**Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский ядерный университет «МИФИ»**

Адрес: г. Москва, Каширское ш., д.31

**Отчет с анализом результатов**

**разработки интерактивного анализатора сетевого трафика**

Москва 2020 г

Исследование по созданию и изучению инструментов анализаторов сетевого трафика.

**Исследовательская команда:**

1. Барабанов Андрей Романович (Б20-505)

2. Кожухов Игорь Олегович (Б20-515)

3. Лебедев Вадим Андреевич (Б20-505)

4. Лебедев Михаил Михайлович (Б20-515)

**Научные руководители:**

1. Лебедев Филипп Владимирович

Содержание проекта:

**Название исследования:** Реализация интерактивного анализатора сетевого трафика.

**Цель исследования:** Разработка инструмента, способного захватывать и анализировать сетевой трафик.

**Задача 1:** Провести исследование о различных.

**Подзадачи:**

1. Изучить теорию сетей и исследовать реализованное программное обеспечение (аналоги поставленной задачи).
2. Разработка общей концепции анализа входящего и выходящего трафика.

**Задача 2:** Разработать информатический инструмент ( in silico) для анализа сетевого трафика.

**Подзадачи:**

1. Определение общей структуры алгоритма анализа сетевого трафика.
2. Написание программных модулей для реализации отдельных шагов алгоритма.

**Достигаемый технический результат:** Реализованный информатический инструмент (in silico) должен анализировать сетевой трафик в соответствии с условиями технического задания и поставленными задачами.

**Дата начала исследований:**8.09.2020 г.

**Дата создания инструмента in silico:**

**Оглавление**

**0. Аннотация**

**1. Исследование теории сетей и работы протоколов**

**1.1.** Вступление

**1.2.** Краткое описание и механизм работы анализаторов трафика

**1.3.** Популярные инструменты по работате с анализом трафика и их общая характеристика

**1.3.1.** Wireshark

**1.4.** Методы применения прослушивания сетевого трафика

**1.4.1.** Введение

**1.4.2.** Сетевая модель OSI.

**1.4.3.** Основы передачи данных в сети

**1.4.4.** IP-пакет

**1.4.5.** Протокол TCP

**1.4.6.** Создание пакетов (packet crafting)

**1.4.7.** Протокол ARP

**1.4.8.** APR-spoofing

**1.4.9.** Протокол HTTP, уязвимости протокола

**1.4.9.1.** Протокол HTTP

**1.4.9.2.** HTTP-запросы

**1.4.9.3.** Статус-коды

**1.4.9.4.** Уязвимости протокола HTTP

**1.4.9.5.** HTTP-сниффер вида Man-in-the-middle

**2. Список литературы**

**СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ**

**In silico** — термин, обозначающий компьютерное моделирование.

**Атака посредника**, или атака «человек посередине» (Man in the middle (MITM)) — вид атаки в криптографии и компьютерной безопасности, когда злоумышленник тайно ретранслирует и при необходимости изменяет связь между двумя сторонами

**Жертва** – пользователь сети, интересующий хакера.

**Логпасс** (Logpass) – Комбинация логина и пароля.

**Пакет** — это определённым образом оформленный блок данных, передаваемый по сети.

**Сниффер** (Sniffer) – программа или устройство для перехвата и анлаиза сетевого трафика.

**Спуфинг** - (spoofing) — ситуация, в которой один человек или программа успешно маскируется под другую путём фальсификации данных и позволяет получить незаконные преимущества.

**Хакер** - пользователь, вторгающийся в программное обеспечение.

**Хиты** (Hits) – действия пользователя на сайте.

**Аннотация**

В настоящее время активно используется передача данных через интернет.

К сожалению, все системы имеют ряд серьезных уязвимостей, использовав которые, злоумышленнику не составит труда получить интересующую его информацию – будь то пароли, сообщения, банковские и паспортные данные.

Одним из таких способов заполучения желаемой информации – является сниффер пакетов, способный перехватытвать информацию.

Целью исследования является разработка сниффера пакетов.

Пакетный сниффер - это программный инструмент для перехвата входящего и выходящего трафика, способный анализировать сеть и проходящие через нее данные.

В исследовании разбираются основы теории сетей, структуру пакетов и их уязвимости.

**1. Исследование теории сетей и работы протоколов**

* 1. **Вступление**

Скорость проникновения интернета значительно увеличивается с каждым годом. [internet world stat , 2020]. В таком темпе, анализ и оценка сетевого трафика черезвычайна сложна. Параллельно с ростом пользователей сети интернет, увеличивается количество киберпреступлений, связанных с незащищенностью интернет-потоков. Такие преступления являются серьезной проблемой, поэтому специалисты по информационной безопасности изучают новые способы защиты трафика сети.

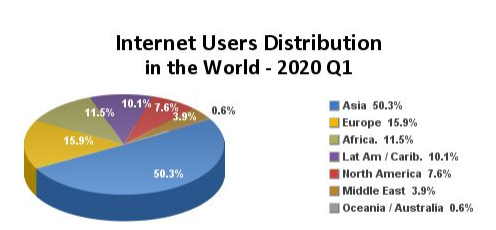


Рис. 1 – *Статистика пользователей интернета на 2020 год.*

**1.2. Краткое описание и механизм работы анализаторов трафика**

Технология перехвата сетевого трафика применима исключительно к сетям на основе протоколов TCP/IP и реализуемому соединению посредством сетевых карт Ethernet. Беспроводные сети тоже могут подвергаться анализу, все равно ведь изначально в такой системе присутствует проводное подключение (к маршрутизатору, к раздающему ноутбуку или стационарному ПК).

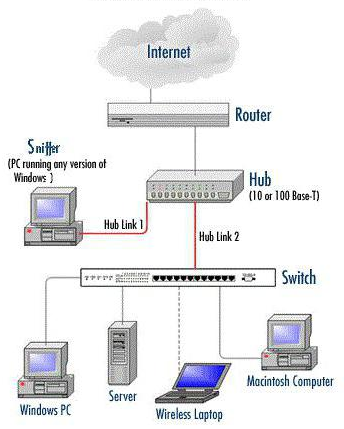


Рис. 2 – *Сеть.*

Передача данных в сети осуществляется не цельным блоком, а с помощью его разделения на стандартные пакеты и сегменты, которые при получении принимающей стороной объединяются в единое целое. Программа-сниффер способна отслеживать все возможные каналы передачи каждого сегмента, и в момент передачи (переадресации) незащищенных пакетов на подключенные к сети устройства (маршрутизаторы, хабы, свитчи, компьютеры или мобильные девайсы) производится извлечение нужных данных, которые могут содержать пароли. Таким образом, взлом пароля становится обычным делом техники, тем более, если он не зашифрован.

Но даже при использовании современных технологий шифрования пароля он может передаваться вместе с соответствующим ключом. Если это ключ открытого типа, получить пароль проще простого. Если и ключ зашифрован, злоумышленник запросто может применить программу-дешифратор, что в конечном итоге тоже приведет к потере данных.

**1.3. Популярные инструменты по работате с анализом трафика и их общая характеристика**

**1.3.1** **Wireshark**

Wireshark — это достаточно известный инструмент для захвата и анализа сетевого трафика.  
Wireshark работает с подавляющим большинством известных протоколов, имеет понятный и логичный графический интерфейс и мощнейшую систему фильтров.  
Кроссплатформенный, работает в таких ОС как Linux, Solaris, FreeBSD, NetBSD, OpenBSD, Mac OS X и Windows.

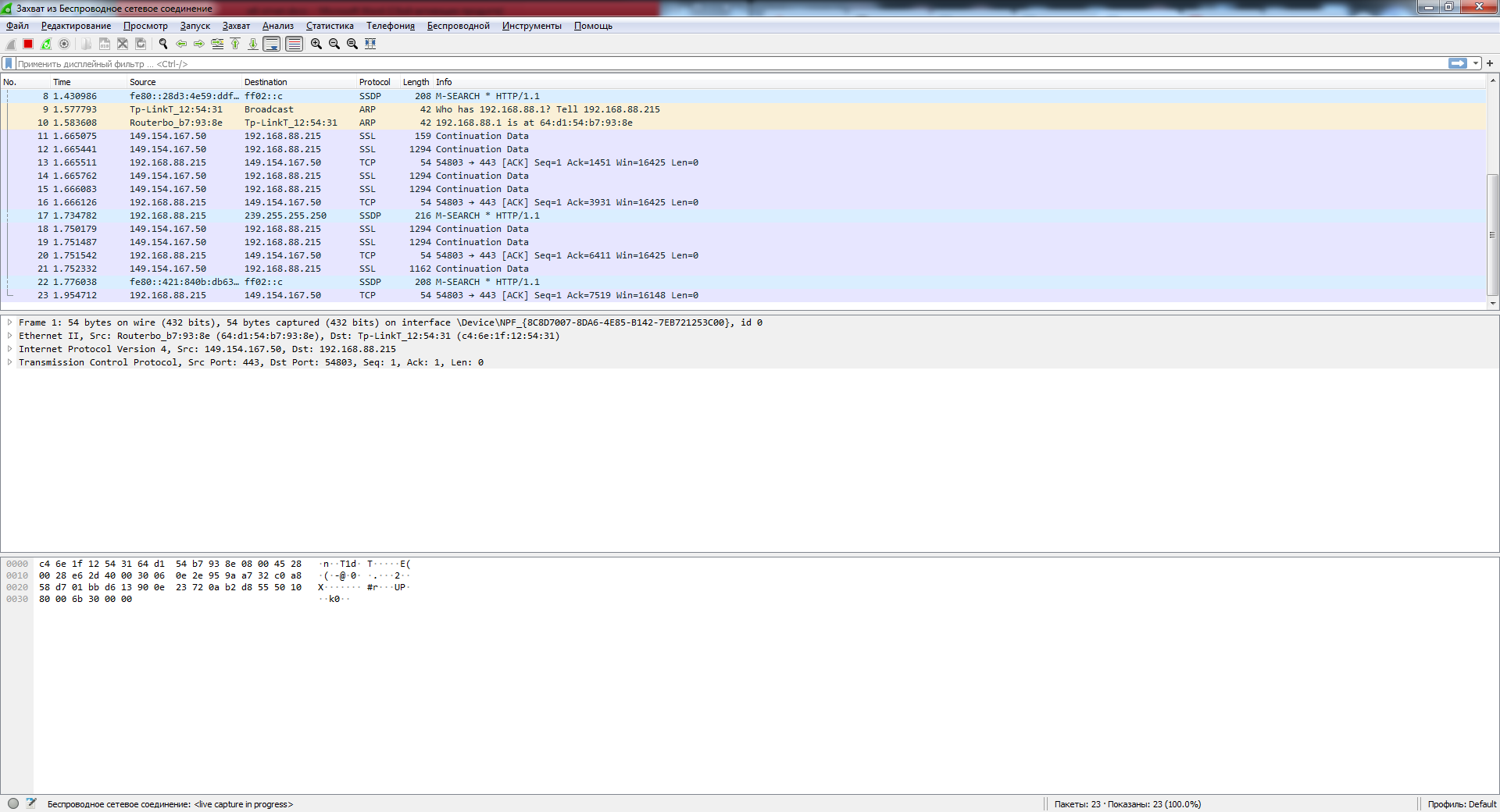


Рис. 3 – *Интерфейс программы Wireshark.*

**1.4. Методы применения прослушивания сетевого трафика**

**1.4.1.** **Введение**

Снифферы - это программы, которые перехватывают весь сетевой трафик.  
Данный тип программы способен перехватывать и расшифровывать имена и пароли пользователей, конфиденциальную информацию, нарушать работу отдельных компьютеров и сети в целом.

Известно, что в большинстве протоколов передачи данных ( FTP , POP, HTTP, telnet ) секретная информация между клиентом и сервером передаётся открытым текстом. Поэтому, не составляет большого труда получить доступ к такой информации. Прослушивание сетевого трафика - один из наиболее популярных видов атаки.

**1.4.2. Сетевая модель OSI**

Сетевая модель OSI (Open System Interconnection) обеспечивает взаимодействие различных сетевых устройств друг с другом. Модель определяет различные сетевые протоколы, распределяя их на 7 логических уровней, которые выполняют определенные функции. Управление сетевой передачей переходит от уровня к уровню, последовательно подключая протоколы на каждом из уровней.



Рис. 4 – *Сетевая модель OSI.*

1. **Физический уровень (*physical layer*)**

На первом уровне происходит передача данных (физических сигналов), представленных в двоичном коде, от одного устройства к другому.

Протоколы физического уровня: Ethernet, 802.11 WI-FI, IEEE 802.15 (Bluetooth) и т. д.

1. **Канальный уровень (*data link layer*)**

На втором уровне полученные с физического уровня данные упаковывают в кадры, проверяют их на целостность и исправляют ошибки. Здесь появляются первые идентификаторы – МАС-адреса отправителя и получателя. Канальный уровень делится на два подуровня: управление логическим каналом (LLC, Logical Link Control) и управление доступом к среде (MAC, Media Access Control).

Протоколы канального уровня: PPP, CDP, MPLS и т. д.

1. **Сетевой уровень (*network layer*)**

На этом уровне происходит маршрутизация трафика.

Здесь появляются IP-адреса отправителя и получателя.

Адресация происходит по IP-адресам.

Протоколы сетевого уровня: IP/IPv4/IPv6 (Internet Protocol), ARP, RIP,OSPF и т.д.

1. **Транспортный уровень (*transport layer*)**

Обеспечивает надёжную передачу данных от отправителя к получателю.

Протоколы транспортного уровня: TCP, UDP, ATP и т. д.

1. **Сеансовый уровень (*session layer*)**

Обеспечивает поддержание связи, позволяя приложениям взаимодействием между собой длительное время, управляет обменом информации, синхронизацией задач и т. д.

Протоколы сеансового уровня: PPTP, L2TP и т. д.

1. **Уровень представления (*presentation layer)***

Обеспечивает преобразование форматов сообщений, представление и шифрование данных.

Протоколы уровня представления: AFP, ICA, LPP и т. д.

1. **Прикладной уровень (*application layer*)**

Обеспечивает взаимодействие пользовательских приложений с сетью: позволяет приложениям использовать сетевые службы, отвечает за передачу служебной информации, формирует запросы к уровню представления.

Протоколы прикладного уровня: HTTP, HTTPS, FTP, TELNET, RDP и т. д.

**1.4.3.** **Основы передачи данных в сети**

Каждому устройству в сети Ethernet, соответствует уникальный шестибайтовый MAC-адрес (Media Access Control).

Единицей передачи данных, в такой сети является кадр (frame), который имеет определённую структуру и несёт в себе информацию о получателе, отправителе и самих данных. Кадр может иметь размер, примерно от 60 до 1514 байт, из них первые 14 байт являются служебными.

Когда требуется передать большое количество данных, они разбиваются на фрагменты и последовательно направляются в сеть. Кадр передаётся по сети и получает его каждое устройство этой сети.

Завершающая секция кадра служит для проверки целостности передаваемых данных и использует код циклического контроля ( CRC 32 - cyclic redundancy check ). Это мощная хэш-функция, для выявления искажённости числовых данных. Обычно функция аппаратно реализована в сетевой плате.

В качестве данных может быть IP-пакет или ARP-пакет. ARP (address resolution protocol ) - это служебный вспомогательный протокол, который осуществляет, трансляцию MAC-адресов в логические IP -адреса на основе широковещательной рассылки запросов.

**1.4.4.** **IP-пакет**

Пакет состоит из заголовка, служебной информации (options) и данных.

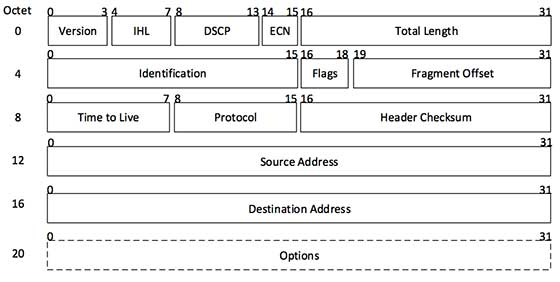


Рис. 5 – *Структура заголовка IP-пакета.*

Для того, чтобы наглядно рассмотреть структуру IP-пакета, отправляется несколько пакетов на адрес публичного DNS-сервера Google.

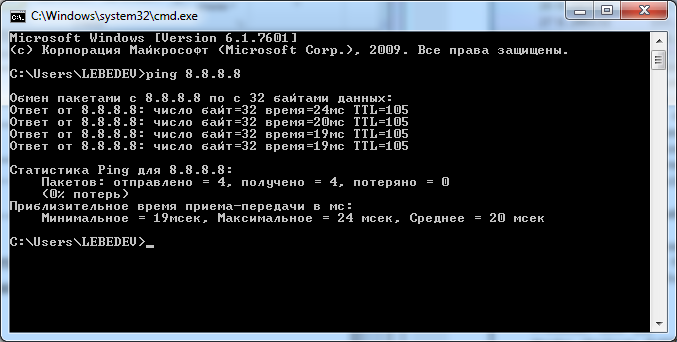


Рис. 6 – *Отправка запроса на DNS-сервер Google.*

При помощи вышеупомянутой в 1.3.1. программы Wireshark проанализирована структуру IP-пакета

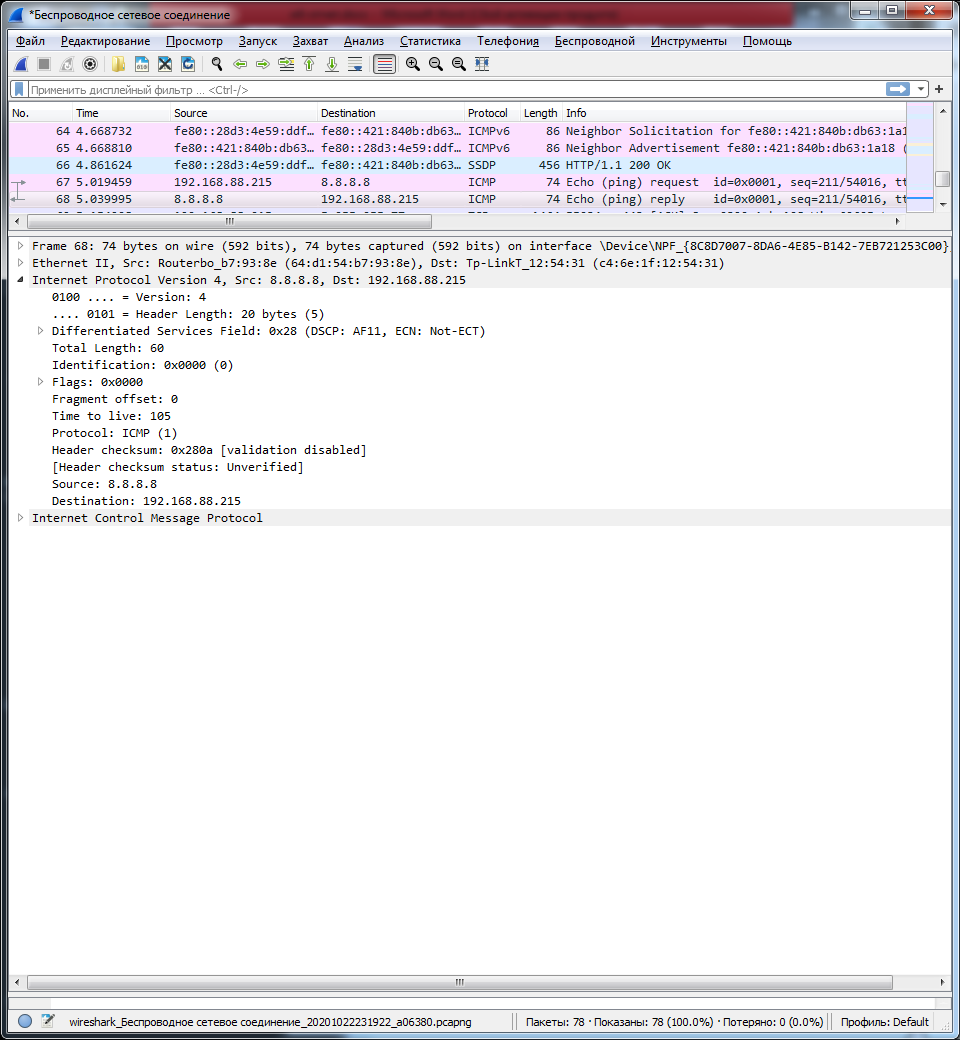


Рис. 7 – *IP-пакет в Wireshark.*

Жизнь пакета состоит из четырех сценариев: создание, пересылка, преобразование и завершение.

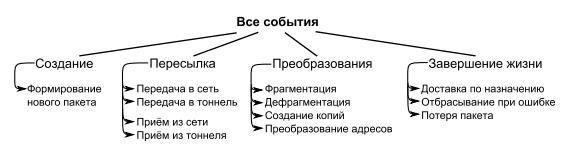


Рис. 8 – *Все события жизни пакета.*

Структура IP-пакета (анализ Рис. 7)

1. 0100 ... = Version: 4. Это первое поле, в котором указана версия протокола, Также, к значению в двоичном виде (0100) есть пояснение.
2. ... 0101 = Header Length: 20 bytes (5). Длина заголовка, в данном случае она равна 20 байт.
3. Differentiated Services Field: 0×00 (DSCP: CS0, ECN: Not-ECT)
   * 0000 00... = Differentiated Services Codepoint: Default (0)
   * ... ...00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
   * Третье поле DSCP + ECN - Используется приоритет по умолчанию, а также не используется указатель перегрузки.
4. Размер IP-пакета 60 байт, Total Length: 60.
5. Identification: 0xeba8 (60328). В данном случае IP-пакет не фрагментировался и поле идентификатор не представляет особого интереса.
6. Flags: 0×00
   * o 0... ... = Reserved bit: Not set;
   * o .0... ... = Don't fragment: Not set;
   * o ...0. ... = More fragments: Not set;

Убедиться в том, что пакет не фрагментировался нам поможет значение флагов, в данном случае все флаги выставлены в ноль, а значит фрагментации не было.

1. Fragment offset: 0. Также мы видим, что смещение у нас тоже нулевое.
2. Time to live: 105. В данном случае компьютер решил, что значение TTL=105 будет достаточным, чтобы добраться до конечного узла.
3. Protocol: ICMP (1). Поле протокол повествует о том, что внутри IP-пакета находится ICMP вложение, у протокола ICMP числовой код 1.
4. Header checksum: 0x87fb. После того, как пакет был сформирован, для его заголовка была посчитана контрольная сумма.
5. Source: 8.8.8.8. [IP-адрес](https://zametkinapolyah.ru/kompyuternye-seti/struktura-ip-adresa-kak-uznat-nomer-seti-i-nomer-uzla.html) назначения.
6. Destination: 192.168.88.215. IP-адрес источника.

**1.4.5. Протокол TCP**

TCP — это транспортный протокол, который выполняет функции управления передачей данных и следит за их сохранностью, считается надежным. Расшифровывается как Transmission Control Protocol (протокол управления передачей).

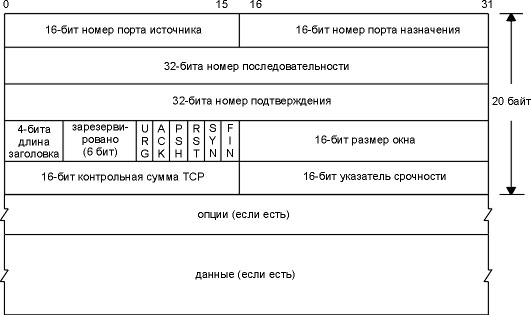


Рис. 9 – *Протокол TCP.*

У каждого TCP сегмента указывается порт источника и назначения, с помощью которых происходит идентификация отправляющего и принимающего приложения. Эти порты вместе с IP адресами уникально идентифицируют каждое соединение. Комбинация IP и порта — это сокет (socket).

Соединение отправителя и получателя (два узла) происходит по алгоритму:

**1.** Отправитель отсылает получателю специальный пакет, именуемый SYN, т.е. пригашает к соединению.  
**2.** Получатель отвечает пакетом SYN-ACK, т.е. соглашается  
**3.** Отправитель отсылает спец. пакет ACK, т.е. подтверждает, что согласие получено.

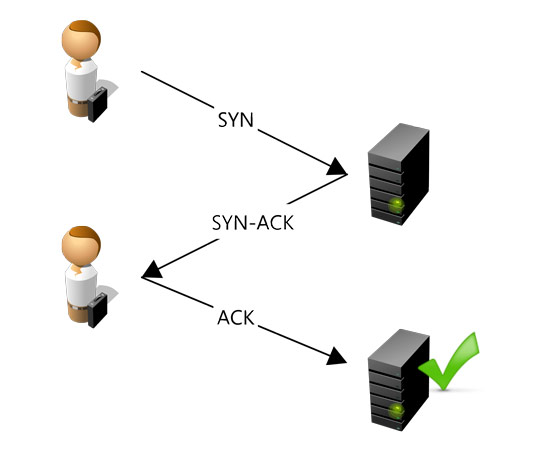


Рис. 10 – *соединение по TCP*

На этом TCP-соединение успешно установлено и получатель с отправителем могут спокойно обмениваться информацией. При передаче все пакеты данных нумеруются, отсылаются подтверждения о получении каждого из них, а потерянные пересылаются заново.

**1.4.6. Создание пакетов (packet crafting)**

Scapy является инструментом для создания пакетов вручную. У этой утилиты есть множество возможностей для применения: сборка пакетов с последующей отправкой их в есть, захват пакетов, исследование сети и многое другое.

Scapy позволяет сразу создавать пакеты высоких уровней (сетевого и прикладного), при этом Scapy автоматически дополняет нижние уровни.

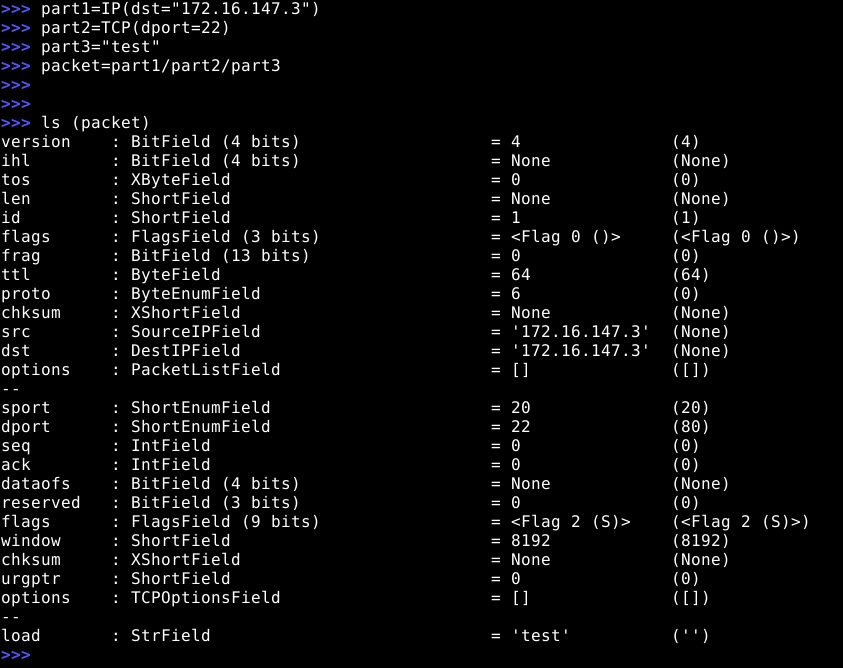
Однако можно и вручную собирать пакеты, начиная с канального уровня.

В упрощённом виде собранный пакет выглядит так:

Ether() / IP() / TCP() / “App Data”

Так выглядит собранный пакет. Здесь указан адрес назначения, порт и нагрузка в виде слова “test”.

*Рис. 11 – собранный пакет.*

Пакет создается по частям, а потом собрается эти в один пакет.

После чего выводится содержимое пакета:

*Рис. 12 – создание пакета.*

В любой момент времени можно поменять значение любого поля:

*Рис. 13 – изменение значения ппакета.*

Бывают ситуации, когда нужно отправить множество пакетов на множество портов.

Чтобы увидеть, какие пакеты будут отправлены, задействуется цикл for.

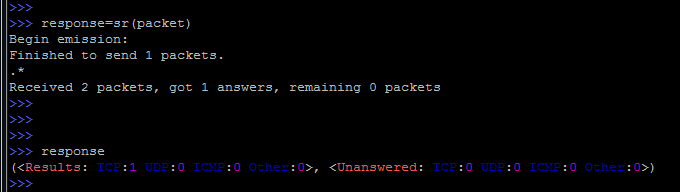


*Рис. 14 – отправка множества пакетов.*

**Отправка пакетов:**

1. функция **send()** – отправляет пакеты, используя сетевой (L3) уровень, никак не обрабатывая ответы. Используется принцип — отправили и забыли;
2. функция **sendp()** – отправляет пакеты, используя канальный (L2) уровень, учитываются указанные параметры и заголовки Ethernet кадров. Ответы всё так же не ожидаются и не обрабатываются;
3. функция **sr()** – является аналогичной send(), исключение составляет то, что она уже ожидает ответные пакеты;
4. функция **srp()** – отправляет и принимает пакеты, уровень L2

Чтобы обрабатывать ответные пакеты, можно назначить переменную, которая будет содержать ответ.

*Рис. 15 –* *ответные пакеты.*

**1.4.7. Протокол 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).

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).

ARP состоит из двух частей. Первая – определяет физический адрес при посылке пакета, вторая – отвечает на запросы других станций.

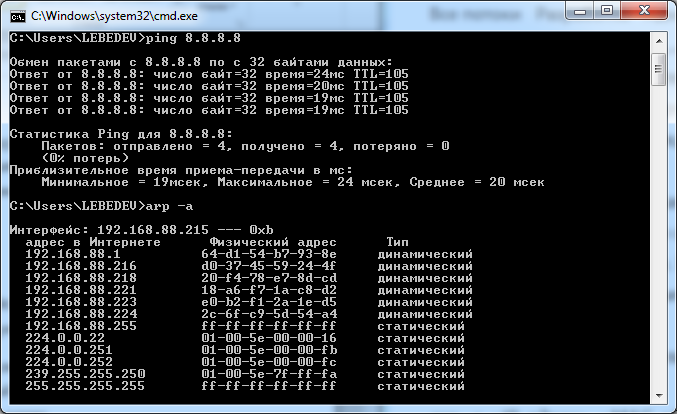
https://knazdom.ru/assets/dpic98014798.png

Рис. 16 – *ARP-таблица.*

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.

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

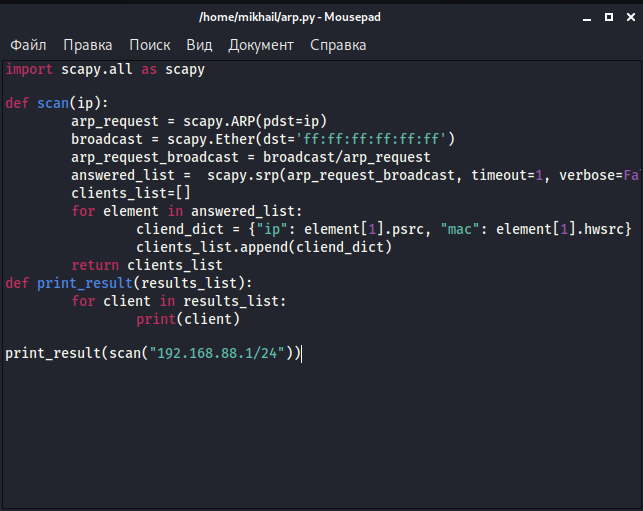


Рис. 17 – *ARP-запрос всем устройствам сети.*

Прежде, чем подключиться к одному из устройств, IP-протокол проверяет, есть ли в его ARP-таблице запись о соответствующем устройстве.

Если такая запись имеется, то происходит подключение и передача пакетов. Если же нет, то посылается широковещательный ARP-запрос, который выясняет, какому из устройств принадлежит IP-адрес.

Идентифицировав себя, устройство посылает в ответ свой MAC-адрес, а в ARP-таблицу отправителя заносится соответствующая запись.

Главным достоинством проткола ARP является его простота, что порождает в себе и главный его недостаток – абсолютную незащищенность, так как протокол не проверяет подлинность пакетов, и, в результате, можно осуществить подмену записей в ARP-таблице, вклинившись между отправителем и получателем.

**1.4.8. APR-spoofing**

ARP-spoofing (ARP — poisoning) — разновидность сетевой атаки типа MITM (англ. Man in the middle), применяемая в сетях с использованием протокола ARP.

Даннаый тип атаки основан на недостатках протокола ARP.



Рис. 18 – *Схема ARP-атаки.*

До выполнения ARP-атаки 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-атаки 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)

Для реализации сетевой атаки нам необходимо стать “человеком в середине” между двумя устройствами, в примере проводится ARP-атака между роутером и пользователем сети.

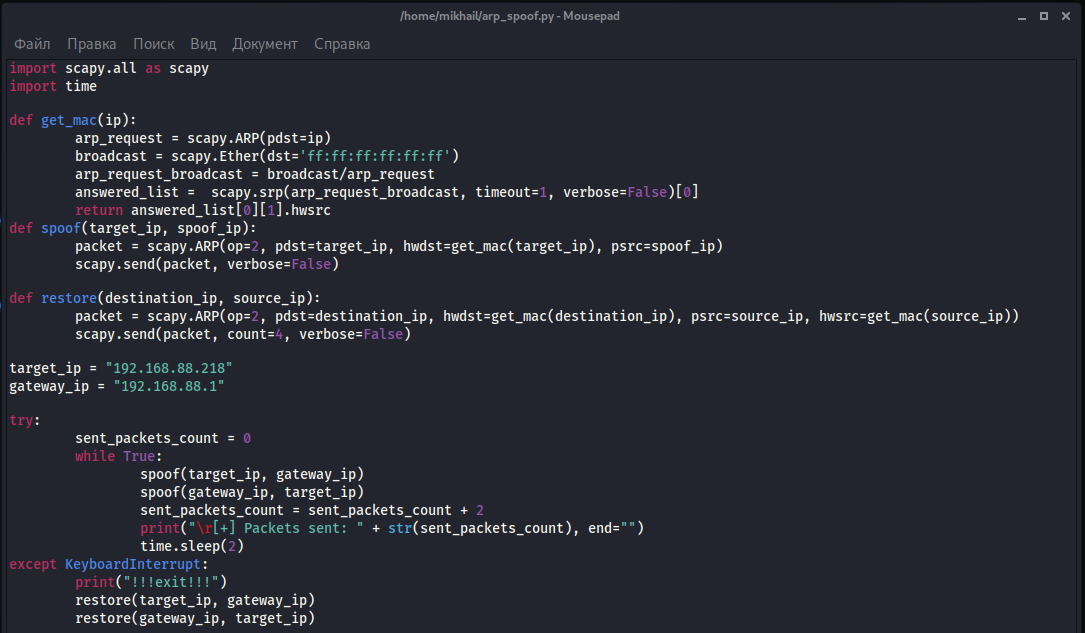


Рис. 19 – *Реализация ARP-атаки.*

Таким образом, в реализованном алгоритме постоянно посылается ARP-запрос на роутер (192.168.88.1), позволяющий нам мимикрировать под жертву(192.168.88.218), в свою очередь, жертве “посредник” представляется как марштрутизатор, тем самым выполненный код позволяет получать весь проходящий трафик.

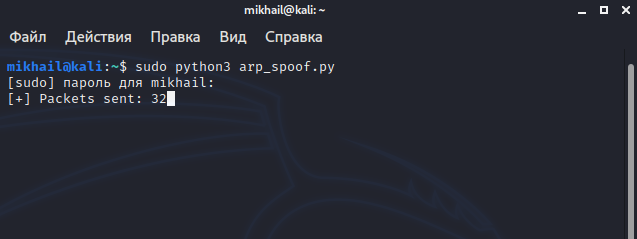


Рис. 20 – *Отправка 32 пакетов ARP для атаки типа MITM.*

Для устройства с IP-адресом 192.168.88.218 (жертва), исполнитель представился как 192.168.88.1 (роутер), и наоборот соответственно.

**1.4.9. Протокол HTTP, уязвимости протокола.**

**1.4.9.1. Протокол HTTP**

**HTTP** — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов.

Аббревиатура HTTP расшифровывается как *HyperText Transfer Protocol*, «протокол передачи гипертекста». В соответствии со спецификацией [OSI](http://en.wikipedia.org/wiki/OSI_model), HTTP является протоколом прикладного (верхнего, 7-го) уровня. Актуальная на данный момент версия протокола, HTTP 1.1, описана в спецификации [RFC 2616](http://tools.ietf.org/html/rfc2616).  
  
Задача, которая традиционно решается с помощью протокола HTTP — обмен данными между пользовательским приложением, осуществляющим доступ к веб-ресурсам (обычно это веб-браузер) и веб-сервером. На данный момент именно благодаря протоколу HTTP обеспечивается работа Всемирной паутины.

Как правило, передача данных по протоколу HTTP осуществляется через TCP/IP-соединения. Серверное программное обеспечение при этом обычно использует TCP-порт 80.

Вследствие его широкой распространенности он довольно часто подвергается атакам. Так, например, по данным SANS Institute's Internet Storm Center (Incidents.org), занимающегося вопросами интернет-безопасности и проблемами обнаружения атак, 80 порт (порт протокола HTTP) последние пять лет входит в пятерку наиболее атакуемых портов.

**1.4.9.2. HTTP-запросы**

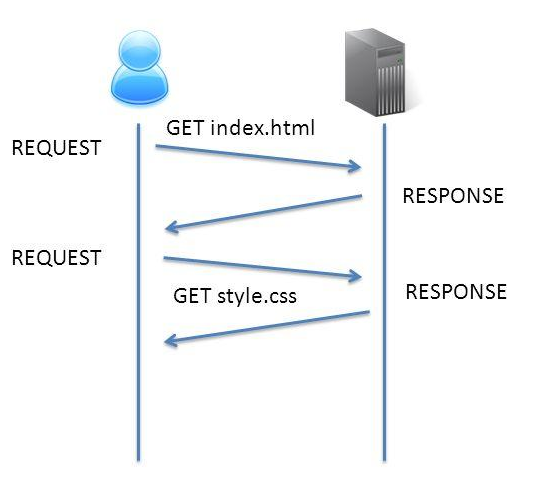


Рис. 21 – *HTTP протокол.*

Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. После этого клиентское приложение может продолжить отправлять другие запросы, которые будут обработаны аналогичным образом.

### Метод GET служит для получения любой информации, идентифицированной URI-Запроса. Если URI- Запроса ссылается на процесс, выдающий данные, в качестве ответа будут выступать данные, сгенерированные данным процессом, а не код самого процесса (если только это не является выходными данными процесса).

### Метод POST используется для запроса сервера, чтобы тот принял информацию, включенную в запрос, как субординантную для ресурса, указанного в Строке Статус в поле URI-Запроса.

**1.4.9.3. Статус-коды**

1. 1xx: Информационный — В этот класс выделены коды, информирующие о процессе передачи.
2. 2xх: Успех — Запрос был полностью получен, понят, и принят к обработке.
3. 3xx: Перенаправление — Клиенту следует предпринять дальнейшие действия для успешного выполнения запроса. Необходимое дополнительное действие иногда может быть выполнено клиентом без взаимодействия с пользователем, но настоятельно рекомендуется, чтобы это имело место только в тех случаях, когда метод, использующийся в запросе безразличен (GET или HEAD).
4. 4xx: Ошибка клиента — Запрос, содержащий неправильные синтаксические конструкции, не может быть успешно выполнен. Класс 4xx предназначен для описания тех случаев, когда ошибка была допущена со стороны клиента. Если клиент еще не завершил запрос, когда он получил ответ с Статус-Кодом- 4xx, он должен немедленно прекратить передачу данных серверу. Данный тип Статус-Кодов применим для любых методов, употребляющихся в запросе.
5. 5xx: Ошибка Сервера — Сервер не смог дать ответ на корректно поставленный запрос. В этих случаях  
   сервер либо знает, что он допустил ошибку, либо не способен обработать запрос. За исключением ответов на запросы HEAD, сервер посылает описание ошибочной ситуации и то, является ли это состояние временным или постоянным, в Содержание-Ответа. Данный тип Статус-Кодов применим для любых методов, употребляющихся в запросе

**1.4.9.4. Уязвимости протокола HTTP**

Изначально протокол HTTP разрабатывался в качестве протокола, ориентированного на высокую производительность при обмене электронной информацией, и поддерживает очень небольшое количество встроенных возможностей контроля за безопасностью. Рассмотрим, какие возможности протокола HTTP могут быть использованы в качестве уязвимостей, а также различные виды атак, которым он может быть подвергнут. Атаки можно объединить в четыре основных группы – атаки на протокол, атаки с использованием кэша HTTP, атаки на веб-приложения и атаки на доверительную модель протокола HTTP.

Информация, передаваемая в пакетах и заголовках HTTP, может быть использована для получения данных о веб-сервере или клиенте и, впоследствии, для выбора типа атаки. Атакующий может получить от веб-сервера следующие типы информации:

• Информация об учетных записях пользователей

• Коммерческая информация

• Информация о хосте (сервер или клиент).

• Взлом учетных записей пользователей.

**1.4.9.5. Основные HTTP -атаки**

**Атаки на HTTP сессию** (Session ID Hacking). Протокол HTTP не позволяет сохранять информацию о состоянии между сессиями. Поэтому веб-сайты используют такие механизма сохранения состояния, как cookies, динамические или статические поля URL и скрытые теги для контроля над сессиями. Большинство из этих механизмов обеспечивает авторизацию и аутентификацию пользователя и могут быть использованы при проведении атак подмены (Session ID Hijacking – см. рис.14) или повтора сессии (Session Replay).

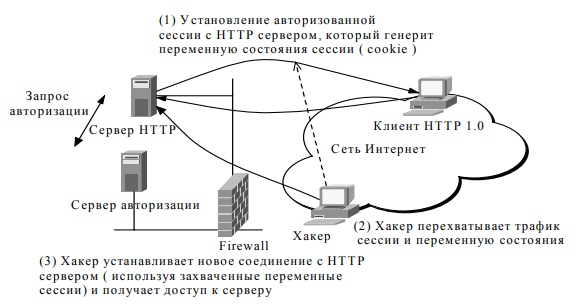


Рис. 22 – *Атаки на HTTP-сессию*

**HTTP Spoofing, HTTP Redirection**. Данные атаки используются для перенаправления HTTP клиентов на сайт злоумышленника с целью перехвата информации или для совершения взлома сервера.

**Атаки вида Man-in-the-middle** (перехват и подмена данных). Данный вид атаки включает в себя перехват и изменения пакетов данных с целью внедрения системы злоумышленника в TCP сессию.

**1.4.9.6. HTTP-сниффер вида Man-in-the-middle**

Предварительно инициализировав ARP-атаку, для того, чтобы весь трафик проходил через аппарат, используется скрипт, который анализирует входящий и выходящий трафик по протоколу HTTP.

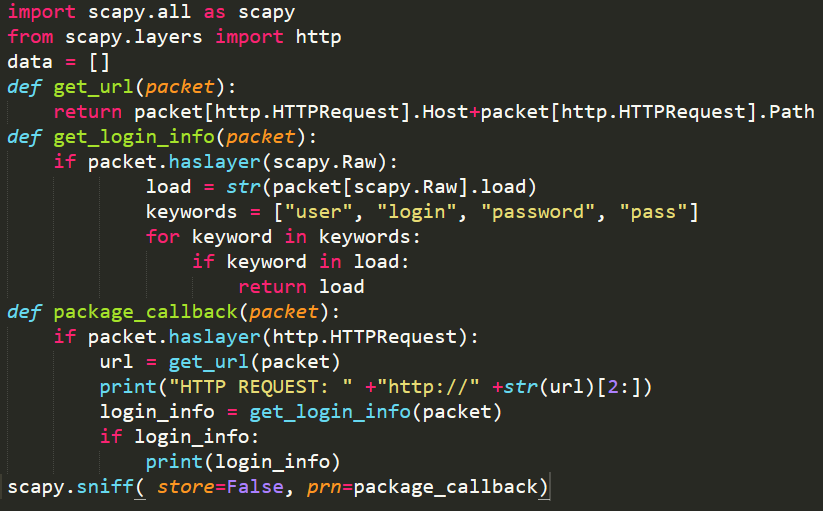


Рис. 23 – *Сниффер HTTP-трафика.*

Результатом исполняемого сценария будет вывод всех HTTP-запроросов, проходящих через “посредника”, таким образом, можно отслеживать какие веб-страницы посещает жертва.

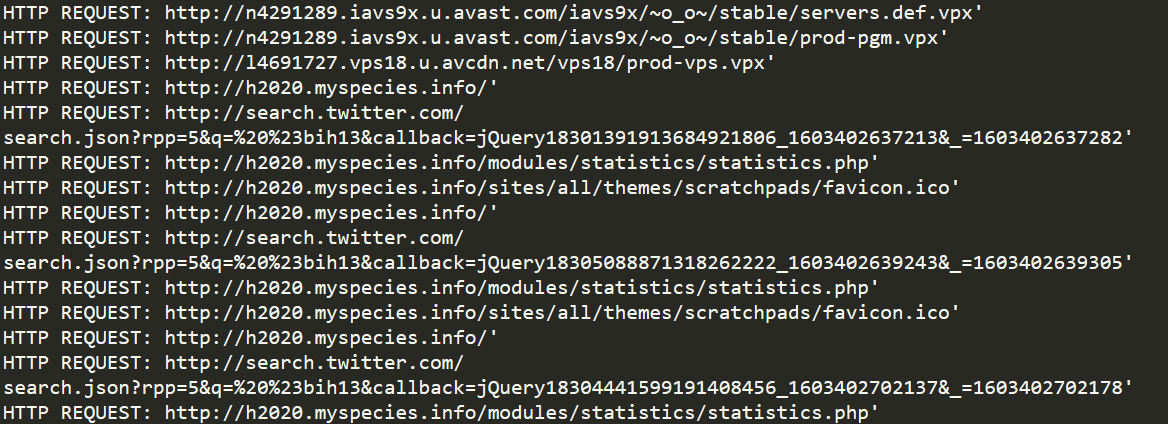


Рис. 24 – *Вывод истории запросов жертвы.*

Через протокол HTTP так же передаются пароли, банковские данные, которые часто интересны злоумышленикам.

Алгоритм поиска паролей строится на основе анализа POST-запроса, если после определенных хитов жертва отправляет запрос, который содержит так называемые таргетированные слова.

В примере используются стандратные цели: username, login, pass, password.

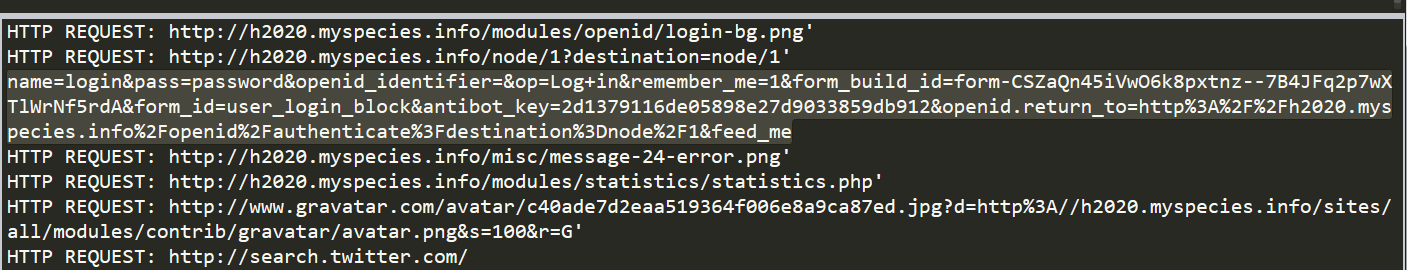


Рис. 25 – *POST-запрос, в котором содержится логпасс.*

Таким образом происходит перехват запросов, содержащий нужную информацию.



Рис. 26 – *Вывод логпассов жертвы.*

**СПИСОК ЛИТЕРАТУРЫ**

1. Avast Wi-Fi Hack Experiment Demonstrates «Reckless» Behavior of Mobile World Congress Attendees [Электронный ресурс]. URL: https://press.avast.com/en-us/avast-wi-fi-hack-experimentdemonstrates-reckless-behavior-of-mobile-world-congress-attendees
2. enDeering S. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard). 1998.
3. HTTP Authentication: Basic and Digest Access Authentication (RFC 2660, E. Rescorla, A. Schiffman, Aug. 1999) 166
4. Hypertext Transfer Protocol – HTTP/1.1 (RFC 2617, J. Franks, P. Hallam-Baker,J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, June 1999).
5. Information Security Management Handbook, 5th Edition, Harold F. Tipton, Micki Krause (Auerbach Publications, ISBN: 0-8493-1997-8).
6. Internet security : cryptographic principles, algorithms, and protocols, Man Young Rhee ( Wiley, ISBN 0-470-85285-2).
7. Internet Users Distribution in the world Available on https://www.internetworldstats.com/stats.htm [ Accessed on 2020/05/28]
8. Klaus Wehrle, Frank Pahlke, Hartmut Ritter et al. { Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004. { P. 648
9. Magers Daniel.(2002). Packet Sniffing: An Integral Part of Network Defense ,SANS Institute 6. Scapy official documentation Available on https://scapy.readthedocs.io/en/latest/
10. Mccanne St., Jacobson V. The BSD Packet Filter: A New Architecture for User-level Packet Capture // Proc. 1993 Winter USENIX Conference, 1993. P. 259{269.
11. Miller, R. (2019). The OSI Model: An Overview. SANS Institute., Page(s):5-12
12. Nimisha P, R. G. (2014). Packet Sniffing: Network Wiretapping. IEEE International Advance Computing Conference.
13. Pallavi Asrodia, H. (2012 ). Network Traffic Analysis Using Packet Sniffer . International Journal of Engineering Research and Applications .
14. Pizzonia M., Rimondini M. Easy Emulation of Complex Networks on Inexpensive Hardware // Proc. 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM 2008). Innsbruck. { 2008. Pp. 7:1{7:10. 5. Klaus Wehrle. Linux Network Architecture //
15. Postel J. Internet Protocol. RFC 791 (Standard). 1981.
16. Web Hacking: Attacks and Defense, Stuart McClure, Saumil Shah, Shreeraj Shah (Addison Wesley, ISBN 0-201-76176-9).
17. Web Security (A Stepby-Step Reference Guide), Lincoln D. Stein (Addison Wesley, ISBN 0-201-63489-9).
18. Wehrle K., Pahlke F., Ritter H. Linux Network Architecture. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004. 648 p.
19. Wi-Fi to carry up to 60 % of mobile data traffic by 2019 [Электронный ресурс] // JuniperResearch : офиц. сайт. URL: http://www.juniperresearch.com/press/press-releases/Wi-Fi-to-carry-60pc-of-mobile-datatraffic-by-2019
20. Кручинин С.В. Стеки сетевых технологий TCP/IP и OSI/ISO // Вопросы науки. 2015. Т. 3. С. 145-147.
21. Локхарт, Э. Антихакинг в сети. Трюки / Э. Локхарт. – СПб. : Питер, 2005. – 296 с.
22. Скудис, Э. Противостояние хакерам / Э. Скудис ; – М. : ДМК Пресс, 2003. – 506 с.
23. Чирилло, Дж. Обнаружение хакерских атак / Дж. Чирилло. – СПб. : Питер, 2003.