УДК 004.056.5 URL: https://ptsj.bmstu.ru/catalog/icec/insec/1113.html

ПОЧЕМУ BINARYFORMATTER — САМАЯ ОПАСНАЯ ФУНКЦИЯ ПЛАТФОРМЫ .NET

Г.А. Крашкин krashkinga@student.bmstu.ru
Е.В. Глинская glinskaya@bmstu.ru
SPIN-код: 5430-3023

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

В статье исследованы риски применения небезопасной десериализации в веб-приложениях на платформе .NET Core. Обнаружена критическая уязвимость, позволяющая злоумышленнику выполнить произвольный код на сервере. Предложены практические рекомендации по замене устаревших компонентов и внедрению безопасных методов кодирования. Результаты исследования помогут разработчикам программного обеспечения устранить распространенные ошибки и повысить уровень защищенности приложений в целом, что, безусловно, представляет собой важный и своевременный вклад в решение актуальных вопросов обеспечения информационной безопасности современных программных решений на базе платформы .NET Core, имеющих широкое распространение в корпоративной среде.

Ключевые слова: обмен данными, веб-приложения, небезопасная десериализация, уязвимости .NET, выполнение кода, BinaryFormatter, защита приложений

Введение. Проблема уязвимостей, связанных с небезопасной десериализацией данных, приобретает все большую актуальность в современных веб-приложениях [1]. Многие разработчики используют сериализацию для обмена данными, не учитывая рисков выполнения произвольного кода.

Основные причины уязвимостей — использование устаревших форматов сериализации, таких как BinaryFormatter, недостаточная осведомленность разработчиков о механизмах эксплуатации и отсутствие строгой валидации входных данных. Решение проблемы требует комплексного подхода: отказа от опасных сериализаторов, внедрения безопасных альтернатив, таких как System.Text.Json, и повышения грамотности в вопросах безопасности среди разработчиков [2].

В рамках данного исследования был проведен анализ современных подходов к десериализации данных в приложениях, построенных на платформе .NET. Основное внимание уделялось выявлению системных недостатков в конфигурациях и практике реализации. Методология исследования включала:

1) *статический анализ кода* типовых проектов с открытым исходным кодом и корпоративных решений для выявления использования устаревших и небезопасных API, в первую очередь BinaryFormatter;

 ISSN 2541-8009

2) *практическое моделирование атак* в контролируемой тестовой среде (стенде) для эмпирической демонстрации последствий уязвимых конфигураций. Среда включала тестовое веб-приложение на ASP.NET Core [3] с умышленно внесенной уязвимостью;

3) *сравнительный анализ* безопасных альтернатив (System.Text.Json, XmlSerializer, DataContractSerializer) и методов защиты (например, использование SerializationBinder) на основе комплекса критериев, включая производительность, функциональность и соответствие современным стандартам информационной безопасности, таким как OWASP Top 10 [4].

Критические недостатки современных подходов к десериализации. Проведенный анализ выявил устойчивую тенденцию к использованию рискованных практик, корень которых лежит в применении механизмов, не предназначенных для обработки ненадежных данных. К числу наиболее распространенных проблем относятся:

– *использование небезопасных интерфейсов API*, в частности, BinaryFormatter. Данный сериализатор, разработанный для полного доверия в среде Windows Forms [5], не обеспечивает валидацию десериализуемых типов. Это позволяет злоумышленнику передать вредоносный объект, в процессе десериализации которого будет выполнен произвольный код;

– *недостаточное понимание разработчиками механизмов эксплуатации.* Многие программисты воспринимают десериализацию как рутинную операцию ввода-вывода, не отдавая себе отчета в том, что это процесс реконструкции объектов со всей их логикой, включая выполнение кода в конструкторах, сеттерах свойств и методах десериализации;

– *системное отсутствие строгой проверки и санации входных данных.* Входные данные, поступающие от клиента или из внешних систем, часто передаются на десериализацию без предварительной проверки их целостности, подлинности или структуры.

Последствия эксплуатации уязвимостей десериализации. Практическое моделирование атак на уязвимые конфигурации однозначно подтвердило возможность реализации деструктивных сценариев высшей степени тяжести:

– *удаленное выполнение кода (Remote Code Execution, RCE).* Атакующий получает возможность выполнить произвольные команды на атакуемом сервере в контексте учетной записи, под которой работает пул приложений. Это эквивалентно полному компрометированию сервера [6];

– *обход механизмов аутентификации и авторизации.* Путем подмены сериализованных объектов, отвечающих за хранение идентификатора пользователя или его ролей, злоумышленник может повысить свои привилегии в

системе [7] и получить доступ к чужой учетной записи или административному функционалу;

– *чтение и хищение конфиденциальных данных.* Эксплуатация уязвимости позволяет инициировать произвольные операции чтения из памяти процесса, что может привести к утечке сессионных токенов, ключей шифрования, паролей и другой чувствительной информации [8].

Предлагаемый комплекс мер по устранению уязвимостей. На основе анализа и тестирования сформирован комплекс практических рекомендаций для кардинального повышения безопасности:

– *поэтапный отказ от BinaryFormatter и его полное устранение из кодовой базы.* В качестве замены следует применять безопасные по своей природе сериализаторы, такие как System.Text.Json или XmlSerializer, которые не допускают десериализацию произвольных типов и выполняемого кода;

– *внедрение строгого контроля типов.* В сценариях, где использование мощных сериализаторов неизбежно, необходимо задействовать механизмы белого списка, такие как SerializationBinder, чтобы ограничить круг десериализуемых классов только явно разрешенными, доверенными типами;

– *повышение осведомленности разработчиков.* Ключевым элементом защиты служит человеческий фактор. Необходимо внедрять процедуры безопасного кодирования (Secure Coding) [9], проводить регулярный аудит кода [10] на предмет поиска уязвимых паттернов и организовывать обучающие семинары, наглядно демонстрирующие последствия атак.

Заключение. Проведенное исследование демонстрирует, что небезопасная десериализация представляет собой одну из наиболее серьезных угроз для веб-приложений, позволяя злоумышленникам достичь полного контроля над системой. Устаревшие практики разработки и недостаточная осведомленность программистов усугубляют эту проблему.

Литература

[1] Барабаш М.А. Распределенные системы. Сериализация. *RSDN Magazine*, 2012, № 1, с. 43–50. EDN: OXVAUL

[2] Волков М.Р. Идентификация и аутентификация в ASP.NET Core с использованием ASP.NET Core Identity. *Развитие науки и технологий в современной России (шифр – ВКРН). II Всерос. науч.-практ. конф.: сб. матер.* Москва, Изд-во Академическая среда, 2024, с. 41–56. EDN: AYUBUV

[3] Зыков С.В. *Проектирование автоматизированных систем.* Москва, Ай Пи Ар Медиа, 2024, 394 с. EDN: DKVCLV

[4] Козлов А.Д., Лебедев В.Н., Мараканов И.Н. *Современные технологии обмена данными для распределенных корпоративных информационных систем.*

 ISSN 2541-8009

Москва, Институт проблем управления им. В.А. Трапезникова РАН, 2016, 191 с. EDN: VWGRLV

[5] Черномордов С.В. Подключение базы данных к Windows Forms. Научные достижения и открытия современной молодежи. *Сб. ст. победителей Междунар. науч.-практ. конф.* Пенза, Наука и Просвещение (ИП Гуляев Г.Ю.), 2017, ч. 2, с. 144–147. EDN: YGQKTJ

[6] Истомин Д.А. Технология .net Core для разработки веб-приложений. *78-я науч. конф. студентов и аспирантов Белорусского государственного университета: матер.* Минск, Белорусский государственный университет, 2021, ч. 3, с. 310–313. EDN: HCYUYT

[7] Кравец В.В., Иванов О.А., Шорин Р.В. Логические уязвимости повышения привилегий в ОС Windows. *Теория и практика обеспечения информационной безопасности. Всерос. науч.-теор. конф.: сб. науч. тр.* Москва, Московский технический университет связи и информатики, 2021, с. 6–9. EDN: KIKXVU

[8] Кущий Д.Н., Федотенко И.И. Сравнение и примеры использование средств защиты информации в приложениях ASP.NET CORE. *Научные тенденции: вопросы точных и технических наук. XXX Междунар. науч. конф.: сб. науч. тр*. Санкт-Петербург, Центр Научных Публикаций Международной Объединенной Академии Наук, 2020, с. 9–13. https://doi.org/10.18411/sciencepublic-12-08-2020-03

[9] Barabanov A.V. Method for collecting architecture security-specific information for microservice-based systems. *Secure Information Technologies*, 2021, pp. 27–31. EDN: CZJFGD

[10] Хандожко Г.В. Аудит безопасности программного кода. *Студент: наука, профессия, жизнь. VII Всерос. студ. науч. конф. с междунар. уч.: матер.* Омск, Омский государственный университет путей сообщения, 2020, ч. 1, с. 291–296. EDN: SHSLBZ

Поступила в редакцию 09.11.2025

Крашкин Глеб Александрович — студент кафедры «Информационная безопасность», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Глинская Елена Вячеславовна — старший преподаватель кафедры «Информационная безопасность», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Научный руководитель — Басараб Михаил Алексеевич, доктор физико-математических наук, заведующий кафедрой «Информационная безопасность», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Ссылку на эту статью просим оформлять следующим образом:

Крашкин Г.А., Глинская Е.В. Почему BinaryFormatter — самая опасная функция в .NET. *Политехнический молодежный журнал*, 2026, № 03 (104). URL: https://ptsj.bmstu.ru/catalog/icec/insec/1113.html

WHY BINARYFORMATTER IS THE MOST DANGEROUS FUNCTION IN .NET

G.A. Krashkin krashkinga@student.bmstu.ru
E.V. Glinskaya glinskaya@bmstu.ru
SPIN-code: 5430-3023

Bauman Moscow State Technical University, Moscow, Russian Federation

In the article explores the risks of using unsafe deserialization in web applications on the .NET Core platform. A critical vulnerability is identified, allowing an attacker to execute arbitrary code on the server. Practical recommendations are provided for replacing outdated components and implementing secure coding methods. The research results will help software developers to eliminate common errors and improve the overall security of applications, which is an important and timely contribution to solving the current issues of information security in modern software solutions based on the .NET Core platform, which are widely used in corporate environments.

Keywords: data exchange, web applications, insecure deserialization, .NET vulnerabilities, code execution, BinaryFormatter, application security

Received 09.11.2025

Krashkin G.A. — Student of the Department of Information Security, Bauman Moscow State Technical University, Moscow, Russian Federation.

Glinskaya E.V. — Senior Lecturer at the Department of Information Security, Bauman Moscow State Technical University, Moscow, Russian Federation.

Scientific advisor — Basarab M.A., Dr. of Phys. and Math. Sci., Head of Department of Information Security, Bauman Moscow State Technical University, Moscow, Russian Federation.

Please cite this article in English as:
Krashkin G.A., Glinskaya E.V. Why BinaryFormatter is the most dangerous function in .NET. *Politekhnicheskiy molodezhnyy zhurnal*, 2026, no. 03 (104). (In Russ.). URL: https://ptsj.bmstu.ru/catalog/icec/insec/1113.html

 ISSN 2541-8009