УДК 004.423

БЕЗОПАСНОСТЬ В C#: МЕТОДЫ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ПРИ РАЗРАБОТКЕ ПРИЛОЖЕНИЙ НА C#

Кулинча П.В.
студент направления подготовки информатика и вычислительная техника,
Хакасский государственный университет имени Н.Ф. Катанова,
г. Абакан, Россия [1]

Аннотация: Безопасность является одним из наиболее важных аспектов разработки приложений на языке программирования C#. Язык программирования C#, разработанный компанией Microsoft, широко применяется для создания разнообразных приложений, включая веб-сервисы, настольные приложения и мобильные приложения. В данной статье рассматриваются методы обеспечения безопасности при разработке приложений на C#, а также предоставляются примеры кода и подробные объяснения.
Ключевые слова: языки программирования, C#, безопасность

SECURITY IN C#: A STUDY OF SECURITY METHODS IN THE DEVELOPMENT OF APPLICATIONS IN C#

Kulincha P.V.
student of computer science and computer engineering department,
N.F. Katanov Khakass State University,
Abakan, Russia

Abstract: Security is one of the most important aspects of application development in the C# programming language. The C# programming language developed by Microsoft is widely used to create a variety of applications, including web services, desktop applications and mobile applications. This article discusses security methods for

[1] Научный руководитель: Спирин Д.В. доцент кафедры ЦТиД, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия

developing applications in C#, as well as provides code examples and detailed explanations.

Keywords: programming languages, C#, security

Безопасность является одним из основных аспектов при разработке приложений на языке программирования C#. В современном информационном обществе, где взломы и утечки данных становятся все более распространенными, обеспечение безопасности приложений становится важной задачей для разработчиков [1].

Анализ уязвимостей является важным этапом при разработке безопасных приложений на C#. Разработчики должны быть в состоянии идентифицировать потенциальные уязвимости и применять соответствующие меры для их устранения [1].

Для анализа уязвимостей в приложении на C# можно использовать инструменты статического анализа кода, такие как ReSharper или SonarQube. Эти инструменты позволяют обнаруживать проблемные участки кода, такие как использование небезопасных функций, неправильное управление памятью или некорректную обработку пользовательского ввода (Рис. 1).

```
1  string userInput = GetUserInput();
2  if (!string.IsNullOrEmpty(userInput))
3  {
4      // Использование небезопасной функции
5      Console.WriteLine("User input: " + userInput);
6  }
```

Рисунок 1 – Инструменты статического анализа кода [разработано автором]

В приведенном выше примере кода, если разработчик не проверяет на пустое значение ввод пользователя (функция GetUserInput()), то пользователь может вводить произвольные данные, включая вредоносный код, который будет выводиться напрямую на консоль. Это может привести к уязвимости в приложении [2].

Анализ уязвимостей помогает выявить подобные проблемы и предлагает пути их решения, такие как валидация ввода, использование параметризованных запросов и т.д.

Шифрование данных является важной составляющей безопасности приложений на C#. Оно обеспечивает конфиденциальность и защиту данных от несанкционированного доступа.

Одним из распространенных методов шифрования данных в C# является использование класса System.Security.Cryptography.Aes, который предоставляет возможности симметричного шифрования. На рисунке 2 представлен пример использования шифрования в C#.

```
using System.Security.Cryptography;

public static byte[] EncryptData(byte[] data, byte[] key, byte[] iv)
{
    using (Aes aes = Aes.Create())
    {
        aes.Key = key;
        aes.IV = iv;

        using (MemoryStream memoryStream = new MemoryStream())
        {
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, aes.CreateEncryptor(), CryptoStreamMode.Write))
            {
                cryptoStream.Write(data, 0, data.Length);
                cryptoStream.FlushFinalBlock();
                return memoryStream.ToArray();
            }
        }
    }
}
```

Рисунок 2 – Метод шифрования данных [разработано автором]

В данном примере кода используется класс Aes для создания экземпляра алгоритма AES. Затем данные (data), ключ (key) и вектор инициализации (iv) устанавливаются в соответствующие свойства алгоритма. Создается MemoryStream, в который будут записываться зашифрованные данные. Затем создается CryptoStream, который использует алгоритм шифрования AES и записывает данные в MemoryStream. В результате вызова метода EncryptData возвращается массив зашифрованных данных [3].

Аутентификация играет важную роль в обеспечении безопасности приложений на C#. Надежная аутентификация позволяет проверить подлинность пользователей и предотвратить несанкционированный доступ к приложению.

В C# существует несколько методов обработки аутентификации, включая базовую аутентификацию, форму аутентификацию, аутентификацию на основе токенов и т.д. Важно правильно выбрать метод аутентификации в зависимости от требований приложения. Пример обработки аутентификации представлен на рисунке 3.

```
[HttpPost]
public IActionResult Login(LoginModel model)
{
    if (IsValidCredentials(model.Username, model.Password))
    {
        // Успешная аутентификация
        var claims = new List<Claim>
        {
            new Claim(ClaimTypes.Name, model.Username)
        };

        var claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);
        var authProperties = new AuthenticationProperties
        {
            ExpiresUtc = DateTimeOffset.UtcNow.AddHours(1),
            IsPersistent = model.RememberMe
        };

        HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
                                new ClaimsPrincipal(claimsIdentity),
                                authProperties);

        return RedirectToAction("Index", "Home");
    }

    ModelState.AddModelError(string.Empty, "Invalid login attempt.");
    return View(model);
}
```

Рисунок 3 – Обработка аутентификации [разработано автором]

В приведенном выше примере кода демонстрируется обработка аутентификации с использованием формы аутентификации. После проверки введенных пользователем учетных данных и подтверждения их подлинности, создается список утверждений (claims), содержащий информацию о пользователе (в данном случае, имя пользователя). Затем создается идентификация утверждений (claimsIdentity), используя схему аутентификации на основе куки. Дополнительно задаются параметры аутентификации, такие как срок действия и постоянность аутентификации. Затем вызывается метод SignInAsync, чтобы установить аутентификационные куки и перенаправить пользователя на главную страницу приложения [3].

В данной статье были рассмотрены методы обеспечения безопасности при разработке приложений на C#. Анализ уязвимостей, шифрование данных и обработка аутентификации играют важную роль в создании безопасных приложений. Описанные примеры кода и объяснения помогут разработчикам улучшить безопасность своих приложений и защитить данные пользователей от угроз.

Библиографический список:

1. Фленов М.Е. C# глазами хакера [Текст] / Фленов М.Е. – СПб.: БХВ-Петербург, 2023. – 224 с.: ил. – (Глазами хакера)
2. Троелсен, Э., Джепикс, Ф. Язык программирования C# 9 и платформа .NET 5: основные принципы и практики программирования, том 1, 10-е изд./Эндрю Троелсен, Филипп Джепикс; пер. с англ. Ю.Н. Артеменко. — Киев. : “Диалектика”, 2022.— 770 с.: ил. — Парал.тит. англ.
3. Тюкачев Н. А., Хлебостроев В. Г. Т 98 C#. Алгоритмы и структуры данных: Учебное пособие. — 3-е изд., стер. — СПб.: Издательство «Лань», 2022. — 232 с.: ил. (+ CD). — (Учебники для вузов. Специальная литература).

Оригинальность 76%