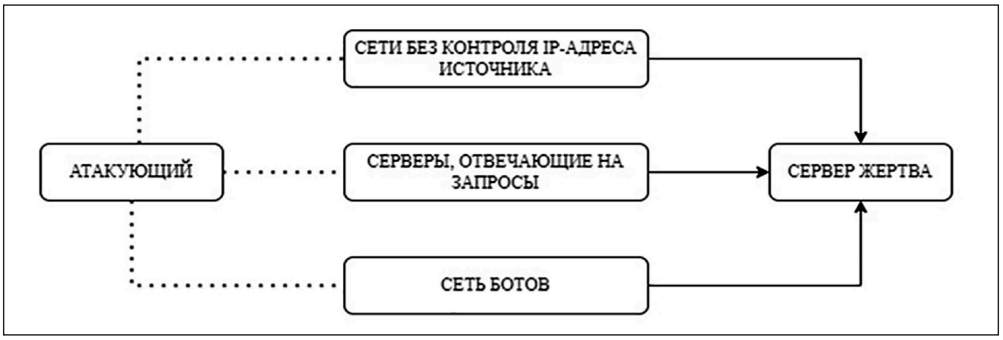

Рис. 1. Принцип DDoS-атаки

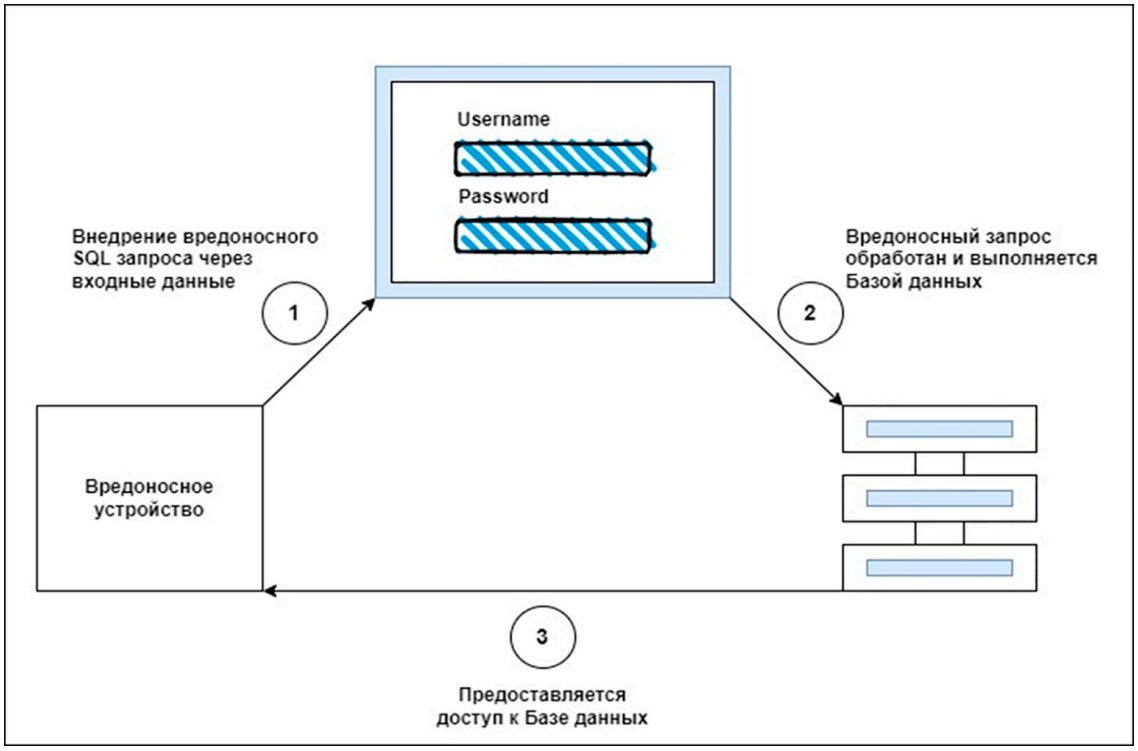

Рис. 2. Схема SQL-инъекции

– JavaScript экранирование (JavaScript Escaping). Если пользовательские данные используются в JavaScript-коде, они также должны быть экранированы, чтобы избежать выполнения вредоносных скриптов. Это особенно важно, когда данные вставляются в строки JavaScript, например, для работы с DOM;

– CSS экранирование (CSS Escaping). Если пользовательские данные влияют на стили CSS, то экранирование должно применяться для избегания инъекций CSS. Это позволяет избежать изменения стилей страницы нежелательным образом;

– SQL экранирование (SQL Escaping). При построении SQL-запросов из пользовательских данных, экранирование используется для предотвращения SQL-инъекций. Это гарантирует, что пользовательские данные не могут быть использованы для выполнения зловредных SQL-запросов к базе данных.

При этом можно отметить следующие примеры функций экранирования:

– в PHP: htmlspecialchars() и mysqli_real_escape_string()дляHTML-экранирования и SQL-экранирования соответственно;

– в JavaScript: encodeURIComponent() для URL-экранирования и методы, такие как createElement(), для безопасного создания DOM-элементов.

Важно понимать, что правильное экранирование должно соответствовать контексту использования данных (HTML, JavaScript, CSS, SQL) и выполняться на серверной стороне перед выводом данных на клиентскую сторону. Это помогает предотвратить множество видов атак и обеспечивает безопасность веб-приложений и пользовательских данных [12].

Продолжая анализ по представленной теме, важно отметить, что работа с Cookie и HTTP-заголовками в контексте безопасного кода имеет также большое значение для защиты веб-приложений от различных угроз. Далее представлены некоторые ключевые моменты относительно данного вопроса:

1. Cookie безопасность:

– HttpOnly – необходимо устанавливать флаг HttpOnly для Cookie. Это предотвращает доступ к Cookie из JavaScript и снижает риск кражи сеансов (session hijacking).

– Secure – необходимо устанавливать флаг Secure для Cookie, если они должны использоваться только через HTTPS. Это предотвращает перехват Cookie в открытых сетях.

– SameSite – необходимо использовать атрибут SameSite для ограничения того, какие сайты могут получать доступ к вашим Cookie. Рекомендуется установить его как «Strict» или «Lax» [13].

2. HTTP-заголовки безопасности:

– HTTP Strict Transport Security (HSTS). Необходимо включить заголовок HSTS, чтобы заставить клиентов использовать только HTTPS. Это предотвращает атаки перехвата. Примеры:

```
markdownCopy code
Strict-Transport-Security:max-age=31536000;
includeSubDomains; preload
```

– Content Security Policy (CSP). Необходимо установить заголовок CSP, чтобы ограничить источники, с которых загружается контент на сайте. Это помогает предотвратить атаки XSS. Примеры:

```
markdownCopy code
Content-Security-Policy: default-src 'self';
script-src 'self' 'unsafe-inline' 'unsafe-eval';
style-src 'self' 'unsafe-inline'
```

– X-Content-Type-Options. Необходимо установить заголовок X-Content-Type-Options для предотвращения атак с подменой типа контента. Примеры:

```
markdownCopy code
X-Content-Type-Options: nosniff
```

– X-Frame-Options: Необходимо установить заголовок X-Frame-Options для предотвращения атак clickjacking. Примеры:

```
markdownCopy code
X-Frame-Options: DENY
```

– X-XSS-Protection. Необходимо установить заголовок X-XSS-Protection для защиты от атак типа XSS. Примеры:

```
markdownCopy code
X-XSS-Protection: 1; mode=block
```

– Referrer-Policy. Необходимо установить заголовок Referrer-Policy, чтобы контролировать, какие данные о referer отправляются при переходах между страницами. Примеры:

```
markdownCopy code
Referrer-Policy: same-origin
```

Данные меры при написании кода помогут улучшить безопасность веб-приложения, защищая Cookie и управляя HTTP-заголовками. Заключительным из основных аспектов написания безопасного кода является работа с SQL. Обработка входных данных и работа с SQL в рамках безопасного кода крайне важны для предотвращения атак на базу данных, таких как, например, SQL инъекции. Далее представлены основные меры безопасности относительно данного вопроса [14]:

1. Обработка входных данных:

– параметризованные запросы. Важно всегда использовать параметризованные запросы или подготавливаемые выражения для выполнения SQL-запросов. Это предотвращает возможность выполнения SQL-инъекции;

– валидация входных данных. Необходимо обеспечить возможность при написании кода по проверке входных данных на корректность и соответствие ожидаемому формату;

– использование ORM. Использование Object-Relational Mapping (ORM) фреймворков, таких как SQLAlchemy в Python или Entity Framework в .NET, может упростить работу с базой данных и предотвратить многие типы инъекций.

2. Защита от SQL-инъекций:

– экранирование строк. Если отсутствует возможность использовать параметризованные запросы, то необходимо обязательно экранировать строки перед вставкой их в SQL запросы. При этом можно воспользоваться функциями экранирования, предоставляемыми конкретной СУБД;

– ограничение прав доступа. Важно назначить минимальные права доступа к базе данных пользователям и приложениям. «Принцип наименьших привилегий» помогает ограничить возможности злоумышленников;

– мониторинг и журналирование. Важно вести журнал запросов к базе данных и выполнять его мониторинг на предмет подозрительной активности. Это поможет быстро выявить попытки атак;

– обновление библиотеки и СУБД. Необходимо регулярно обновлять библиотеки и саму СУБД для устранения известных уязвимостей;

– защита от времени выполнения. Защита от атак может обеспечиваться на основе таймингов, устанавливая максимальное время выполнения запросов и отключая выполнение нескольких запросов в одном [15].

Заключение

Таким образом, основной целью представленного исследования являлось выполнение анализа относительно ключевых вопросов, связанных с написанием безопасного кода. В результате работы произведен анализ по таким направлениям, как защита

от атак на XML-парсеры (защита от XXE), кодирование и экранирование при работе с браузерами, работа с Cookie и HTTP-заголовками, а также обработка входных данных и работа с SQL.

Материалы статьи отражают комплексный подход к анализу по представленному направлению, связанному с безопасным программированием.

В заключение необходимо отметить, что интенсивно расширяющееся использование информационных технологий и программного обеспечения значительно увеличивает риски информационной безопасности. Электронная информация и данные, хранящиеся на веб-серверах, в информационных системах и иных ресурсах непрерывно подвергаются потенциальным угрозам со стороны злоумышленников. В связи с этим на сегодняшний день занимают особенное значение вопросы, связанные с безопасным программированием для возможности предотвращения негативных последствий уже на этапе проектирования программного обеспечения. Также, как уже было отмечено ранее, такой подход является наиболее экономически эффективным, так как интеграция мер по защите информации уже к готовым решениям является более дорогой задачей.

Список литературы

1. Попов В.Г., Галиаскаров Д.Ф., Гвоздев Л.Б. Актуальность обеспечения информационной безопасности в сетях IoT // StudNet. 2021. № 4 (4). С. 94–100.

2. Осинцев А.А., Лапонина О.Р. Тестирование уязвимостей во внешних сущностях xml веб-приложений // International Journal of Open Information Technologies. 2019. № 7 (10). С. 71–79.

3. Шутько Н.А. Теоретические понятия защиты Web-приложений от уязвимостей // Вестник науки. 2022. № 4-11 (56). С. 8–11.

4. The NIST National Vulnerability Database (NVD). [Электронный ресурс]. URL: https://nvd.nist.gov (дата обращения: 05.09.2023).

5. Шелухин О.И., Игумнов В.О. Гибридный способ обеспечения конфиденциальности пользователя при работе с веб-браузером // Наукоемкие технологии в космических исследованиях Земли. 2022. № 6. С. 4–12.

6. Сигалов Д.А., Хашаев А.А., Гамаюнов Д.Ю. Обнаружение серверных точек взаимодействия в веб-приложениях на основе анализа клиентского JavaScript-кода // Прикладная дискретная математика. 2022. № 53. С. 32–54.

7. Беликов Г.В., Крылов И.Д., Селищев В.А. SQL-инъекция как способ обхода авторизации // Известия ТулГУ. Технические науки. 2021. № 12. С. 217–221.

8. Наумова А.В., Чукова Д.И. Безопасное проектирование базы данных при использовании ORM // Вопросы кибербезопасности. 2019. № 3 (31). С. 14–20.

9. Байко Д.А., Гурвиц Г.А. Специфика создания современного бизнес-приложения в среде СУБД и обеспечение его безопасной работы // Проблемы Науки. 2019. № 1 (134). С. 20–28.

10. Комаров Т.И., Чепик Н.А., Иванов М.А. Разработка высокопроизводительного и безопасного с точки зрения работ с памятью ядра ОС // Вопросы кибербезопасности. 2019. № 3 (31). С. 25–32.

11. Polyzos G.C., Fotiou N. Building a reliable Internet of things using information-centric networking // Journal of Reliable Intelligent Environments. 2015. No. 1. P. 47–58.

12. L. Li, M.S. Hossain, A.A. Abd El- Latif, M.F. Alhamid. Distortion less secret image sharing scheme for Internet of Things system // Cluster Computing. 2017. P. 1–15.

13. Cetinkaya A., Ishii H., Hayakawa T. An Overview on Denial-of-Service Attacks in Control Systems // Attack Models and Security Analyses. 2019. No. 21. P. 1–29.

14. Chifor B., Patriciu V. Mitigating DoS attacks in publish-subscribe IoT networks // Proc. of Conf. Electronics, Computers and Artificial Intelligence. 2017. P. 1–6.

15. Maynard P., McLaughlin K., Sezer S. Decomposition and sequential-AND analysis of known cyber-attacks on critical infrastructure control systems // Journal of Cybersecurit. 2020. No. 6(1). P. 1–20.