Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторным работам №4, 5

**Разработка защищенных приложений**

Выполнил:

Студент гр. 053501

Кривецкий Р.А.

Проверил:

Протько М. И.

Минск 2023

**Содержание**:

[1. **Введение** 2](#_Toc96083970)

[2. **Краткие теоретические сведения** 3](#_Toc96083971)

[3. **Демонстрация работы программы** 7](#_Toc96083972)

[4. **Вывод** 10](#_Toc96083973)

# **Введение**

Цель лабораторной работы – познакомиться с концепцией ролевого управления доступом и способами защиты программного обеспечения от существующих угроз.

Научиться разрабатывать приложения, которые используют ролевое управление доступом для разграничения полномочий пользователей. Получить навыки защиты разработанной программы от несанкционированного копирования и других угроз, которым может подвергаться программное обеспечение.

# **Краткие теоретические сведения**

**2.1 Возможные атаки и защита от них**

XSS (межсайтовый скриптинг) атаки – один из самых распространенных вариантов атаки на веб-сервер или компьютер пользователя, суть которой заключается во встраивании определенных HTML элементов и JavaScript кода в отданную сервером разметку, который выполнится на стороне клиента и может запустить какой-либо вирус или же произвести несанкционированное действие от имени пользователя на сервере. Принцип атаки приведен на рисунке 1.

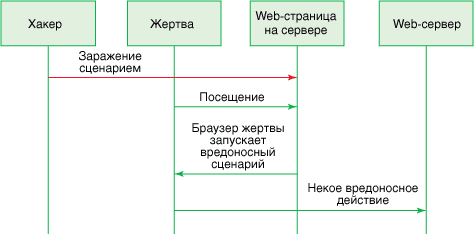


Рисунок 1. Принцип XSS атак

Если говорить немного подробнее, то можно выделить следующие виды XSS атак:

Хранимые (Постоянные)

Отражённые (Непостоянные)

Пример постоянных:

- Введённое злоумышленником специально сформированное сообщение в гостевую книгу (комментарий, сообщение форума, профиль) которое сохраняется на сервере, загружается с сервера каждый раз, когда пользователи запрашивают отображение этой страницы.

- Злоумышленник получил доступ к данным сервера, например, через SQL инъекцию (смешение двух типов атак – SQL injections и XSS атак), и внедрил в выдаваемые пользователю данные злонамеренный JavaScript код.

Пример непостоянных:

- На сайте присутствует поиск, который вместе с результатами поиска показывает что-то вроде «Вы искали: [строка поиска]», при этом данные не фильтруются должным образом. Поскольку такая страница отображается только для того, у кого есть ссылка на неё, то пока злоумышленник не отправит ссылку другим пользователям сайта, атака не сработает. Вместо отправки ссылки жертве, можно использовать размещение злонамеренного скрипта на нейтральном сайте, который посещает жертва.

Пример: “http://example.com/search.php?q=<script>DoSomething();</script>”

Если сайт не экранирует угловые скобки, преобразуя их в «&lt;» и «&gt;», получим скрипт на странице результатов поиска.

Для предотвращения XSS-атак приложение должно шифровать выходные данные страницы перед передачей их конечному пользователю. При шифровании выходных данных разметка HTML заменяется альтернативными представлениями - объектами. Браузер отображает эти объекты, но не запускает их. Например, *<script>* преобразуется в *&lt;script&gt;.*

Когда браузер встречает объекты, они преобразуются обратно в HTML и распечатываются, но они не запускаются. Например, если атакующий внедрит в переменное поле Web-страницы сервера строку *<script>alert("вы атакованы")</script>*, то при использовании описанной стратегии сервер возвратит строку *&lt;script&gt;alert("вы атакованы")&lt;/script&gt;.*

Когда браузер загрузит зашифрованный сценарий, он преобразует его к виду *<script>alert("вы атакованы")</script>* и отобразит сценарий в составе Web-страницы, но не запустит его.

SQL инъекции – также один из очень распространенных видов атак на приватные пользовательские данные или же непосредственно на устройство пользователя, а также и на сам сервер с целью уничтожения или распространения данных. Суть заключается в том, что некоторые веб формы могут быть связаны с SQL запросами и если пользователь введет в такую форму SQL запрос, он может исполниться прямо на стороне сервера и принести очень много проблем.

В реализации приложения этой лабораторной работы была использована библиотека System.Data.SqlClient. Сама по себе технология позволяет обернуть все SQL запросы в классы и функции языка программирования, тем самым не давая простым SQL запросам исполняться вовсе.

Также одной из атак является web crawling засекреченной информации. В частности такое можно увидеть при web crawling'е, когда по различным сайтам ходят программы, которые собирают различные данные, иногда к которым доступа у них не должно быть. Или же злоумышленник может ввести определенную ссылку исходя из логических именований и получить доступ к редактированию или эксплуатации чужих данных.

Основной защитой от такого вида атаки является постоянная проверка аутентификации и подлинности текущего пользователя на данную страницу. В примере 2 лабораторной работы при реализации Kerberos протокола мы обеспечивали такую защиту. Также одним из видом защиты может стать генерация уникальных ссылок и соответственно вероятность перехода по такой ссылке может быть крайне мала (если брать Sha-256 для примера, то вероятность коллизии будет равна 1/(16 ^ 256), т е 5.56е-309).

DoS атаки также популярный вид атак, который заключается в перегрузке сети запросами (рисунок 2). По сути это  [хакерская атака](https://ru.wikipedia.org/wiki/Хакерская_атака) на вычислительную систему с целью довести её до отказа, то есть создание таких условий, при которых добросовестные пользователи системы не могут получить доступ к предоставляемым системным ресурсам (серверам), либо этот доступ затруднён. Отказ «вражеской» системы может быть и шагом к овладению системой (если в нештатной ситуации ПО выдаёт какую-либо критическую информацию — например, версию, часть программного кода и т. д.). Но чаще это мера экономического давления: потеря простой службы, приносящей доход, счета от провайдера и меры по уходу от атаки ощутимо бьют «цель» по карману. В настоящее время DoS и DDoS-атаки наиболее популярны, так как позволяют довести до отказа практически любую систему, не оставляя юридически значимых улик.

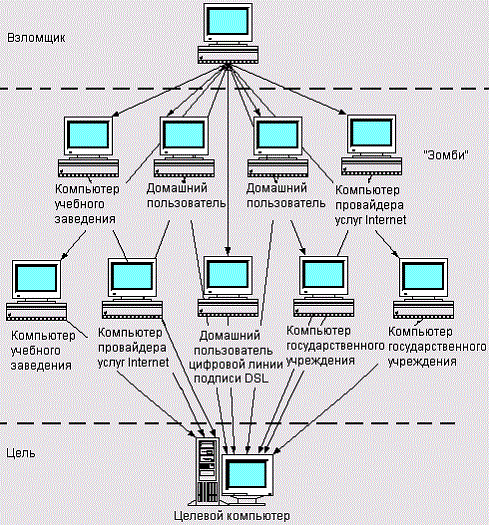


Рисунок 2. Принцип DoS атаки

Защититься от DoS атаки является очень трудной задачей. Для защиты от сетевых атак применяется ряд фильтров, подключенных к интернет-каналу с большой пропускной способностью. Фильтры действуют таким образом, что последовательно анализируют проходящий [трафик](https://ru.wikipedia.org/wiki/Сетевой_трафик), выявляя нестандартную сетевую активность и ошибки. В число анализируемых шаблонов нестандартного трафика входят все известные на сегодняшний день методы атак, в том числе реализуемые и при помощи распределённых бот-сетей. Фильтры могут реализовываться как на уровне [маршрутизаторов](https://ru.wikipedia.org/wiki/Маршрутизатор), управляемых [свичей](https://ru.wikipedia.org/wiki/Сетевой_коммутатор), так и специализированными аппаратными средствами. Также возможна установка задержек на запросы от пользовательских машин (такой способ и был реализован в лабораторной работе), однако при большом количестве устройств данная защита начинает заметно проседать.

**2.2 Инсталлятор**

Для постановки клиентской части приложения будет использоваться специальная программа-инсталлятор, которая позволяет установить программу на десктоп. Для этого будем использовать возможности Visual Studio 2019, а именно стандартное расширение Microsoft Visual Studio Installer Projects. Для создания инсталлятора необходимо добавить в решение новый проект и произвести стандартные манипуляции по настройке: указать имя разработчика, название приложения, веб сайт приложения, контактные данные и т.д. Также можно добавить собственную иконку приложения. После этого необходимо определить, какие файлы необходимо включить в инсталлятор (можно использовать Debug и Release версии приложения).

# **Демонстрация работы программы**

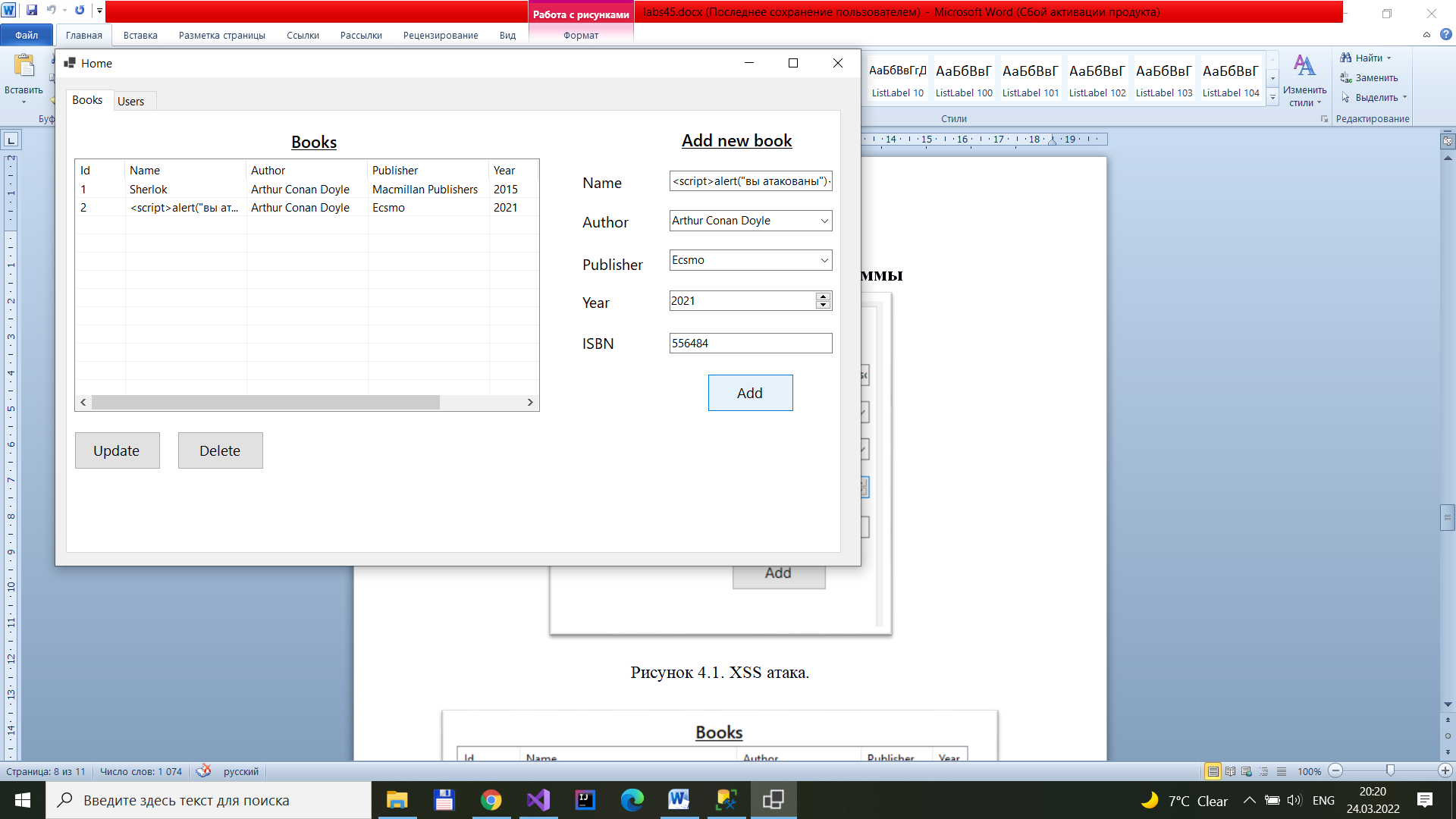


Рисунок 4.1. XSS атака.

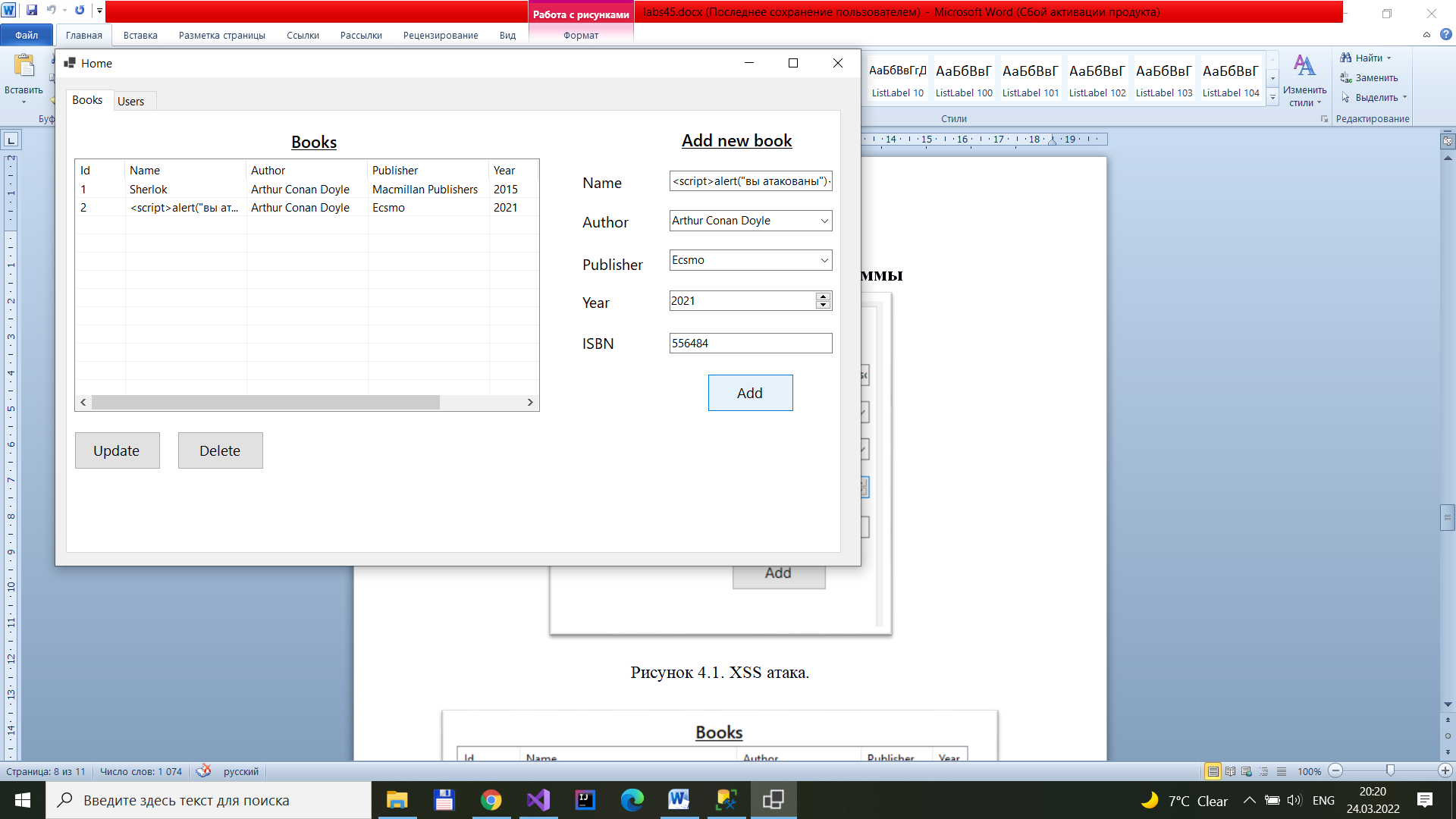


Рисунок 4.2. Подавленная XSS атака.

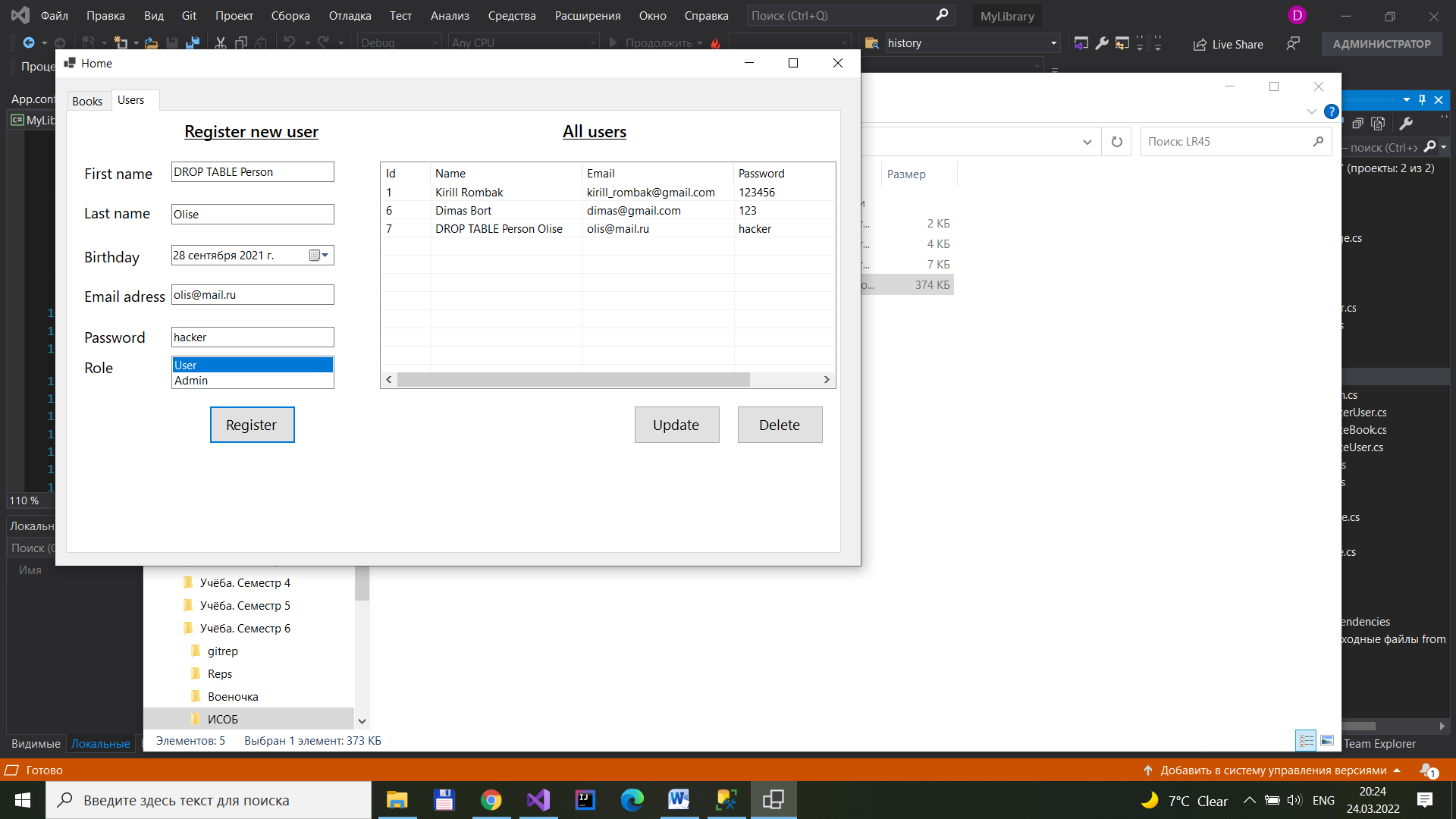


Рисунок 4.3. SQL-инъекция

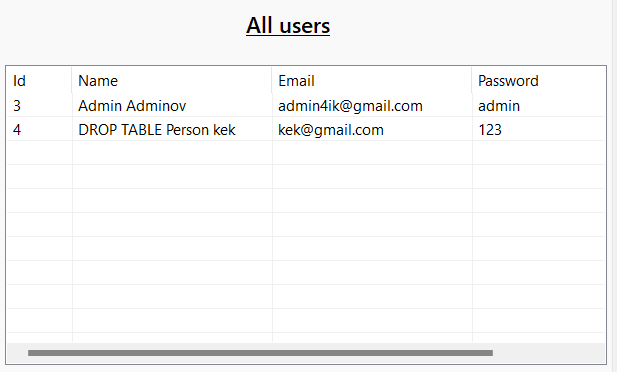


Рисунок 4.4. Подавленная SQL-инъекция

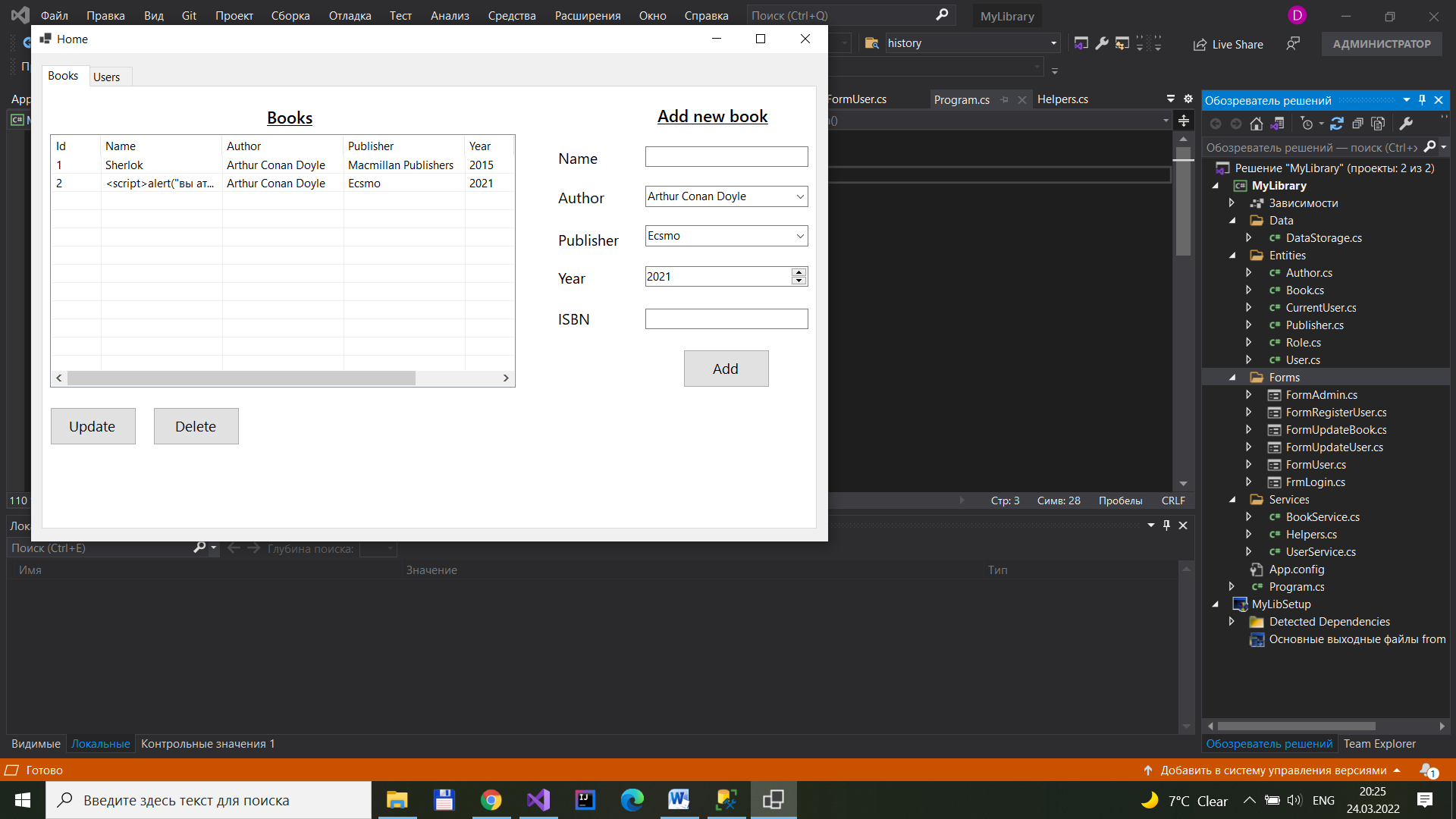


Рисунок 4.4. Принцип минимизации привилегий (роль admin)

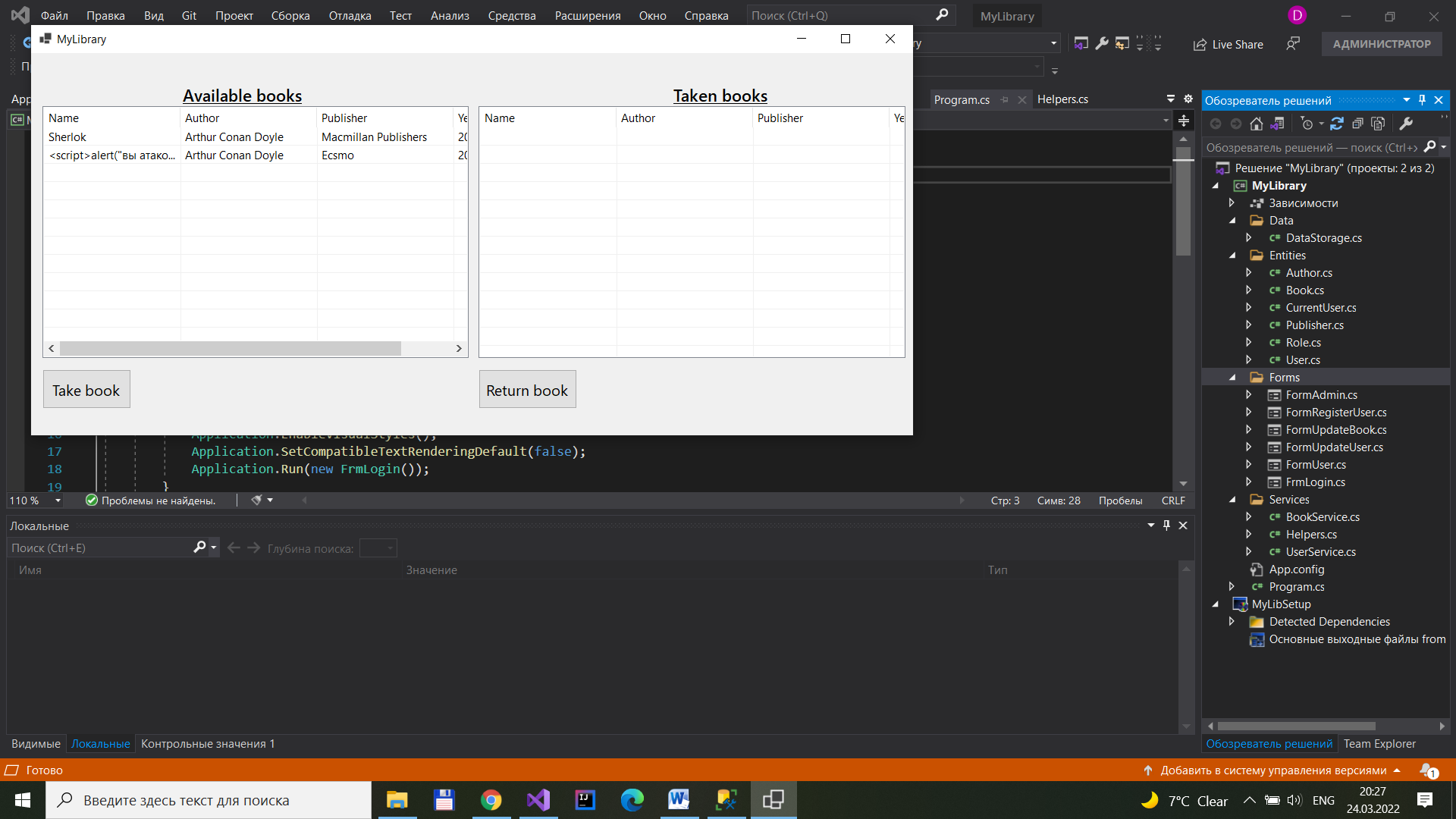


Рисунок 4.5. Принцип минимизации привилегий (роль user)

Так же фактически была реализована защита от атак типа «Переполнение буфера», поскольку программный продукт использует только безопасные конструкции языка C# и не использует потенциально уязвимых библиотек, например, библиотека Marshal.

# **Вывод**

В ходе выполнения лабораторной работы были реализована концепцией ролевого управления доступом и способами защиты программного обеспечения от существующих угроз. Было разрабатывать приложения, которые используют ролевое управление доступом для разграничения полномочий пользователей. Получить навыки защиты разработанной программы от несанкционированного копирования и других угроз, которым может подвергаться программное обеспечение.