Математические структуры и моделирование
2018. № 4(48). С. 99–112

УДК 004.7:004.056
DOI: 10.25513/2222-8772.2018.4.99-112

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ МОДЕЛИРОВАНИЯ СЕТИ И ИМИТАЦИИ АТАК НА КОМПЬЮТЕРНУЮ СЕТЬ

А.В. Баженов
студент, e-mail: dr.bazhenoff2017@yandex.ru
А.К. Гуц
д.ф.-м.н., профессор, e-mail: guts@omsu.ru

Омский государственный университет им. Ф.М. Достоевского, Омск, Россия

Аннотация. В статье представлено программное приложение, позволяющее моделировать компьютерные сети и атаки на них.

Ключевые слова: программное приложение, моделирование, сетевые атаки, smurfing.

1. Введение

Целью данной работы является разработка программного обеспечения для моделирования различных сетевых атак. Такое приложение полезно в первую очередь для обучения студентов и администраторов, которые по долгу службы обязаны обеспечивать безопасность информационных ресурсов.

Для достижения цели требуется решение следующих задач:

1. Анализ существующих видов сетевых атак.
2. Разработка программных библиотек для имитации основных сетевых устройств.
3. Разработка пользовательского интерфейса для работы с программной библиотекой.
4. Тестирование разработанного программного комплекса с помощью демонстрации реализации сетевой атаки (в статье представлена имитация атаки Smurfing).

Данное исследование продолжает разработки по моделированию атак на компьютерные сети посредством создания специализированного программного обеспечения, начатые в [1]. Приложение не предусматривает демонстрацию защиты от сетевых атак.

2. Основные понятия

2.1. Сетевой протокол

Сетевой протокол — это набор правил и соглашений, который определяет единообразный способ передачи информации и обработки ошибок при взаимо-

действии и позволяет осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами [2].

Каждый сетевой протокол работает на определённом уровне модели OSI. Протоколы высшего уровня строятся поверх протоколов более низкого уровня.

2.2. Уязвимость

В компьютерной безопасности термин «уязвимость» (англ. vulnerability) используется для обозначения недостатка в системе, используя который можно намеренно нарушить её целостность и вызвать неправильную работу. Уязвимость может быть результатом ошибок программирования, недостатков, допущенных при проектировании системы, ненадёжных паролей, вирусов и других вредоносных программ, скриптовых и SQL-инъекций. Некоторые уязвимости известны только теоретически, другие же активно используются и имеют известные эксплойты.

Метод информирования об уязвимостях является одним из пунктов спора в сообществе компьютерной безопасности. Некоторые специалисты отстаивают немедленное полное раскрытие информации об уязвимостях, как только они найдены. Другие советуют сообщать об уязвимостях только тем пользователям, которые подвергаются наибольшему риску, а полную информацию публиковать лишь после задержки или не публиковать совсем. Такие задержки могут позволить тем, кто был извещён, исправить ошибку при помощи разработки и применения патчей, но также могут и увеличивать риск для тех, кто не посвящён в детали.

Обычно уязвимость позволяет атакующему «обмануть» приложение — заставить его совершить действие, на которое у того не должно быть прав. Это делается путём внедрения каким-либо образом в программу данных или кода в такие места, что программа воспримет их как «свои». Некоторые уязвимости появляются из-за недостаточной проверки данных, вводимых пользователем, и позволяют вставить в интерпретируемый код произвольные команды (SQL-инъекция, XSS, SiXSS). Другие уязвимости появляются из-за более сложных проблем, таких как запись данных в буфер без проверки его границ (переполнение буфера) [3]. Атаки, рассмотренные в данной работе, как правило, используют недостатки сетевых протоколов.

3. Сетевые атаки

3.1. Метод оценки уязвимости

При рассмотрении атак будет использоваться система оценок CVSSv3.

Стандарт Common Vulnerability Scoring System был разработан группой экспертов по безопасности National Infrastructure Advisory Council. В эту группу вошли эксперты из различных организаций, таких как CERT/CC, Cisco, DHS/MITRE, eBay, IBM Internet Security Systems, Microsoft, Qualys, Symantec.

CVSS предлагает простой инструментарий для расчёта числового показателя по десятибалльной шкале, который позволяет специалистам по безопасности оперативно принимать решение о том, как реагировать на ту или иную уязвимость. Чем выше значение метрики, тем более оперативная реакция требуется.

Использование метрик CVSS для оценки уязвимостей закреплено в стандартах PCI DSS и СТО БР ИББС [4].

3.2. Фрагментация данных

При передаче пакета данных протокола IP по сети может осуществляться деление этого пакета на несколько фрагментов. Впоследствии, при достижении адресата, пакет восстанавливается из этих фрагментов. Злоумышленник может инициировать посылку большого числа фрагментов, что приводит к переполнению программных буферов на приёмной стороне и, в ряде случаев, к аварийному завершению работы системы [5].

Оценка CVSSv3: 7.5 [6].

3.3. Ping flooding

Данная атака требует от злоумышленника доступа к быстрым каналам в интернете.

Программа ping посылает ICMP-пакет типа ECHO REQUEST, выставляя в нём время и его идентификатор. Ядро машины-получателя отвечает на подобный запрос пакетом ICMP ECHO REPLY. Получив его, ping выдаёт скорость прохождения пакета.

При стандартном режиме работы пакеты высылаются через некоторые промежутки времени, практически не нагружая сеть. Но в «агрессивном» режиме поток ICMP echo request/reply-пакетов может вызвать перегрузку небольшой линии, лишив её способности передавать полезную информацию [5].

Оценка CVSSv3: 5.9 [6].

3.4. Smurfing

Атака заключается в передаче в сеть широковещательных ICMP запросов от имени компьютера-жертвы. В результате компьютеры, принявшие такие широковещательные пакеты, отвечают компьютеру-жертве, что приводит к существенному снижению пропускной способности канала связи и, в ряде случаев, к полной изоляции атакуемой сети. Эта атака исключительно эффективна и широко распространена [5].

Оценка CVSSv3: 7.5 [6].

3.5. DNS Cache Poisoning

Для осуществления атаки атакующий использует уязвимость в конфигурации DNS. Если сервер не проверяет ответы DNS на корректность, чтобы убедиться в их авторитетном источнике, он будет кэшировать некорректные

ответы локально и использовать их для ответов на запросы других пользователей, пославших такие же запросы.

Данная техника может использоваться для того, чтобы перенаправить клиентов на другой сайт по выбору атакующего. Например, при помощи спуфинга можно направить клиента на DNS-сервер, который выдаст заведомо неверный IP-адрес сайта и таким образом направит адресата на сервер, контролируемый злоумышленником [5].

Оценка CVSSv3: 9.0 [6].

3.6. Навязывание пакетов

Злоумышленник отправляет в сеть пакеты с ложным обратным адресом. С помощью этой атаки злоумышленник может переключать на свой компьютер соединения, установленные между другими компьютерами. При этом права доступа злоумышленника становятся равными правам того пользователя, чьё соединение с сервером было переключено на компьютер злоумышленника [5].

Оценка CVSSv3: 9.0 [6].

3.7. SYN Flood

Обычно, когда клиент пытается установить TCP-соединение с сервером, они обмениваются сообщениями по следующей схеме:

1. Клиент запрашивает соединение, посылая SYN-пакет.
2. Сервер подтверждает полученный запрос, отвечая SYN-ACK-пакетом.
3. Клиент, подтверждая, что соединение установлено, отправляет ACK-пакет.

Данная атака основывается на отправке множества SYN-пакетов с разных IP-адресов, которые могут быть даже не закреплены за кем-то или закреплены за устройствами, не принимающими участие в атаке.

Для каждого полученного SYN-пакета сервер выделяет место в буфере до тех пор, пока не получит ACK-ответ. Таким образом, атака может заполнить буфер сервера так, что другие попытки установить с ним соединение не увенчаются успехом из-за отсутствия свободного места в буфере.

Оценка CVSSv3: 5.9 [6].

3.8. UDP Flood

Данная атака основывается на отправке большого количества UDP-пакетов на случайные порты удалённого хоста. Так как UDP-протокол не требует установления соединения, хост-жертва сразу будет проверять наличие приложений, на порты которых отправлены UDP-пакеты. Так как выбор портов случаен, наверняка, большая часть из них будет не привязана к какому-то приложению, и хост будет отправлять ICMP Destination Unreachable.

Таким образом, жертва будет занята отправкой ICMP-пакетов, а не обработкой реальных запросов.

Оценка CVSSv3: 7.0 [6].

3.9. Land Attack

Атака заключается в отправке хосту пакетов с идентичными адресами отправителя и получателя. Может возникнуть бесконечная петля обращений хоста жертвы к самой себе.

Оценка CVSSv3: 5.9 [6].

3.10. ARP Spoofing

Данная атака основывается на проблеме аутентификации в ARP-протоколе. Любой хост в сети может отправить ARP-пакет, который ассоциирует любой IP-адрес с MAC-адресом злоумышленника, таким образом, хост-жертва будет думать, что отправляет пакеты одному хосту, но на самом деле эти пакеты будет получать хост с заданным MAC-адресом.

Оценка CVSSv3: 8.8. [6].

3.11. MAC Flooding

Данная атака основывается на отправке коммутатору большого количества Ethernet-кадров, которые содержат различные MAC-адреса отправителя. Свитч заносит все эти MAC-адреса в свою таблицу, тем самым забивая её. Если таблица забита, коммутатор работает в режиме хаба: отправляет пакет на все интерфейсы, кроме того, откуда пришёл пакет. Таким образом можно реализовывать другие атаки в рамках целых сетей, к которым подключён свитч.

Оценка CVSSv3: 8.6.[6].

3.12. IP Spoofing

Данная атака заключается в изменении адреса отправителя в заголовке IP-пакета. Хост-жертва будет думать, что получает пакеты от одного хоста, а на самом деле — совершенно от другого. Данная атака позволяет обойти защиту по белому списку, анонимизировать злоумышленника или перенаправить трафик в сети.

Оценка CVSSv3: 7.5. [6].

3.13. MAC Spoofing

Эта атака аналогична атаке IP Spoofing, кроме того, что используется на более низком уровне модели OSI.

Оценка CVSSv3: 7.5. [6].

3.14. Sniffing

Данная атака основывается на специальном программном обеспечении для перехвата и анализа сетевого трафика. Сниффер может анализировать весь трафик, который проходит через сетевую карту. Внутри одного сегмента сети Ethernet-трафик рассылается всем машинам, таким образом, в такой сети весь трафик можно прослушивать.

Данную атаку можно легко сочетать с другими атаками, например, IP Spoofing и MAC Spoofing.

Оценка CVSSv3: 8.6 [6].

4. Разработка программной библиотеки для имитации основных сетевых устройств

Для разработки программной библиотеки был выбран язык $C\sharp$, так как он позволяет сконцентрироваться на абстракциях, а не на деталях реализации.

Приложение[1] работает под Windows с Microsoft.Net Framework 4.5.3+ и Linux (MacOS) с последней версией Mono Framework. Операционная система 64-битная, но запускается и на x86. Файл запуска программы UserInterface.exe.

4.1. Разработка механизма передачи сообщений

Для передачи сообщений было решено использовать протокол UDP, который позволяет отправлять данные без предварительного открытия соединения. Был реализован класс UdpClient, который содержит в себе экземпляр класса System.Net.Sockets.UdpClient и хранилище пар IP:Port для всех, кто отправлял сообщения по адресу нашего UdpClient:

```
private readonly System.Net.Sockets.UdpClient client;
protected readonly ISet<IPEndPoint> _connections = new
HashSet<IPEndPoint>();
```

У данного класса были реализованы методы для отправки сообщения по IP-адресу, Broadcast отправки, а также получения сообщения:

```
public void Send(string ipAddress, string message) {
var port = ipAddress.HashCode();
var endpoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), port);
var data = Encoding.UTF8.GetBytes(message);
client.Send(data, data.Length, endpoint);
}
public void SendBroadcast(string message) {
foreach (var endpoint in _connections)
{
var data = Encoding.UTF8.GetBytes(message);
```

[1]Приложение доступно по адресу: http://fkn.univer.omsk.su/teaching/Simulator/Bazhenov.zip

```
client.Send(data, data.Length, endpoint);
}
}
public async Task<Received> Receive() {
var result = await client.ReceiveAsync();
_connections.Add(result.RemoteEndPoint);
return new Received
{
Message = Encoding.UTF8.GetString(result.Buffer, 0,
result.Buffer.Length),
Sender = result.RemoteEndPoint
};
} [7].
```

4.2. Разработка модели физического интерфейса сетевого устройства

Был разработан класс NetworkInterface, который наследует функционал UdpClient и содержит поля, хранящие его IP-адрес и тип устройства, чей это интерфейс.

4.3. Разработка моделей компьютера и компьютера злоумышленника

Указанные классы отличаются лишь тем, что компьютер злоумышленника может отправлять сообщения, редактируя адрес отправителя, поэтому описания компьютера будет достаточно.

Класс компьютера определяется тремя параметрами: IP-адрес, IP-адрес сетевого устройства, к которому он подключён, и UdpClient для отправки сообщений.

При проектировании был принят унифицированный формат сообщений: “IP_источника/IP_назначения/сообщение/Тип_Устройства_Отправителя”.

При создании экземпляра класса Computer или HackerComputer необходимо указать адрес шлюза по умолчанию, это может быть адрес интерфейса маршрутизатора или коммутатора, на этот адрес будет отправлен пакет SYN и ожидаться ACK-подтверждение.

Компьютер умеет принимать и обрабатывать сообщения, а также ведёт журнал, в котором отображается, какие пакеты и от кого он получает. Для обработки сообщений было решено использовать асинхронную модель: создаётся поток, который обрабатывает пришедшие пакеты.

4.4. Разработка модели маршрутизатора

Был разработан класс Router, который имеет набор экземпляров класса NetworkInterface, таблицу маршрутизации и очередь для приходящих сообщений.

Для каждого интерфейса запускается отдельный поток, который ожидает сообщения.

Когда сообщение получено, этот поток его обрабатывает:

1. Если сообщение SYN, то отправляется ACK-ответ, а пара адрес отправителя – интерфейс заносится в таблицу маршрутизации.
2. Если сообщение DIST (означает, что оно содержит информацию о сетях, отправленную другим сетевым устройством), то роутер добавляет пары сеть – интерфейс в свою таблицу маршрутизации.
3. Если это широковещательное сообщение, то оно отправляется на все интерфейсы маршрутизатора.
4. Если сообщение не одно из выше перечисленных, то оно заносится в очередь.

Отдельным потоком работает функция маршрутизации: ждёт, пока в очереди появится сообщение, далее ищет адрес назначения в таблице маршрутизации и, если находит, — отправляет на нужный интерфейс, причём если есть несколько подходящих интерфейсов, то сообщение отправляется на любой, не являющийся роутером, чтобы избежать петель маршрутизации.

```
    private void StartRouting() =>
    Task.Factory.StartNew(async () => {
    while (true) {
    var msg = await queue.ReceiveAsync();
    var target = msg.Split('/')[1];
    if (routingTable.ContainsKey(target)) {
    var I = routingTable[target];
    var i = interfaces.OrderBy(x => x.Type ==
    NetworkDeviceType.ROUTER).First();
    interface.Send(msg);
    }
}
    }); [3]
```

Был реализован функционал рассылки таблицы маршрутизации на все интерфейсы, к которым подключён другой роутер. Сообщение с рассылаемой таблицей маршрутизации в адресе назначения имеет широковещательный адрес. Для рассылки используется отдельный поток, который повторяет её каждые 0,1 секунды, чтобы сходимость в сети была быстрая.

```
    private void StartTableDistribution() =>
    Task.Factory.StartNew(() => {
    while (true) {
    Interfaces.Where(i => i.Type ==
    NetworkDeviceType.ROUTER).ToList()
    .ForEach(i => {
    var serializedTable = Serialize(routingTable);
    var interfaces = string.Join("#
```

```
Interfaces.Select(x => x.IpAddress));
var routingInfo = Merge(serializedTable,
interfaces);
i.Send($"{i.IpAddress}/255.255.255.255/DIST
/{NetworkDeviceType.ROUTER}/
{routingInfo}");
});
Thread.Sleep(100);
}
});
```

4.5. Разработка модели коммутатора

Был разработан класс Switch, работающий по принципу, схожему с Router. Отличие состоит в том, что коммутатор не получает рассылку таблицы маршрутизации от роутеров и, если пришедший пакет имеет адрес назначения, которого нету в его таблице маршрутизации, то свитч отправит этот пакет на все порты, кроме того, с которого он пришёл.

Коммутатор и маршрутизатор имеют метод Connect, который имитирует подключение между своим портом и указанным портом другого сетевого устройства. Этот метод использует двойное рукопожатие для установления соединения.

```
public void Connect(string sourceIp, string destinationIp) {
var srcInterface = Interfaces.First(i => i.IpAddress == sourceIp);
srcInterface.Send(destinationIp, $"{sourceIp/destinationIp}/
{PacketType.HANDSHAKE.SYN}/{NetworkDeviceType.Switch
}"
);
}
```

5. Разработка модели пользовательского интерфейса

Так как для разработки программной библиотеки был использован язык $C\sharp$, то для реализации пользовательского интерфейса лучше всего подойдёт Windows Forms [3] фреймворк.

5.1. Разработка основного окна

Основное окно должно включать функционал для создания окон остальных сетевых элементов. Для создания компьютера и компьютера-хакера: поля для ввода собственного IP-адреса и IP-адреса маршрута по умолчанию, а также кнопка. Для маршрутизатора и коммутатора — окно для ввода IP-адресов интерфейсов (см. рис. 1).

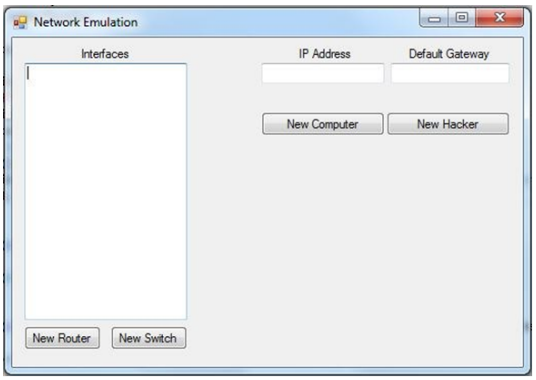

Рис. 1. Основное окно

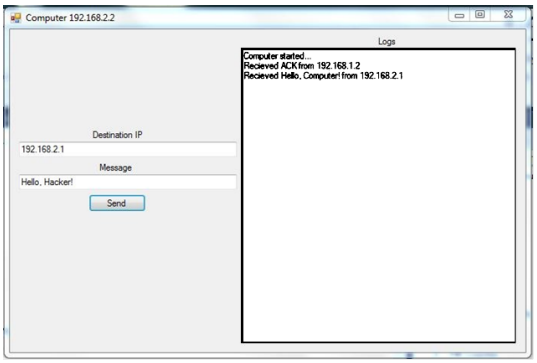

Рис. 2. Окно компьютера

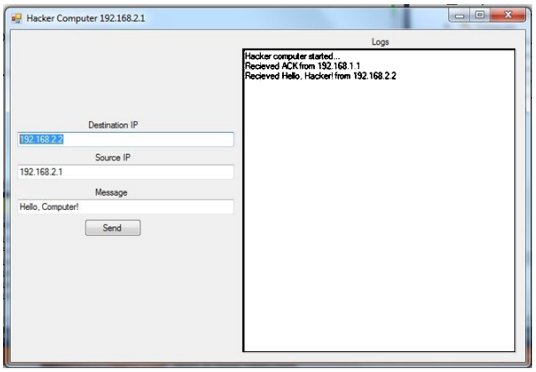

Рис. 3. Окно компьютера-злоумышленника

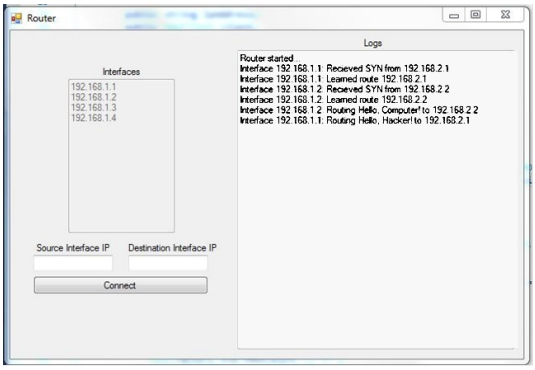

Рис. 4. Окно маршрутизатора

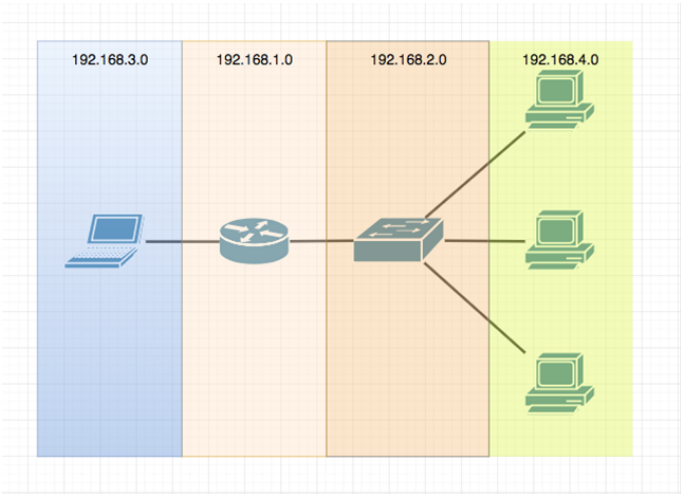

Рис. 5. Топология сети

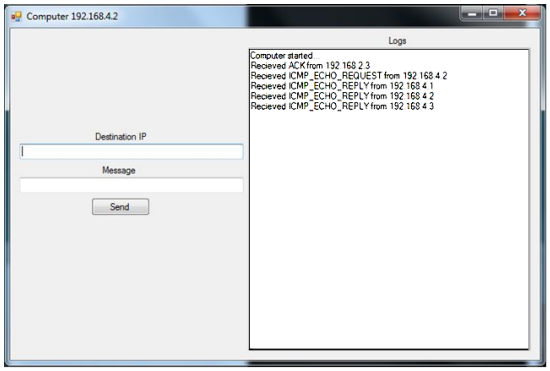

Рис. 6. Окно логгирование у компьютера с IP адресом 192.168.4.2

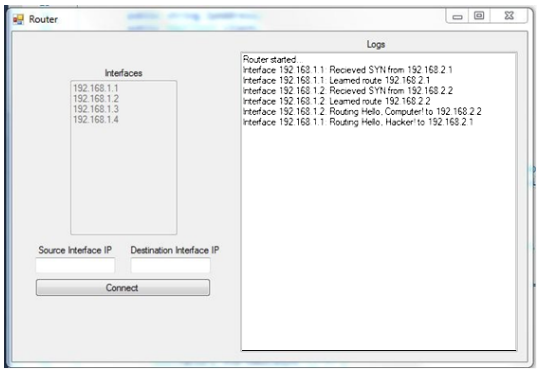

Рис. 7. Окно маршрутизатора после атаки

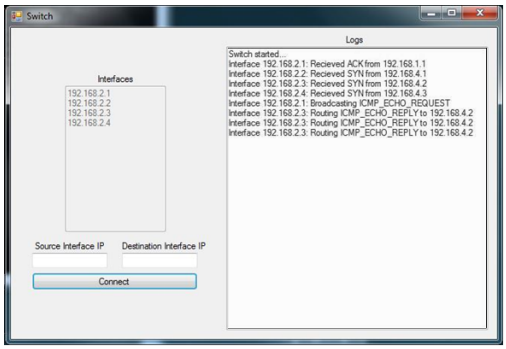

Рис. 8. Окно коммутатора после атаки

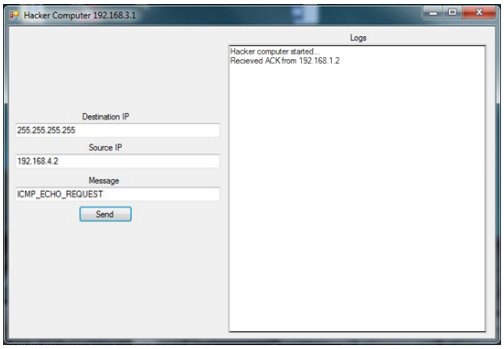

Рис. 9. Окно компьютера-злоумышленника после атаки

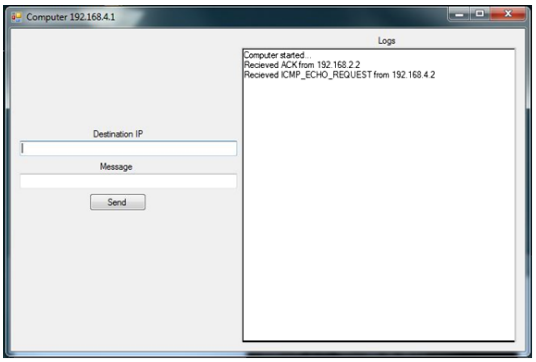

Рис. 10. Окно компьютера 192.168.4.1 после атаки

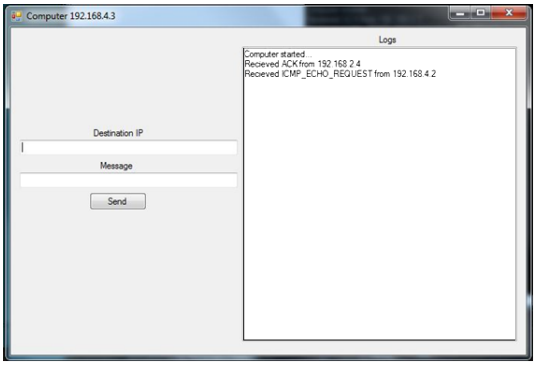

Рис. 11. Окно компьютера 192.168.4.3 после атаки

5.2. Разработка окна компьютера

Окно компьютера должно иметь функционал для отправки сообщения по IP-адресу: поля для ввода IPадреса назначения и сообщения, а также кнопка “Send”. Для удобства и понимания того, что происходит с компьютером, было создано окно с логгированием событий (см. рис. 2).

5.3. Разработка модели окна компьютера-злоумышленника

Компьютер злоумышленника должен иметь тот же функционал, что и обычный компьютер, но должен иметь возможность отправлять сообщения с произвольного IP-адреса (см. рис. 3).

5.4. Разработка моделей окон маршрутизатора и коммутатора

Для функционирования сети нужна возможность соединять сетевые устройства друг с другом. Для этого предусмотрены два окна ввода для IP-адресов: собственного интерфейса и интерфейса другого сетевого устройства, а также кнопка “Connect”.

Также предусмотрены окна для демонстрации списка IP-адресов собственных интерфейсов и отображения системных сообщений маршрутизатора (см. рис. 4).

Окно коммутатора внешне не отличается от окна маршрутизатора.

6. Демонстрация работоспособности программного обеспечения на примере простейшей сетевой атаки

В качестве атаки используем Smurfing, так как она довольно проста в реализации, но, тем не менее, отлично покажет работоспособность программного обеспечения на примере реальной сети. Будем имитировать сеть со следующей топологией (см. рис. 5).

Шаг 1. Создать маршрутизатор с двумя интерфейсами, которые имеют IP-адреса 192.168.1.1 и 192.168.1.2.

Шаг 2. Создать коммутатор с четырьмя интерфейсами, которые имеют IP-адреса 192.168.2.1, 192.168.2.2, 192.168.2.3, 192.168.2.4.

Шаг 3. Подключить коммутатор интерфейсом с IP адресом 192.168.2.1 к интерфейсу маршрутизатора с IP-адресом 192.168.1.1.

Шаг 4. Создать компьютер злоумышленника, используя IP-адрес 192.168.3.1 и шлюз по умолчанию 192.168.1.2.

Шаг 5. Создать три компьютера с IP-адресами 192.168.4.1, 192.168.4.2, 192.168.4.3 и шлюзами 192.168.2.2, 192.168.2.3, 192.168.2.4 соответственно.

Шаг 6. Отправить сообщение ICMP_ECHO_REQUEST с адресом отправителя 192.168.4.2 и широковещательным адресом назначения 255.255.255.255 с компьютера злоумышленника.

Шаг 7. Теперь в окне логгирования у компьютера с IP-адресом 192.168.4.2 три сообщения ICMP ECHO REPLY (см. рис. 6), это значит, что все компьютеры в сети получили ICMP ECHO REQUEST сообщение и отправили ICMP ECHO REPLY по IP-адресу отправителя, которым был указан хост 192.168.4.2.

Окно маршрутизатора после атаки представлено на рис. 7, а окно коммутатора после атаки — на рис. 8.

Окно компьютера-злоумышленника после атаки представлено на рис. 9, а окно компьютера 192.168.4.1 после атаки — на рис. 10.

Окно компьютера 192.168.4.3 после атаки — см. рис. 11.

Заключение

В этой работе были описаны некоторые атаки на уязвимости сетевых протоколов. Знание принципов этих атак поможет администратору избежать значительных моральных и финансовых потерь.

Представлена программная библиотека на языке $C^\sharp$, которая позволяет моделировать реальные масштабные сети. Она содержит классы для имитации работы настоящих компьютеров, маршрутизаторов и коммутаторов как поодиночке, так и в связке друг с другом.

Был разработан пользовательский интерфейс, который позволяет любому человеку, не вникая в детали реализации, построить сеть любых размеров. Был смоделирован прототип реальной сети, на который успешно проведена атака Smurfing.

Разработанное программное обеспечение не накладывает на программиста никаких ограничений и может быть использовано для дальнейшей реализации сетевых протоколов и имитации различных атак на компьютерные сети.

Другой подход по моделированию атак на компьютерные сети разработан в [8]. Был разработан прототип, т. е. черновая реализация мультиагентного компьютерного симулятора атак. Каждый из компонентов симулятора атаки, а точнее атакующий, атакуемая компьютерная сеть и трафик (сеансы между хостами, общий трафик — суммарный поток между компонентами симулятора) был создан как агент многоагентной системы. Каждый агент взаимодействует с другими агентами, средой, которая воспринимается и, возможно, модифицируется агентами, и пользователь взаимодействует с агентами через свой интерфейс.

Ознакомление администраторов с атаками на сети возможно и без создания специального программного обеспечения. Например, только с использованием технологии виртуальных машин для имитации сетевых соединений [9, Приложение 1].

Очевидно, что крайне желательно иметь программное приложение, которое демонстрирует не только атаки, но и способы защиты от них. Последнее исследуется, например, в работах [10, 11].

Литература

1. Гуц А.К., Эннс Е.П. Программа, моделирующая компьютерную сеть и сетевые атаки // Математические структуры и моделирование. 2017. № 3(43). С. 139–149.
2. Википедия. Протокол передачи данных. URL: `https://ru.wikipedia.org/wiki/Протокол_передачи_данных` (дата обращения: 27.11.2017).
3. Cleary S. Concurrency in $C^\sharp$ Cookbok. O'Reilly Media, 2014. 207 p.
4. Positive Technologies. Оценка уязвимостей CVSS 3.0. URL: `https://habr.com/company/pt/blog/266485/` (дата обращения: 16.03.2018).
5. Hauzer. Сетевые атаки и кое-что ещё. URL: `http://forum.is.ua/showthread.php?t=16215` (дата обращения: 19.12.2017).
6. Common Vulnerability Scoring System Version 3.0 Calculator. URL: `https://www.first.org/cvss/calculator/3.0` (дата обращения: 16.03.2018).
7. MSDN. Task Class (System.Threading.Tasks). URL: `https://msdn.microsoft.com/en-us/library/system.threading.tasks.task(v=vs.110).aspx` (дата обращения: 26.02.2018).
8. Gorodetski V., Kotenko I. Attacks against Computer Network: Formal Grammarbased Framework and Simulation Tool // Lecture Notes in Computer Science. 2002. V. 2516. P. 219–238.
9. Синадский Н.И., Хорьков Д.А. Защита информации в компьютерных сетях: учеб. пособие. Екатеринбург : УрГУ, 2008. 225 с.
10. Котенко И.В., Шоров А.В. Имитационное моделирование механизмов защиты компьютерных сетей от инфраструктурных атак на основе подхода «нервная система сети» // Труды СПИИРАН. 2012. Вып. 3(22). С. 45–70.
11. Бекенева Я.А., Шипилов Н.Н., Борисенко К.А., Шоров А.В. Моделирование DDOS-атак и механизмов защиты от них // Известия СПбГЭТУ «ЛЭТИ». 2015. № 3. С. 32–39.

SOFTWARE FOR MODELING OF NETWORK AND SIMULATION OF COMPUTER NETWORK ATTACKS

A.V. Bazhenov
Student, e-mail: dr.bazhenoff2017@yandex.ru
A.K. Guts
Dr.Sc. (Phys.-Math.), Professor, e-mail: guts@omsu.ru

Dostoevsky Omsk State University, Omsk, Russia

Abstract. The article presents a software application that simulates different computer networks and attacks on these networks.

Keywords: software application, modeling, network attack, smurfing.

Дата поступления в редакцию: 17.11.2018