Лекция №16
«Защищенность программы»

Москва 2019

Атаки

Современные приложения должны уметь противостоять атакам различных типов.
Необходимо знать тип атакк которым уязвимо ваше приложение.

Почему программисты разрабатывает уязвимые приложения?

- При разработке приложений часто не учитывается что будут предприниматься какие-либо атаки.
- Разработчики не имеют навыков написания защищённого кода
- Программисты это люди, а людям свойственно ошибаться

Создать полностью защищенное приложение от всех

Атаки

Переполнение буфера (англ. *Buffer Overflow*) — явление, возникающее, когда компьютерная программа записывает данные за пределами выделенного в памяти буфера.

Переполнение буфера обычно возникает из-за неправильной работы с данными, полученными извне, и памятью, при отсутствии жесткой защиты со стороны подсистемы программирования (компилятор или интерпретатор) и операционной системы. В результате переполнения могут быть испорчены данные, расположенные следом за буфером (или перед ним)[1].

Переполнение буфера является одним из наиболее популярных способов взлома компьютерных систем[2], так как большинство языков высокого уровня использует технологию стекового кадра — размещение данных в стеке процесса, смешивая данные программы с управляющими данными (в том числе адреса начала стекового кадра и адреса возврата из исполняемой функции).

Переполнение буфера может вызывать аварийное завершение или зависание программы, ведущее к *отказу обслуживания* (denial of service, DoS). Отдельные виды переполнений, например переполнение в стековом кадре, позволяют злоумышленнику загрузить и выполнить произвольный машинный код от имени программы и с правами учетной записи, от которой она выполняется[3].

Эксплоит

Эксплóит (англ. exploit, эксплуатировать) — компьютерная программа, фрагмент программного кода или последовательность команд, использующие уязвимости в программном обеспечении и применяемые для проведения атаки на вычислительную систему. Целью атаки может быть как захват контроля над системой (повышение привилегий), так и нарушение её функционирования (DoS-атака).
В зависимости от метода получения доступа к уязвимому программному обеспечению эксплоиты подразделяются на удалённые (англ. remote) и локальные (англ. local).

Удалённый эксплоит работает через сеть и использует уязвимость в защите без какого-либо предварительного доступа к уязвимой системе;
Локальный эксплоит запускается непосредственно в уязвимой системе, требуя предварительного доступа к ней. Обычно используется для получения взломщиком прав суперпользователя.

Code Red — компьютерный вирус, представляющий собой многовекторный сетевой червь, выпущенный в сеть 13 июля 2001 года. Он атаковал компьютеры с работающим веб-сервером Microsoft IIS, после успешного заражения начинал DoS-атаку на веб-страницу whitehouse.gov

Вставка SQL

Вставка SQL

Во время атаки на основе *вставки SQL* в данные, вводимые пользователем, вставляются команды для базы данных, изменяющие команды, отправляемые приложением базе данных на сервере. Таким атакам подвержены приложения, использующие в запросах SQL данные, вводимые пользователями.

Рассмотрим упрощенный пример кода C#, определяющий корректность введенного номера заказа (вводимого пользователем и хранящегося в переменной `Id`):

C#:

```
sql.Open();
sqlstring="SELECT HasShipped FROM orders WHERE ID='" + Id + "'";
SqlCommand cmd = new SqlCommand(sqlstring,sql);
if ((int)cmd.ExecuteScalar() != 0) {
    Status = "Yes";
} else {
    Status = "No";
}
```

Вставка SQL

```
If CType(cmd.ExecuteScalar() <> 0,Integer) Then
    Status = "Yes"
Else
    Status = "No"
End If
```

Обычные пользователи будут вводить такие ID, как "1234", а код будет присваивать переменной `Status` значение `"Yes"`, если значение из столбца `HasShipped` и строки с указанным ID равно `true`. Однако злоумышленник может ввести такое значение, как `"1234' drop table customers -"`. В этом случае приведенный код C# сконструирует следующий запрос SQL:

```
SELECT HasShipped FROM orders WHERE ID='1234' drop table customers --
```

Примечание

Точная структура команд SQL различна для разных серверов баз данных. Например, некоторые серверы баз данных требуют, чтобы команды были разделены точками с запятыми. Однако Microsoft SQL Server такого требования не предъявляет.

Если таблица с именем `customers` существует, и приложение имеет право на ее удаление, эта таблица будет потеряна. В зависимости от приложения и конфигурации базы данных, такие злонамеренные запросы можно использовать для получения информации из базы данных и выполнения команд операционной системы.

Атаки типа отказа в обслуживании

Атаки типа "отказ в обслуживании"

Атаки типа "отказ в обслуживании" (denial-of-service, DoS) мешают обычным пользователям использовать сетевую службу. Атаки DoS не предоставляют повышенного уровня доступа, вместо этого они приводят к тому, что пользователи не могут обратиться к приложению. Хотя наиболее опасными атаками считаются те, которые предоставляют злоумышленнику повышенные привилегии на целевом компьютере, атаки DoS могут очень дорого стоить тому, на кого направлены. Обычно потеря закрытой информации причиняет больший ущерб, чем атаки DoS, поэтому разработчикам следует направить большую часть усилий на закрытие уязвимостей, которые могут использоваться для получения информации.

Внимание!

В атаках типа "отказ в обслуживании" часто используются ошибки, которые в других случаях можно было бы не считать уязвимостями защиты. Например, закрытие уязвимости в клиент-серверном приложении, предлагающем пользователю ввести имя, но выходящем из строя, если длина введенного имени более 200 символов, может казаться не особо важным. Ведь в нормальных усло-

Атаки типа отказа в

Не все атаки DoS приводят к полной остановке приложения. Возможность для проведения атак DoS существует всегда, когда сервер для обработки запроса использует больше ресурсов, чем требуется клиенту для отправки этого запроса. Например, при создании клиент-серверного приложения, выделяющего 10 Мбайт памяти для каждого аутентифицированного пользователя, может использоваться следующая последовательность действий:

1. Получить запрос от пользователя.
2. Выделить пользователю оперативную память.
3. Выполнить аутентификацию пользователя.

Однако такое приложение будет чрезвычайно уязвимым для атак DoS, поскольку выделяет память прежде, чем аутентифицирует пользователя. Выделение памяти после аутентификации пользователя сделает приложение более устойчивым к таким атакам, так как атакующий должен предоставить учетные данные, перед тем как сервер выделит ему ресурсы. Следовательно, более безопасной будет такая последовательность:

1. Получить запрос от пользователя.
2. Выполнить аутентификацию пользователя.
3. Выделить пользователю оперативную память.

Криптографический взлом

Криптографический взлом

Криптографический взлом — это процесс перехвата зашифрованных данных и получение исходной информации, дешифруя ее без ключа. Криптографический взлом — сложный и затратный процесс, использующийся обычно только в том случае, если другие, более простые атаки на приложение не удаются. Если злоумышленник может сможет взломать код шифрования, он получит доступ к шифрованным каналам связи.

Как компрометируются каналы связи

Взаимодействие клиента с сервером выполняется по сети. Если злоумышленник получил контроль над оборудованием этой сети, он может перехватывать передающиеся данные. Это, как минимум, позволит получить важные сведения о том, как работает приложение. Кроме того, это может раскрыть закрытую информацию, например, имена пользователей и пароли, которые можно использовать для получения повышенных привилегий.

Криптографический взлом

Криптографический взлом

Криптографический взлом — это процесс перехвата зашифрованных данных и получение исходной информации, дешифруя ее без ключа. Криптографический взлом — сложный и затратный процесс, использующийся обычно только в том случае, если другие, более простые атаки на приложение не удаются. Если злоумышленник может сможет взломать код шифрования, он получит доступ к шифрованным каналам связи.

Как компрометируются каналы связи

Взаимодействие клиента с сервером выполняется по сети. Если злоумышленник получил контроль над оборудованием этой сети, он может перехватывать передающиеся данные. Это, как минимум, позволит получить важные сведения о том, как работает приложение. Кроме того, это может раскрыть закрытую информацию, например, имена пользователей и пароли, которые можно использовать для получения повышенных привилегий.

Криптографический взлом

Взломать зашифрованные данные могут только самые опытные злоумышленники. Даже имея необходимые знания и средства, для дешифрации данных могут потребоваться годы (или даже тысячелетия). Однако многие методы шифрования имеют слабости, позволяющие взломать ключ шифрования гораздо быстрее. Кроме того, если разработчик не следует рекомендациям по применению шифрования, надежность метода шифрования может значительно снизиться. Например, если приложение использует устойчивый метод шифрования, но злоумышленник имеет доступ к закрытому ключу, закрытый

Атаки с использованием посредника

Атаки с использованием посредника

Атаки с использованием посредника (man-in-the-middle, MITM) — сложны
атаки, в которых злоумышленник устанавливает сервер, перехватывающи
запросы клиентов. Затем злоумышленник отправляет настоящему сервер
запросы от лица пользователей (рис. 1.3). Таким образом, злоумышленни
может анализировать и изменять данные, передаваемые между клиенто
и сервером.

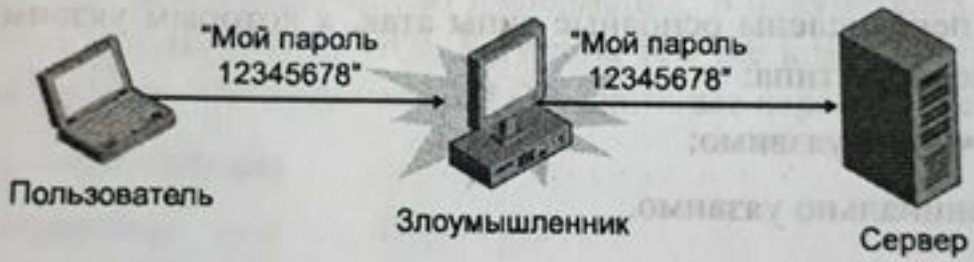


Рис. 1.3. Атака с использованием посредника

Атакам с использованием посредника подвержено любое приложение,
требующее шифрования и аутентификации. Такие атаки обычно предотв
щаются применением аутентификации и шифрования на уровне операци
ной системы, а не приложения. Для предотвращения атак с использовани
посредника при взаимодействии двух узлов, без необходимости реализа

Типы атак, к которым уязвимы приложения

Типы атак, к которым уязвимы определенные приложения

В табл. 1.1 перечислены основные типы атак, к которым уязвимы приложения определенного типа:

- O — обычно не уязвимо;
- X — потенциально уязвимо.

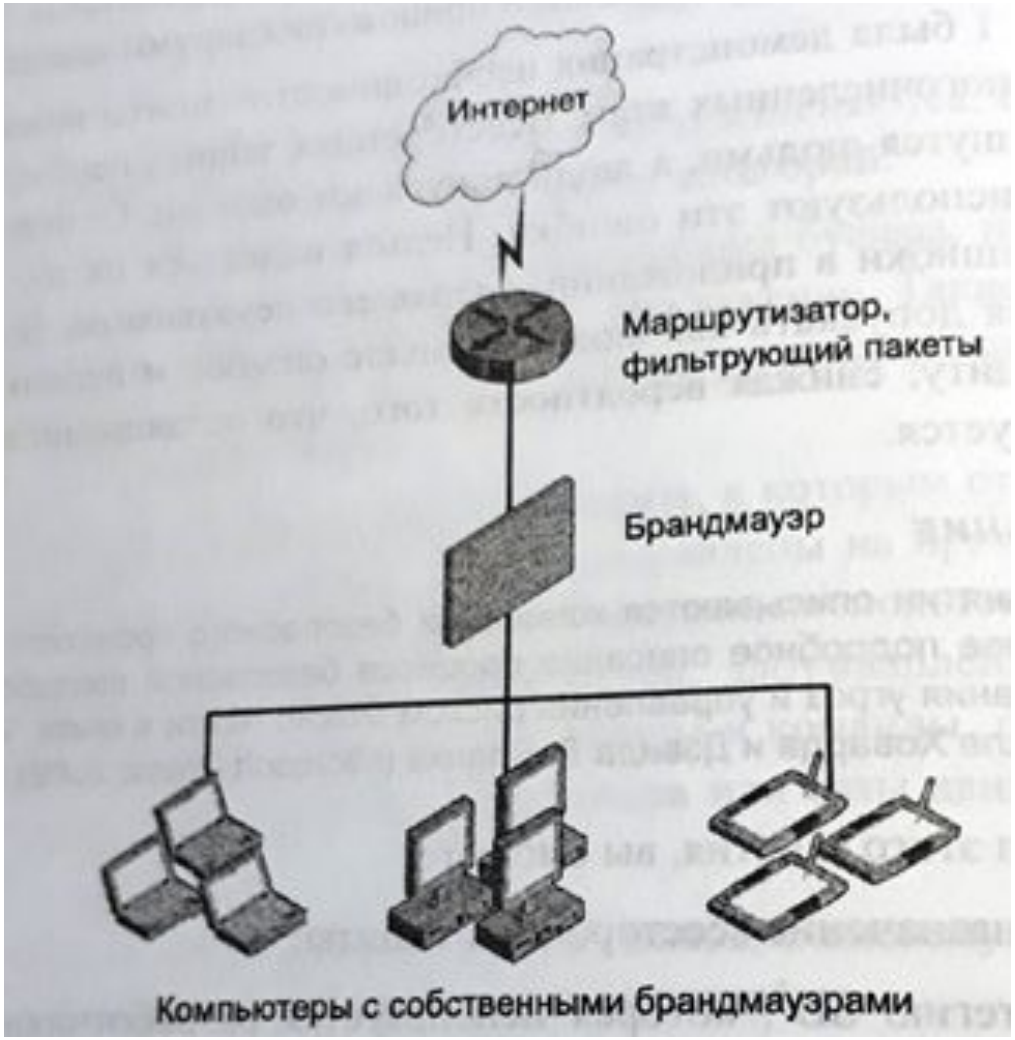
Таблица 1.1. Уязвимость приложений

	Веб-приложения ASP.NET	Приложения x.NET Framework Windows Forms
Переполнение буфера	O	O
Ошибки канонизации	X	X
Межсайтовое кодирование	X	O
Вставка SQL	X	X
Отказ в обслуживании	X	X
Использование посредника	X	X
Взлом паролей	X	X
Криптографический взлом	X	X

Проектирование защиты приложения

- Снизить вероятность появления уязвимостей.
- Изучить концепции безопасного проектирования приложений.
- Проектировать приложения, защищенные по умолчанию.
- Обеспечивать защиту во время и после разработки
- Несколько уровней защиты от возможных атак

Проектирование защиты приложения



Проектирование защиты приложения

- защита при проектировании. Разработчики должны следовать принципам создания защищенного кода и реализовывать функции защиты, чтобы устранить возможные уязвимости;
- защита по умолчанию. Если приложение защищено по умолчанию, конечные пользователи могут устанавливать его, не изменяя параметры по умолчанию. Неиспользуемые или снижающие уровень защиты функции отключены, и если пользователи захотят включить их, им придется сделать это явно;
- защита при развертывании. После установки приложения можно поддерживать его защиту, добавляя исправления системы безопасности, проводя мониторинг атак и выполняя аудит для выявления фактов злонамеренного использования.

Защита при проектировании

Шифрование
Аутентификация (процесс подтверждения личности пользователя при помощи имени пользователя и пароля)
Авторизация (имеет ли пользователь достаточно прав для выполнения запрошенного действия)
Брандмауэр
Если данные хранит или передает данные, ценные для злоумышленников, то используйте шифрование.

Некоторые ошибки разработчиков при выборе сетевых протоколов

Не документируют номера используемых портов, чтобы настроить брандмауэр
Нет возможности изменения номеров портов
Установка сеанса с TCP от сервера к клиенту.

Принцип наименьших привилегий

Проектировать приложения так, чтобы они использовали минимальные привилегий, необходимые для выполнения нужных действий.
Задокументируйте какие именно привилегии необходимы конченым пользователям, чтобы работать с приложению.

Интерактивные приложения следует запускать с привилегиями стандартного пользователя.

Службы в контексте ограниченной учетной записи пользователя.

Различные приложения для различных пользователей.

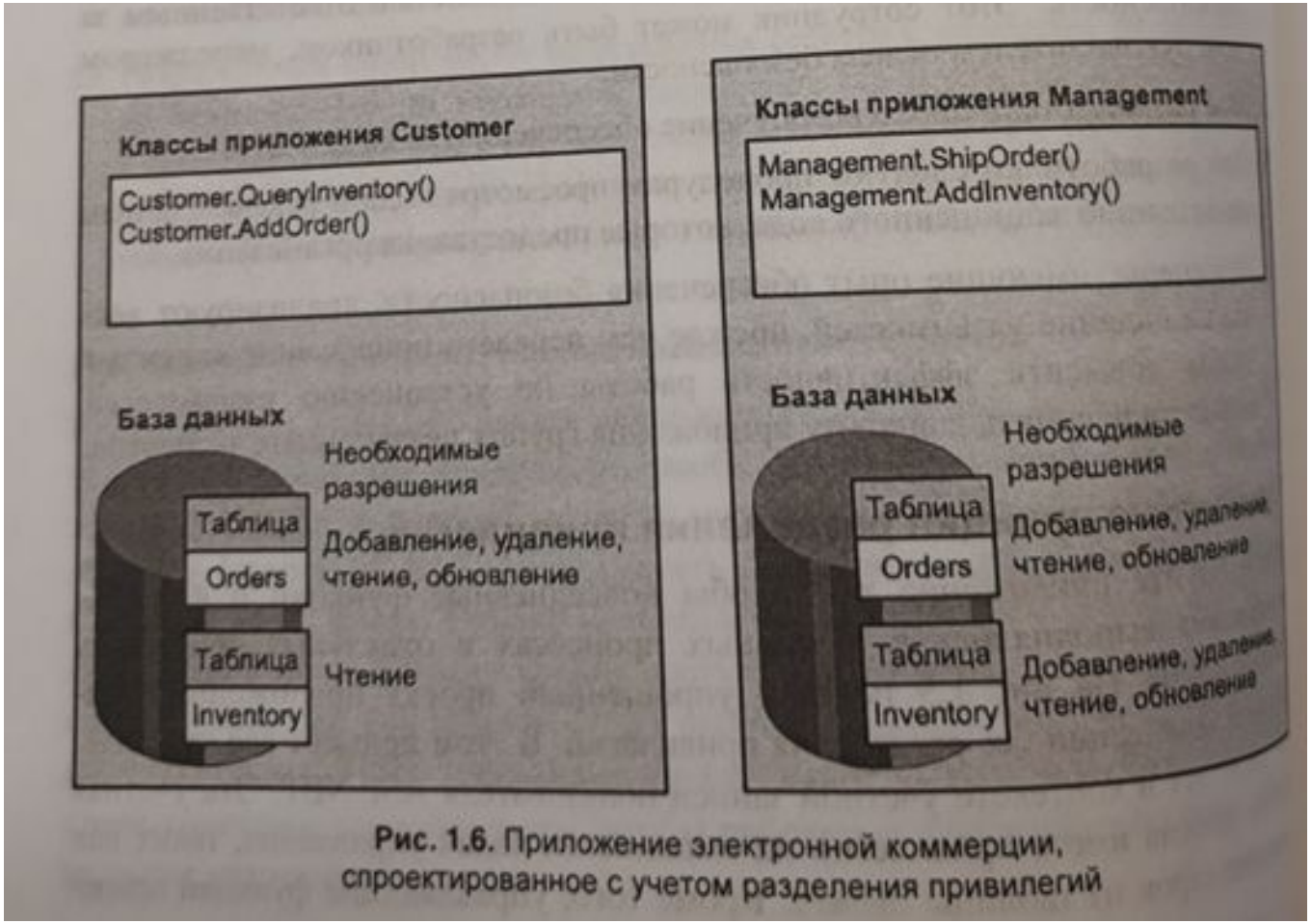


Рис. 1.6. Приложение электронной коммерции, спроектированное с учетом разделения привилегий

Уменьшение поверхности атаки

Приложение должно иметь минимальную поверхность атаки, предлагая пользователю простейший из возможных интерфейсов и предоставляя пользователям (и другим приложениям) минимум способов ввода запросов. Уменьшение поверхности атаки дает злоумышленникам меньше возможностей для проведения атак, что снижает вероятность их успеха. Для уменьшения поверхности атаки используйте следующее:

- принимайте входящие подключения через минимальное количество портов;
- минимизируйте число запущенных служб;
- минимизируйте число страниц в приложении ASP.NET;
- минимизируйте число учетных записей, имеющих право использовать приложение, и следуйте принципам минимальных привилегий и разделения привилегий;
- минимизируйте число доступных пользователям методов аутентификации, но не в ущерб гибкости приложения;
- минимизируйте число методов ввода данных пользователями;
- минимизируйте число компонентов приложения, устанавливаемых по умолчанию.

Уменьшение поверхности атаки

Приложение должно иметь минимальную поверхность атаки, предлагая пользователю простейший из возможных интерфейсов и предоставляя пользователям (и другим приложениям) минимум способов ввода запросов. Уменьшение поверхности атаки дает злоумышленникам меньше возможностей для проведения атак, что снижает вероятность их успеха. Для уменьшения поверхности атаки используйте следующее:

- принимайте входящие подключения через минимальное количество портов;
- минимизируйте число запущенных служб;
- минимизируйте число страниц в приложении ASP.NET;
- минимизируйте число учетных записей, имеющих право использовать приложение, и следуйте принципам минимальных привилегий и разделения привилегий;
- минимизируйте число доступных пользователям методов аутентификации, но не в ущерб гибкости приложения;
- минимизируйте число методов ввода данных пользователями;
- минимизируйте число компонентов приложения, устанавливаемых по умолчанию.

Криптография в NET

Algorithm	Description
AES	AES (Advanced Encryption Standard) is a symmetric algorithm. It was designed for both software and hardware. It has support for 128-bit data and 128,192,256-bit key.
DES	DES (Data Encryption Standard) is a symmetric algorithm published by National Institute of Standard and Technology (NIST).
RC2	RC2 (Ron's Code or Rivest Cipher) also known as ARC2 is a symmetric algorithm designed by Ron Rivest.
Rijndael	Rijndael is symmetric algorithm chosen by NSA as a Advanced Encryption Standard (AES).
TripleDes	TripleDes also known as 3DES (Triple Data Encryption Standard) applies DES algorithm three times to each data block.

Асимметричное шифрование

Асимметричное шифрование Асимметричное шифрование использует пару из двух ключей вместо одного для шифрования. Эти два ключа математически связаны друг с другом. Один из ключей называется открытым ключом, а другой - закрытым. ключ. Вы используете один из ключей для шифрования данных и другой для расшифровки данных. Другой ключ должен быть от пара ключей, которые вы сгенерировали. Шифрование, которое вы делаете с этими ключами, является взаимозаменяемым. Например, если key1 зашифровывает данные, тогда key2 может расшифровать их, и если key2 зашифрует данные, то key1 может расшифровать их, потому что один из них могут быть переданы каждому, а другой должен храниться в секрете.

Асимметричное шифрование

NET Framework предоставляет несколько асимметричных алгоритмов для работы.

Table 13-2. Asymmetric Algorithms

Algorithm	Description
RSA	RSA is an asymmetric algorithm commonly used by modern computers.
DSA	DSA (Digital Signature Algorithm), produced by NIST, is a standard to create digital signatures for data integrity.
ECDsa	ECDsa (Elliptic Curve Digital Signature) offers variant of the DSA.
ECDiffieHellman	Provides a basic set of operations that ECDH implementations must support.

Асимметричное шифрование

Метод ToXmlString возвращает открытый или закрытый ключ на основе логического значения. Для генерации закрытый ключ делает значение истинным, а для открытого ключа значение должно быть ложным.