МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»   
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

**ARP-спуфинг и методы борьбы с ним**

Курсовая работа

по дисциплине «Компьютерные сети»

студента 1 курса группы ИВТ-б-о-222(2)

Чудопалова Богдана Андреевича

Направления подготовки 09.03.01 «Информатика и вычислительная техника»

|  |  |  |
| --- | --- | --- |
| Научный руководитель  старший преподаватель кафедры компьютерной инженерии и моделирования | \_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | Тимофеева С.В. |

Симферополь, 2023

**РЕФЕРАТ**

Чудопалов Б.А. ARP-спуфинг и методы борьбы с ним//Курсовая работа по направлению подготовки: 09.03.04 Информатика и вычислительная техника / Кафедра компьютерной инженерии и моделирования Физико-технического института ФГАОУ “Крымский федеральный университет им. В.И. Вернандского” – Симферополь, 2023

Объект разработки – ARP-спуфинг. Методы обнаружения и предотвращения.

Цель работы - изучение процесса ARP-спуфинга и методов борьбы с ним, а также исследование данного процесса на практике.

ПРОТОКОЛ ARP, НЕДОСТАТКИ ПРОТОКОЛА ARP, ARP-СПУФИНГ, LINUX, ARP-ПАКЕТ, PYTHON, БИБЛИОТЕКА SCAPY, WIRESHARK.

**СОДЕРЖАНИЕ**

Введение……………………………………………………………………………...3

1. Теоретическая часть………………………………………………………………4

1.1 Принцип работы протокола ARP и его недостатки…………………….4

1.2 ARP-спуфинг…………………………………………………,………….10

1.3 Методы обнаружения и борьбы с ARP-спуфингом…………………...14

2. Практическая часть……………………………………………………………...18

2.1 Создание сканера сети…………………………………………………..18

2.2 Написание скрипта для ARP-спуфинга………………………………..23

Заключение………………………………………………………………………...26

Список использованных источников……………………………………………..27

**ВВЕДЕНИЕ**

Тема курсовой работы “ARP-спуфинг и методы борьбы с ним” подразумевает подробное изучение процесса ARP-спуфинга и способов защиты от него.

В ходе работы будет рассмотрен протокол ARP принцип его работы и уязвимости, которые есть у него. Также будет рассмотрен процесс ARP-спуфинга, утилиты, которые для него необходимы. В конце работы будут рассмотрены методы защиты от ARP-спуфинга.

В качестве практической части работы, на языке программирования Python будут написаны скрипты с помощью которых будет проведен ARP-спуфинг. После этого будут применены методы защиты от ARP-спуфинга, которые были рассмотрены в теоретической части.

Актуальность темы “ARP-спуфинг и методы борьбы с ним” обусловлена тем, что протокол ARP используется в большом количестве сетей, вследствие этого ARP-спуфинг является очень серьезным инструментом в руках злоумышленников, с помощью которого можно украсть большое количество личных данных пользователя. Таким образом данная тема является актуальной и интересной проблемой, которую необходимо рассмотреть и изучить.

Целью данной работы является изучение процесса ARP-спуфинга и методов борьбы с ним, а также исследование данного процесса на практике.

Для достижения данной цели необходимо выполнить следующие задачи:

1. Изучить теоретические основы работы протокола ARP.
2. Изучить процесс ARP-спуфинга.
3. Изучить и проанализировать методы борьбы с ARP-спуфингом.
4. Разработать практическую часть, которая включает в себя разработку скрипта для ARP-спуфинга.
5. **ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**1.1 Принцип работы протокола ARP и его недостатки**

Чтобы говорить об ARP-спуфинге необходимо узнать принцип работы протокола ARP.

Адресация в сети Internet представляет собой 32-битовую последовательность 0 и 1, называющихся IP-адресами. Но непосредственно связь между двумя устройствами в сети осуществляется по адресам канального уровня (MAC-адресам).  
 Так вот, для определения соответствия между логическим адресом сетевого уровня (IP) и физическим адресом устройства (MAC) используется описанный в [RFC 826](http://tools.ietf.org/html/rfc826) протокол ARP (Address Resolution Protocol, протокол разрешения адресов).

Протокол имеет буферную память (ARP-таблицу), в которой хранятся пары адресов (IP-адрес, MAC-адрес) с целью уменьшения количества посылаемых запросов, следовательно, экономии трафика и ресурсов. **Пример ARP-таблицы (рис. 1), сделать это можно с помощью команды “arp -a”.**

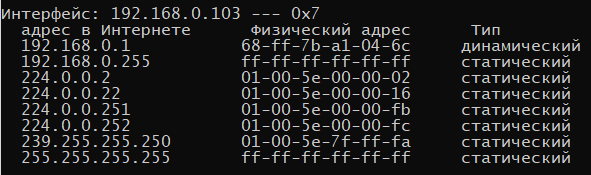
****

Рис. 1. ARP-таблица ПК.

Записи ARP-таблицы бывают двух вид видов: статические и динамические. Статические добавляются самим пользователем, динамические же – создаются и удаляются автоматически. При этом в ARP-таблице всегда хранится широковещательный физический адрес FF:FF:FF:FF:FF:FF (в Linux и Windows).

После добавления записи в таблицу ей присваивается таймер. При этом, если запись не используется первые 2 минуты, то удаляется, а если используется, то время ее жизни продлевается еще на 2 минуты, при этом максимально – 10 минут для Windows и Linux, Cisco IOS – 4 часа, после чего производится новый широковещательный ARP-запрос.

**А вот как происходит определение маршрута с участием протокола ARP.**

Пусть отправитель A и получатель B имеют свои адреса с указанием маски подсети.

1. Если адреса находятся в одной подсети, то вызывается протокол ARP и определяется физический адрес получателя, после чего IP-пакет инкапсулируется в кадр канального уровня и отправляется по указанному физическому адресу, соответствующему IP-адресу назначения.
2. Если нет – начинается просмотр таблицы в поисках прямого маршрута.
3. Если маршрут найден, то вызывается протокол ARP и определяется физический адрес соответствующего маршрутизатора, после чего пакет инкапсулируется в кадр канального уровня и отправляется по указанному физическому адресу.
4. В противном случае, вызывается протокол ARP и определяется физический адрес маршрутизатора по умолчанию, после чего пакет инкапсулируется в кадр канального уровня и отправляется по указанному физическому адресу.

**Процесс работы протокола ARP**

Устройство, которому необходимо узнать идентификатор канальной среды другого устройства, создает пакет **ARP Request**. Данный запрос содержит в себе IP-адрес устройства для которого необходимо узнать идентификатор канального среды (Target), а также данные канальной среды (в данном случае MAC-адрес) и IP-адрес устройства, которое данный запрос сформировало (Source).

Пакет ARP Request инкапсулируется в фрейм. В качестве MAC-адреса источника выступает сам источник, в качестве MAC-адреса назначения используется broadcast адрес (FF:FF:FF:FF:FF:FF).

Вот как выглядит ARP Request если захватить его с помощью Wireshark (рис. 2).

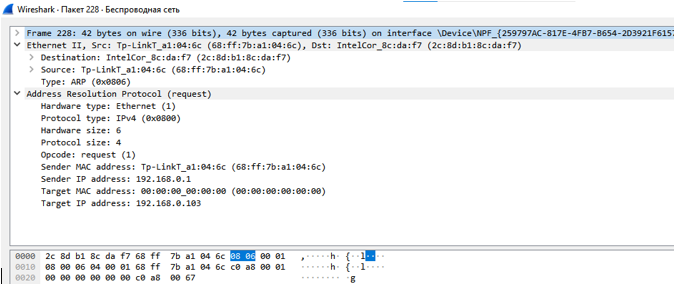


Рис. 2. ARP Request.

Broadcast адрес используется для того чтобы все узлы в канальной среде получили данный фрейм и обработали инкапсулированный внутри пакет. Все устройства, за исключением адресата назначения, поймут, что данный пакет не предназначается для них и попросту отбросят его. Устройство назначения обработает пакет и в ответ отправит **ARP Reply** адресату источника. Внутри ARP Reply будет содержаться как раз-таки MAC-адрес адресата назначения.

Вот как выглядит ARP Reply если захватить его с помощью Wireshark (рис. 3).

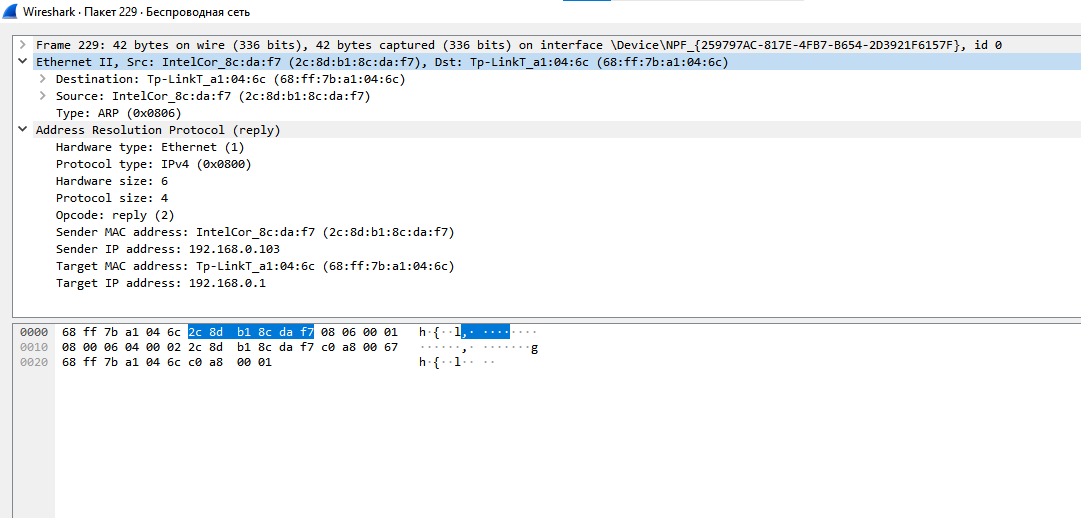


Рис. 3. ARP Reply.

Для полного понимания картины происходящего стоит рассмотреть структуру ARP – пакета (рис. 4).

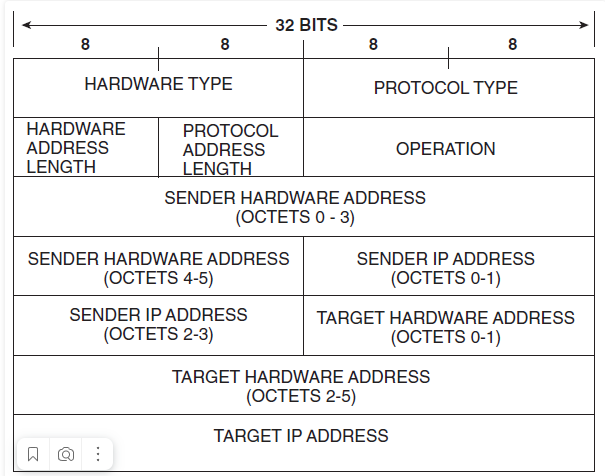


Рис. 4. Структура ARP-пакета.

Это общая схема, которая демонстрирует структуру ARP-пакета, теперь рассмотрим некоторые пункты поподробнее.

**Hardware Type** - 16-битное поле, определяющее “тип канальной среды”. Наиболее часто используемые типы представлены в таблице ниже (рис. 5).

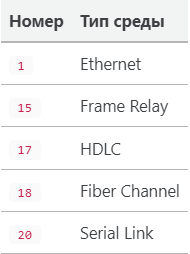


Рис. 6. Типы канальной среды.

Основную часть среди этих типов занимает Ethernet.

**Protocol Type** - 16-битное поле, определяющее протокол сетевого уровня, который отправитель связывает с идентификатором канала передачи данных. Для протокола IP версии 4 значение данного поля равно 0x0800.

**Hardware Address Length** - 8-битное поле, определяющее длину идентификатора канальной среды в байтах. MAC-адреса имеет длину 48 бит или 6 байт.

**Protocol Address Length** - 8-битное поле, определяющее длину адреса сетевого уровня в байтах. IP-адреса имеет длину 32 бита или 4 байта.

**Operation** - 16-битное поле, которое определяет какой тип пакета ARP используется (рис. 7).

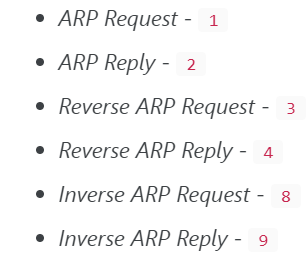


Рис. 7. Типы пакетов ARP и их нумерация.

Последние 20 байт приходятся на адресацию канальной среды и сетевого уровня источника и назначения запроса (MAC-адрес 6 байт \* 2 + IP-адрес 4 байт \* 2 = 20).

Всю информацию об ARP-пакете можно просмотреть в программе Wireshark в разделе “Address Resolution Protocol (request/reply)” (рис. 8).

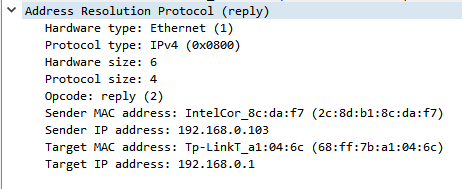


Рис. 8. Информация об ARP-пакете.

Протокол ARP является абсолютно незащищённым. Он не обладает никакими способами проверки подлинности пакетов: как запросов, так и ответов. Ситуация становится ещё более сложной, когда может использоваться самопроизвольный ARP (gratuitous ARP).

Самопроизвольный ARP — такое поведение ARP, когда ARP-ответ присылается, когда в этом (с точки зрения получателя) нет особой необходимости. Самопроизвольный ARP-ответ - это пакет-ответ ARP, присланный без запроса*.* Он применяется для определения конфликтов IP-адресов в сети: как только станция получает адрес по DHCP или адрес присваивается вручную, рассылается ARP-ответ.

Самопроизвольный ARP может быть полезен в следующих случаях:

* Обновление ARP-таблиц, в частности, в кластерных системах;
* Информирование коммутаторов;
* Извещение о включении сетевого интерфейса.

Несмотря на эффективность самопроизвольного ARP, он является особенно небезопасным, поскольку с его помощью можно уверить удалённый узел в том, что MAC-адрес какой-либо системы, находящейся с ней в одной сети, изменился и указать, какой адрес используется теперь.

**1.2 ARP-спуфинг**

ARP-спуфинг работает путем смены MAC-адреса на другой в пакетах ARP. Злоумышленник отправляет в сеть ложные ARP-ответы с обманутым MAC-адресом, когда устройство запрашивает соответствие между MAC-адресом и IP-адресом. В таком случае пакеты посылаются на неправильный MAC-адрес, и злоумышленник может перехватить данные, предназначенные для другого устройства.

ARP-спуфер может использоваться для перехвата сессий, когда пользователь входит в систему и передает свои учетные данные, такие как логины и пароли. Злоумышленник может использовать эти данные для получения несанкционированного доступа к системам, причиняя материальный и репутационный ущерб.

Основными целями атак ARP spoofing являются:

* для трансляции нескольких ARP-ответов по всей сети
* для вставки поддельных адресов в таблицы MAC-адресов коммутаторов

• посылать слишком много ARP-запросов узлам сети.

Когда атака ARP спуфинга успешна, злоумышленник может:

* Выполнить перехват сеанса: Если злоумышленник получает идентификатор сеанса, он может получить доступ к активным учетным записям пользователя.
* Распределенный отказ в обслуживании (DDoS): Вместо того чтобы использовать свою машину для запуска DDoS-атаки, злоумышленники могут указать MAC-адрес сервера, на который они хотят напасть. Целевой сервер будет переполнен трафиком, если они проведут эту атаку на несколько IP-адресов.
* Изменение связи: Загрузка вредоносной веб-страницы или файла на рабочую станцию.

Рассмотрим процесс ARP-спуфинга:

*До выполнения ARP-spoofing'а* в ARP-таблице узлов *A* и *B* существуют записи с IP- и MAC-адресами друг друга. Обмен информацией производится непосредственно между узлами A и B. (зелёная стрелка)

*В ходе выполнения ARP-spoofing'а* компьютер **C**, выполняющий атаку, отправляет ARP-ответы (без получения запросов):

* узлу *A*: с IP-адресом узла *B* и MAC-адресом узла *C*;
* узлу *B*: с IP-адресом узла *A* и MAC-адресом узла *C*.

В силу того, что компьютеры поддерживают самопроизвольный ARP (gratuitous ARP), они модифицируют собственные ARP-таблицы и помещают туда записи, где вместо настоящих MAC-адресов компьютеров *A* и *B* стоит MAC-адрес компьютера **C**. (красные стрелки)

*После того как атака выполнена*, когда компьютер *A* хочет передать пакет компьютеру **B**, он находит в ARP-таблице запись (она соответствует компьютеру *C*) и определяет из неё MAC-адрес получателя. Отправленный по этому MAC-адресу пакет приходит компьютеру *C* вместо получателя. Компьютер *C* затем ретранслирует пакет тому, кому он действительно адресован — т.е. компьютеру *B*. (синие стрелки).

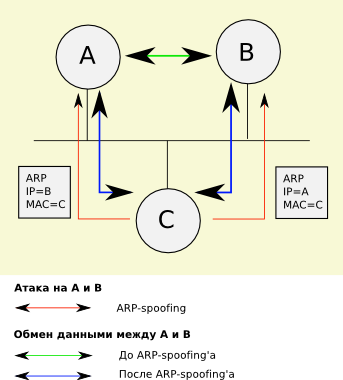


Рис. 9. Схема ARP-спуфинга.

### Инструменты для выполнения ARP-spoofing'а

В настоящее время существует несколько инструментов для выполнения ARP-spoofing'а, работающие как в ОС Linux, так и в ОС Windows.

Наиболее известные:

* [Ettercap](http://ettercap.sourceforge.net/)
* [Cain & Abel](http://www.oxid.it/cain.html)
* [dsniff](http://monkey.org/~dugsong/dsniff/)
* [arp-sk](http://sid.rstack.org/arp-sk/)

Все названные программы распространяются свободно.

Современные программы такого типа предоставляют два вида атаки:

Атака arp-request пакетами

Атака arp-reply пакетами

Атака arp-reply пакетами обычно выглядит так: сначала жертве единожды посылается arp-request пакет, а потом уже посылаются arp-reply пакеты через заданный промежуток времени. Это делается для того, чтобы у жертвы в arp-таблице наверняка появилась запись об адресах подменяемого узла. Если в данный момент времени такой записи не будет, то жертва, принимая пакет arp-reply, никаких данных в свою таблицу не внесёт. Значит, никакой подмены не произойдёт.  
  
Атака arp-request пакетами имеет следующий вид: посылаются пакеты arp-request через заданный промежуток времени. При чём обычно адреса получателя указываются не как Broadcast, а как истинный адрес получателя. То есть такой пакет будет послан только жертве. Другие станции подсети такой пакет не получат. Это разумно: исключается особенность arp-request пакета и атака становится направленной. Зачем другим станциям получать такой пакет? Их таблицу он не отравит. А вот если на какой-то станции стоит ПО для поиска аномалий в сети – это раскроет сам факт атаки. А аномалия здесь налицо: пойман пакет, в котором физический адрес не соответствует легитимному IP-адресу.

1.3 **Методы обнаружения и борьбы с ARP-спуфингом**

**Статический ARP**

Можно бороться со слабостями протокола ARP кардинально — просто не использовать его. ARP-таблицу можно сформировать вручную, при этом она становится неуязвимой к ARP-атакам. Для этого нужно добавить необходимые MAC-адреса в таблицу.

Если при этом отключить использование ARP на сетевых интерфейсах, то доступны будут только те системы, MAC-адреса которых добавлены в ARP-таблицу нашего узла и наш MAC-адрес добавлен в ARP-таблицы узлов, с которыми производится обмен трафиком.

Если не отключать использование ARP на сетевых интерфейсах, MAC-адрес заданный статически имеет приоритет. Если MAC-адрес для какого-то IP-адреса не задан, используется ARP-запрос.

Метод ручного формирования ARP-таблиц имеет следующий недостаток:

* добавляется много рутинной работы, связанной с добавлением и модификацией MAC-адресов. Каждое изменение в сети, связанное с заменой или перестановкой сетевых карт, должно сопровождаться редактирование ARP-таблиц в файлах.

**Использование VLAN**

Компьютер C может использовать ARP-spoofing против компьютера A только в том случае, если они находятся в одной сети канального уровня. В том случае, если они разделены маршрутизатором, атака невозможна (возможна атака на маршрутизатор, но это совсем другое дело).

VLAN’ы помогают сегментировать сеть — превратить одну сеть в множество изолированных на канальном уровне фрагментов, которые соединены между собой маршрутизатором. Атака ARP-spoofing возможна только между компьютерами находящимися в одном VLAN’е. В наиболее крайнем случае, когда в каждом VLAN’е находится только два компьютера: собственно компьютер и маршрутизатор, атака ARP-spoofing становится невозможной в принципе. К сожалению, такая организация сети являются очень требовательной к ресурсам маршрутизатора и используется редко.

##### Преимущества

1. От клиента ничего не требуется. Он работает, как и работал. Если IP-адрес задан статически, необходима смена IP-адреса и сетевой маски.
2. Данные не шифруются, и производительность не теряется. Малые потери на дополнительные поля в заголовке.

##### Недостатки

1. Необходима поддержка VLAN’ов коммутатором.
2. Необходимо потратить время на настройку VLAN’ов, дополнительную настройку DHCP’сервера.
3. Данные не шифруются. Если их прослушивание каким-то образом удастся, то они могут быть прочтены и/или модифицированы.

**Использование VPN**

При подключении к VPN все ваши данные будут проходить через зашифрованный туннель, гарантированно защищающий от любых хакерских атак.

#### **Использование туннелирования PPPoE (Point to Point Protocol over Ethernet)**

Сделать так чтобы абсолютно все передаваемые в сети данные были зашифрованы и прослушивание этих данных, ни к чему бы не привело. Любое же изменение данных, в случае возникновения, сразу же обнаружено.

##### Преимущества

1. Данные шифруются
2. Нет никаких требований к железу. Даже не обязательно чтобы сеть была коммутируемой

##### Недостатки

1. На клиенте требуется настройка доступа по PPPoE. На некоторых системах требуется инсталляция специального клиентского программного обеспечения.
2. За счёт потерь на инкапсуляцию, снижается производительность сети. Если выполняется шифрование данных, то временные потери увеличиваются.
3. При взаимном согласии сторон можно организовать обмен данными между узлами в обход шлюза.
4. При большом количестве соединений PPPoE (>200) возрастает нагрузка на центральный процессор сервера. Иногда, приходится жертвовать шифрованием данных.

**Патчи ядра системы**

Существует патч (предложенные buggzy) для ядер Linux, который позволяет свести к минимуму опасность успешного выполнения атаки ARP-spoofing против пропатченных систем.

Суть метода состоит в следующем. При приеме ARP-ответа производится сравнение старого и нового MAC-адресов, и при обнаружении его изменения запускается процедура верификации. Посылается ARP-запрос, требующий всем хозяевам IP-адреса сообщить свои MAC-адреса.

Если выполняется атака, настоящая система, имеющая этот IP-адрес, ответит на запрос, и, таким образом, атака будет распознана. Если же изменение MAC-адреса было связано не с атакой, а со стандартными ситуациями, ответа, содержащего “старый” MAC-адрес, не будет, и по прошествию определенного таймаута система обновит запись в кеше.

При обнаружении подозрительной ситуации (“двойника”) ядро выводит сообщение: “ARP\_ANTIDOTE: Possible MITM attempt!”, и не обновляет запись ARP-кеша, а наоборот, прописывает старую запись как статическую. После того как источник атаки выявлен и обезврежен, статическую запись можно удалить.

Но этот метод не применим к Windows-системам, и другим системам, работающим под управлением ОС, исходный код которых недоступен.

**2. ПРАКТИЧЕСКАЯ ЧАСТЬ**

**2.1 Создание сканера сети**

Практическая часть моей работы заключается в написании скрипта на языке Python, для демонстрации возможностей ARP-спуфинга. Для этого были написаны 3 скрипта с использованием библиотеки scapy. Также стоит разъяснить о начальных условиях. Есть 4 хоста, которые находятся в одной сети. Один из этих узлов является “хакерской” машиной, которая должна занять позицию между роутером и хостом, чтобы заниматься перехватом пакетов. Все хосты были установлены на виртуальную машину. Просмотрим конфигурации 1 хоста (рис. 10).

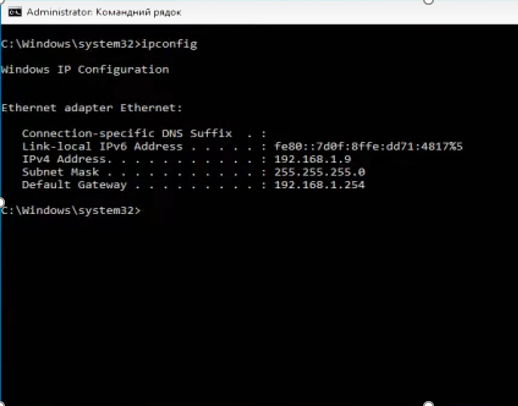


Рис. 10. Конфигурация 1 хоста.

Теперь рассмотрим конфигурацию 2 хоста (рис. 11).

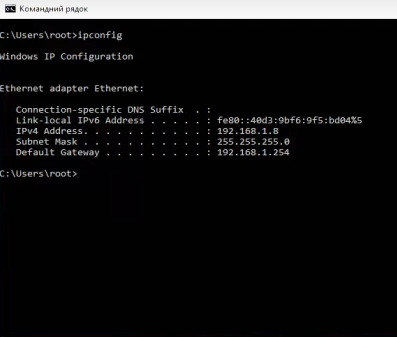


Рис. 11. Конфигурация 2 хоста.

Конфигурация 3 хоста (рис. 12).

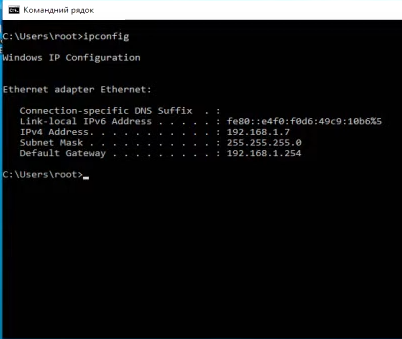


Рис. 12. Конфигурация 3 хоста.

Конфигурация компьютера с которого будет проводиться ARP-спуфинг (рис.13).

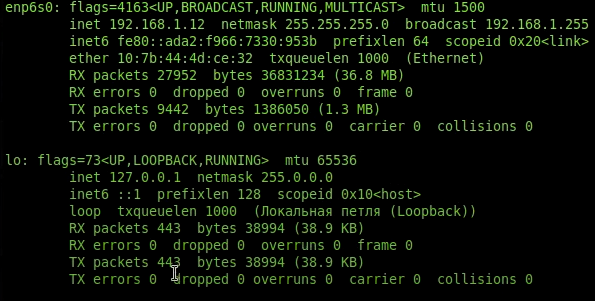


Рис. 13. Конфигурация 4 компьютера.

Первым этапом напишем простой сканер сети, который необходим для обнаружения ip-адресов и сопоставления им MAC-адресов. Сканер будем писать на языке программирования Python, как и все следующие скрипты, также во всех случаях будет задействована библиотека scapy. Процесс разработки сканера будет показан ниже. Первым этапом создаем функцию scan(ip), которая принимает на вход IP-адрес или диапазон IP-адресов в формате ('192.168.1.0/24'), который будет отсканирован. Далее создаем первый пакет - ARP-запрос (создается объект scapy.ARP), который будет отправлен на указанный IP-адрес (рис. 14):

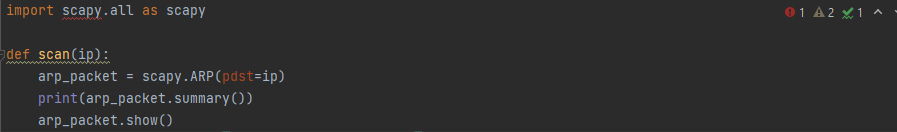


Рис. 14. Создание ARP-запроса.

Также хочется сказать о том, что означает pdst – это адрес доставки нашего пакета, все возможные поля нашего пакета выглядят следующим образом (рис. 15).

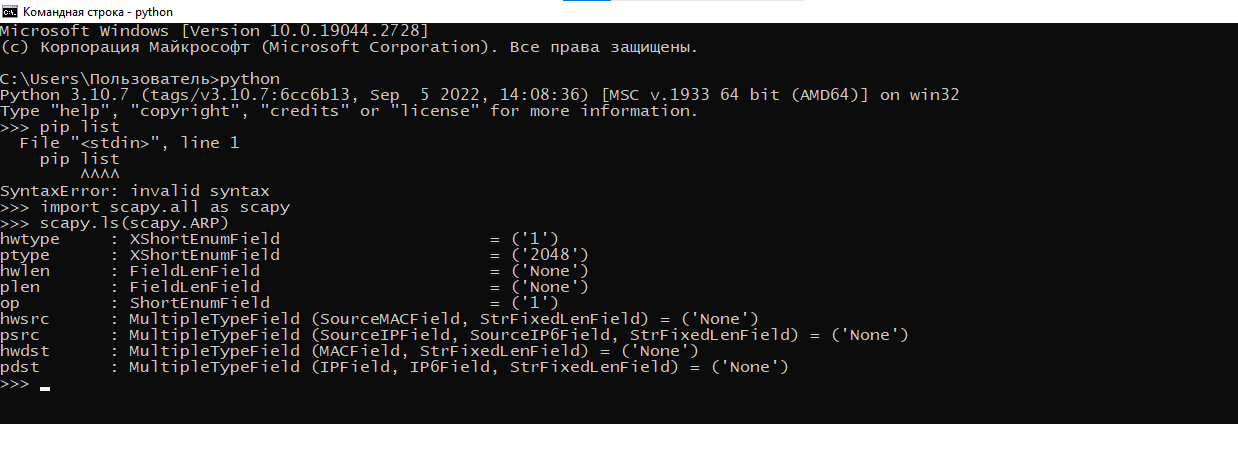


Рис. 15. Возможные поля пакета.

С помощью кода на рисунке 15 выведем наш arp-пакет (рис. 16).

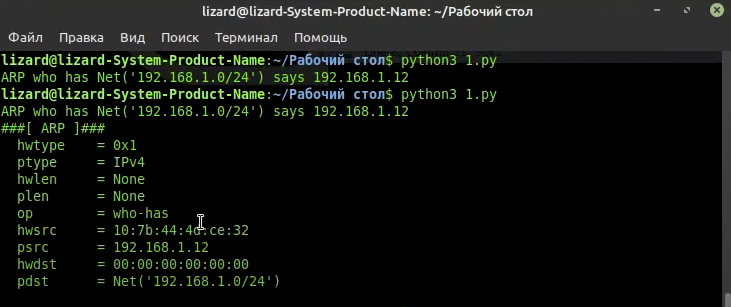


Рис. 16. ARP-запрос.

Следующим шагом создадим второй пакет - Ethernet-пакет (создается объект scapy.Ether), который содержит адрес получателя (broadcast - 'ff:ff:ff:ff:ff:ff').

После создаем третий пакет - объединенный пакет (объект arp\_push), который содержит как ARP-запрос, так и Ethernet-пакет.

Далее с помощью функции scapy.srp() отправляется объединенный пакет arp\_push и ожидается ответ в течение 1 секунды. Результат сохраняется в переменные ans (ответы) и unans (неотвеченные пакеты).

Наконец, выводится сводка принятых пакетов ans.summary(), которые могут свидетельствовать о том, что соответствующие устройства не доступны в сети, либо не отвечают на ARP-запросы. Финальный вариант данного скрипта (рис. 17).

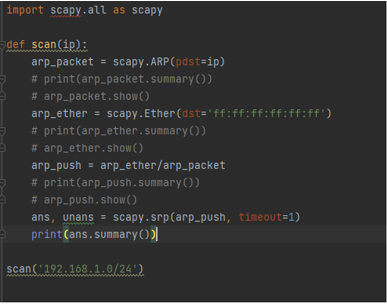


Рис. 17. Финальный вариант сканера.

Строки, которые несут роль комментариев использовались для проверки правильности пакетов путем их вывода, на финальной стадии данная необходимость отпала и поэтому я решил их сделать комментариями. Запустим программу и проверим ее работоспособность (рис. 18).

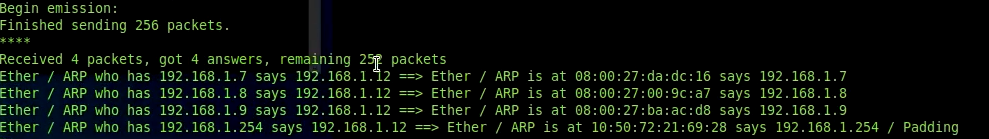


Рис. 18. Работы скрипта.

Для понимания того, что скрипт работает корректно проверим таблицу маршрутизации одного из хостов (рис. 19).

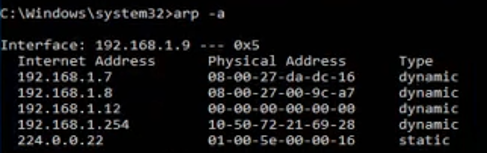


Рис. 19. ARP-таблица хоста.

Как мы можем увидеть сканер определил все правильно.

**2.2 Написание скрипта для ARP-спуфинга**

Теперь переходим ко второму этапу практической части, а именно напишем скрипт для того, чтобы посылать ответы, для того, чтобы стать между роутером и хостом. Сразу покажу финальный код данной программы (рис. 20), после дам пояснения.



Рис. 20. Код программы.

Первая строка импортирует все функции из библиотеки Scapy и дает им псевдоним "scapy", чтобы обращаться к ним более удобно.

В следующих двух строках создаются ARP-запросы. ARP (Address Resolution Protocol) используется для связывания IP-адреса с физическим (MAC) адресом в локальной сети.

Первый ARP-запрос отправляется на устройство с IP-адресом 192.168.1.7, и его MAC-адресом должен быть 08:00:27:da:dc:16. Это означает, что пакет предназначен для клиентского устройства с этим MAC-адресом. Пакет имеет тип операции 2, что означает ARP-ответ.

Второй ARP-запрос отправляется на маршрутизатор с IP-адресом 192.168.1.254, и его MAC-адресом должен быть 10:50:72:21:69:28. Это означает, что пакет предназначен для маршрутизатора с этим MAC-адресом. Пакет имеет тип операции 2, что означает ARP-ответ.

Последние две строки отправляют созданные ARP-запросы с помощью функции "scapy.send()". Эта функция отправляет сформированный пакет на сетевой интерфейс устройства, где он будет передан в локальную сеть. При успешной отправке функция возвращает количество отправленных пакетов.

Запустим программу и проверим ее работоспособность (рис. 21).

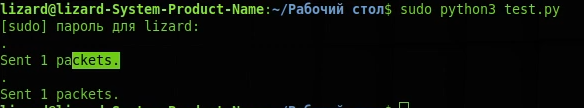


Рис. 21. Работа программы.

Как мы видим, успешно были отправлены 2 пакета. Проведем сравнение ARP-таблиц до и после использования этого скрипта, в верней части таблица “до”, в нижней соответственно “после” (рис. 22).

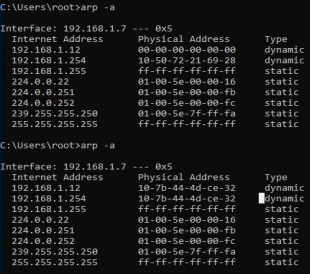


Рис. 22. Сравнение таблиц маршрутизации.

Как мы видим произошла подмена MAC-адреса роутера нашим MAC-адресом, теперь хост будет посылать пакеты нам. На этом практическая часть работы завершена.

**ЗАКЛЮЧЕНИЕ**

В процессе изучения принципа работы протокола ARP можно сказать, что он совершенно не защищен от хакерских атак, которые могут повлечь за собой множество негативных последствий. Принцип проведения атаки ARP-спуфинг был изучен, проанализирован и были написаны скрипты, которые помогли понять этот процесс изнутри. Есть методы защиты от ARP-спуфинга, приведенные в данной работе, большая часть из них сложна в реализации. Но по итогу проделанной работы принцип данных методов был усвоен.

В ходе работы были выполнены следующие задачи:

1. Изучены теоретические основы работы протокола ARP.
2. Изучен процесс ARP-спуфинга.
3. Изучены и описаны методы борьбы с ARP-спуфингом.
4. Были разработаны скрипты на языке Python, которые проиллюстрировали процесс ARP-спуфинга.

Основная цель работы также была достигнута.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. [https://timeweb.com/ru/community/articles/kak-polzovatsya-virtualbox -](https://timeweb.com/ru/community/articles/kak-polzovatsya-virtualbox%20-) “Настройка VirtualBox”.

# <https://iteducate.github.io/networks/arp_prot/> - “Структура и принцип работы протокола ARP”.

1. https://buildmedia.readthedocs.org/media/pdf/scapy/latest/scapy.pdf - “Техническая документация библиотеки scapy”.
2. <https://www.securitylab.ru/analytics/532149.php> - “Что такое ARP-спуфинг и как от него защититься”.
3. <http://xgu.ru/wiki/ARP-spoofing> - “ARP-spoofing”.