УДК 004.584

А. В. Лыско, Е. Г. Шапович
Учреждение образования «Барановичский государственный университет», Барановичи, Республика Беларусь

ПРОБЛЕМА МЕТОДА ПЕРЕХВАТА СЕТЕВОГО ТРАФИКА

Введение. Протоколом в компьютерных сетях называется набор правил по той или иной работе с данными. Сетевой протокол — набор правил и действий (очерёдности действий), позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами [1]. Основным требования ко всем компьютерным сетям всегда была безопасность передачи данных. В данной статье мы рассмотрим метод решения проблема перехвата сетевого трафика.

Основная часть. Наиболее распространённой системой классификации сетевых протоколов является так называемая модель Open System Interconnection (далее — OSI), в соответствии с которой протоколы делятся на 7 уровней по своему назначению — от физического (формирование и распознавание электрических или других сигналов) до прикладного (интерфейс программирования приложений для передачи информации приложениями).

Сетевые протоколы предписывают правила работы компьютерам, которые подключены к сети. Они строятся по многоуровневому принципу. Протокол некоторого уровня определяет одно из технических правил связи. В настоящее время для сетевых протоколов используется модель OSI.

Модель OSI — это 7-уровневая логическая модель работы сети. Модель OSI реализуется группой протоколов и правил связи, организованных в несколько уровней [2]:

– на физическом уровне определяются физические (механические, электрические, оптические) характеристики линий связи;

– на канальном уровне определяются правила использования физического уровня узлами сети;

– сетевой уровень отвечает за адресацию и доставку сообщений;

– транспортный уровень контролирует очередность прохождения компонентов сообщения;

– задача сеансового уровня — координация связи между двумя прикладными программами, работающими на разных рабочих станциях;

– уровень представления служит для преобразования данных из внутреннего формата компьютера в формат передачи;

– прикладной уровень является пограничным между прикладной программой и другими уровнями — обеспечивает удобный интерфейс связи сетевых программ пользователя.

Протокол SSH — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Схож по функциональности с протоколами Telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH допускает выбор различных алгоритмов шифрования. SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.

SSH позволяет безопасно передавать в незащищённой среде практически любой другой сетевой протокол. Таким образом, можно не только удалённо работать на компьютере через командную оболочку, но и передавать по шифрованному каналу звуковой поток или видео (например, с веб-камеры).

В 1995 году Тату Юлёнен, исследователь из Технологического университета Хельсинки, Финляндия, разработал первую версию протокола (теперь называемого SSH-1), вызванную атакой по сбору пароля в его университетской сети. Целью SSH было заменить более ранние протоколы rlogin, TELNET, FTP и rsh, которые не обеспечивали строгую аутентификацию и конфиденциальность. Юлёнен выпустил свою реализацию как бесплатное ПО в июле 1995 года, и инструмент быстро завоевал популярность. К концу 1995 года база пользователей SSH выросла до 20 000 пользователей в пятидесяти странах.

В 1996 году была разработана более безопасная версия протокола, SSH-2, несовместимая с SSH-1. Протокол приобрел ещё большую популярность, и к 2000 году у него было около двух миллионов пользователей. В настоящее время под термином «SSH» обычно подразумевается именно SSH-2, так как первая версия протокола ввиду существенных недостатков сейчас практически не применяется.

Протокол SSH-1, в отличие от протокола telnet, устойчив к атакам прослушивания трафика («снифинг»), но неустойчив к атакам «человек посередине». Протокол SSH-2 также устойчив к атакам путём присоединения посередине, так как невозможно включиться в уже установленную сессию или перехватить её.

Для предотвращения атак «человек посередине» при подключении к хосту, ключ которого ещё не известен клиенту, клиентское ПО показывает пользователю «слепок ключа». Рекомендуется тщательно сверять показываемый клиентским ПО «слепок ключа» со слепком ключа сервера, желательно полученным по надёжным каналам связи или лично [3].

Протокол SSH шифрует передаваемый трафик при помощи алгоритма Диффи-Хеллмана [4].

Сформулируем суть проблемы. Криптографический ключ необходим для шифрования и дешифрования сообщений. Давайте представим, что мы хотим обмениваться секретными сообщениями через курьера. Если сообщение незашифровано, курьер сможет прочитать его. Т. е. нам нужно зашифровать переписку, но вначале

мы должны передать секретный ключ на другой конец провода. Если мы просто передадим ключ, курьер сможет расшифровать наше послание. Алгоритм Диффи-Хеллмана как раз и призван решить эту задачу.

Алиса и Боб хотят использовать общий ключ для шифрования переписки. Рассмотрим детально этот процесс, используя краски вместо чисел (рисунок 1):

1. Алиса и Боб выбрали общую краску.
2. Алиса и Боб выбрали по одной секретной краске.
3. Алиса и Боб смешали общую и секретную краску.
4. Алиса и Боб обменялись получившимися смешанными красками.
5. Алиса смешала полученную смешанную краску от Боба со своей секретной краской.
6. Боб смешал полученную смешанную краску от Алисы со своей секретной краской.
7. Теперь у Алисы и Боба есть общая секретная краска.

Суть смешения красок в том, что один и тот же цвет можно получить смешением различных цветов и чтобы подобрать перебором настоящие исходные цвета нужно совершить огромное количество вычислений.

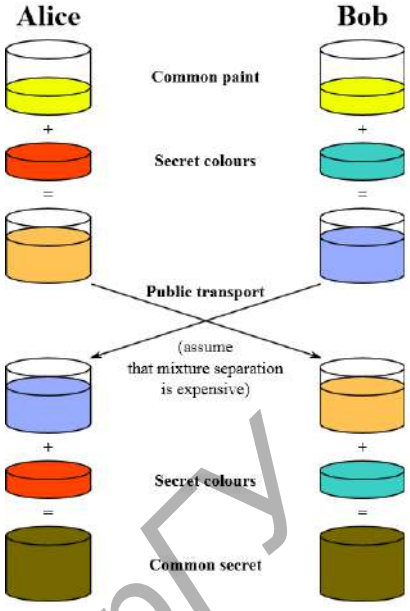

Рисунок 1 — Визуализация алгоритма Диффи-Хеллмана [4]

В реальной задаче вместо цветов используют функцию остатка от деления, так как, так же, как с цветами, функция результат z в выражении $x \ mod \ y = z$, можно получить используя совершенно различные целые числа x и y. Задача нахождения z, является задачей линейного логорифмирования, которую, на данный момент, человечество не научилось решать эффективно, поэтому данный алгоритм является таким надежным.

Заключение. В данной статье мы рассмотрели метод защиты от перехвата сетевого трафика. Данную задачу позволяет решить алгоритм Диффи-Хеллмана. В настоящее время он является достаточно надежным, т. к. еще не появились программные комплексы, имеющие высокую вычислительную мощность.

Список цитируемых источников

1. *Таненбаум, Э.* Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стеен. — СПб. : Питер, 2003. — 877 с
2. *Хант, К.*TCP/IP. Сетевое администрирование : пер. с англ. / К. Хант. — 3-е изд. — СПб : СимволПлюс, 2004. — 816 с.
3. *Олифер, В. Г.* Компьютерные сети. Принципы, технологии, протоколы / В. Г. Олифер, Н. А. Олифер — 4-е изд. — СПб. : Питер, 2014. — 944 с.
1. *Шнайер, Б.* Прикладная криптография. Протоколы, алгоритмы и исходный код на С / Б. Шнайер. — М. : Вильямс, 2016. — 842 с.

УДК 004.65

К. Ю. Матусевич, О. Д. Кравчук

Учреждение образования «Барановичский государственный университет», Барановичи, Республика Беларусь

СОЗДАНИЕ АРИФМЕТИЧЕСКОГО ТРЕНАЖЕРА

Введение. В современном мире каждый человек применяет базовые правила арифметики, даже не задумываясь об этом. Поэтому развитие этих способностей необходимо начинать как можно раньше, с самого детства человека, так как в условиях нашего мира невозможно будет догнать то, что было упущено раньше. Тренажеры могут быть использованы для дополнительной работы с первоклассниками учителями и родителями в классе и дома как для индивидуальной, так и коллективной подготовки [1]. Они способствуют автоматизации вычислительных навыков у ребенка, отработке умений складывать, вычитать, сравнивать и решать простые задачи.

Целью исследования является создание арифметического тренажера, с возможностью выбора количества примеров, разрядности и действия, в котором пропущены разные числа — не обязательно результат.

Объектом исследования являются возможности, компоненты, методы и способы создания приложений, использующих Windows Forms.

Объектом исследования является оконное развивающее приложение для тренировки устного счета.

Основная часть. Оконное приложение — это класс приложений, использующих для взаимодействия с пользователем элементы графического пользовательского интерфейса, т. е. объекты типа: окна, кнопки, поля ввода, элементы контроля и многие другие.

В качестве среды разработки использован Embarcadero C++ Builder, который предоставляет широкие возможности для создания оконных приложений.