Лекция 1

ПРО IP, TCP, UDP, ARP И DNS
А ТАКЖЕ ПРО HTTP

rev. 5

Как работает Интернет?

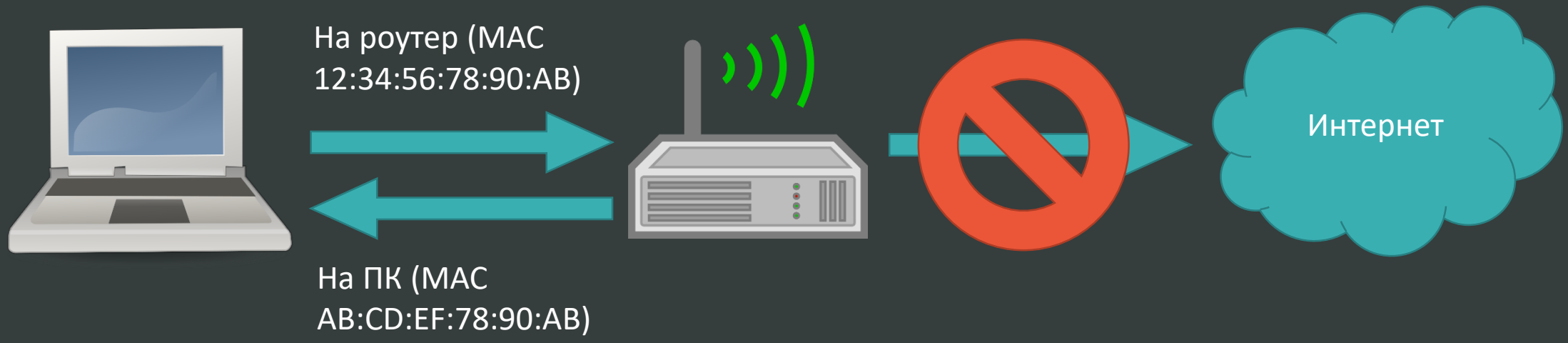
Уровни передачи данных OSI

- L7: Прикладной уровень
- L6: Уровень представления
- L5: Сеансовый уровень
- L4: Транспортный уровень
- L3: Сетевой уровень
- L2: Канальный уровень
- L1: Физический уровень

L2: Канальный уровень

- Данный уровень позволяет передавать данные в том же сегменте локальной сети
- Для адресации на данном уровне используются MAC-адреса сетевых карт
  - Вопреки интернет-легендам MAC-адрес компьютера не выбирается за пределы роутера или оборудования провайдера, поэтому «забанить по MAC-адресу» можно разве что в программах, которые сами его сообщат куда следует
  - MAC-адрес обычно состоит из 6 пар шестнадцатеричных цифр, например 01:02:03:04:05:06
  - Задается производителем сетевой карты, но обычно может легко быть изменен
- Единица данных на этом уровне называется «фрейм»
- На данном уровне, например, работает протокол ARP (о нем позже)
- Существует возможность отправить пакет «всем» (широковещательный пакет)

L2: Канальный уровень



L3: Сетевой уровень

- Отвечает за трансляцию логических адресов (например IP) в физические (например MAC) и маршрутизацию
  - IP(v4) адрес состоит из 4 байт (0-255), например 12.34.56.78
- Единица данных на этом уровне называется «пакет»
- На данном уровне, например, работает протокол IP

Маска подсети и шлюз

- Маска подсети определяет, какая часть IP-адреса отвечает за подсеть, а какая – за адрес внутри нее
  - Пример маски подсети: 255.255.255.0, такая маска обозначает, что три первых числа IP-адреса соответствуют адресу сети, а последний – адресу устройства
  - В Linux принята также сокращенная запись маски по числу ведущих единичных бит маски, для вышеуказанной маски она будет выглядеть как /24
- IP-адреса из одной подсети могут ходить друг к другу напрямую
- Для доступа к IP-адресам из других подсетей используется шлюз, которому сперва отправляются пакеты в соответствии с таблицей маршрутизации
- Пример:

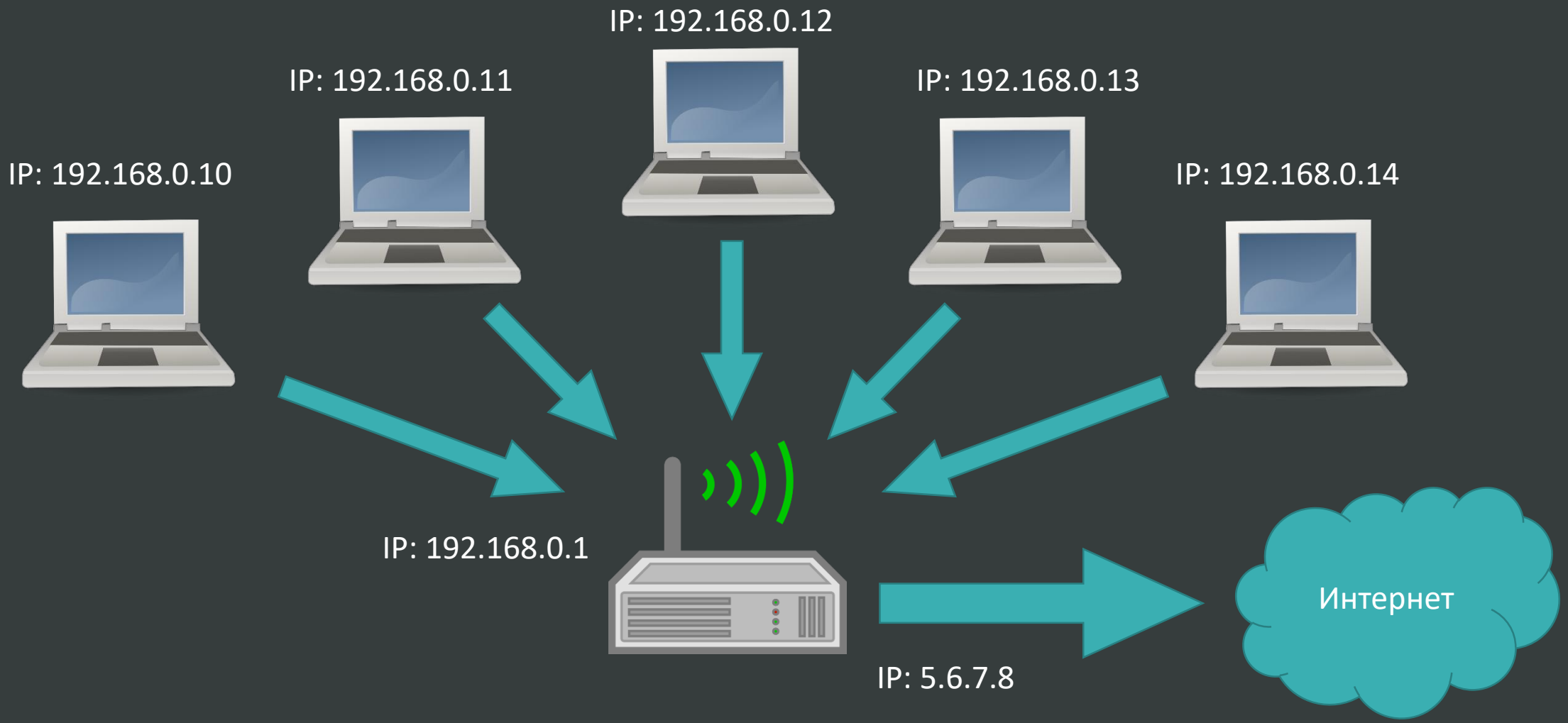
  IP-адрес: 192.168.0.1
  Маска: 255.255.255.0
  Адрес сети: 192.168.0.0

L3: Сетевой уровень

Внешний IP роутера?
NAT (network address translation)

- Данный механизм позволяет прятать за одним внешним IP адресом множество устройств
- Также он может служить некоторым средством защиты от злоумышленников, поскольку подключиться к устройствам внутри сети из интернета становится затруднительно: у них просто нет внешнего адреса
  - В случаях когда необходимо обеспечить возможность подключаться к компьютерам из локальной сети по внешнему IP роутера, применяют технологию port forwarding («проброс портов») или UPnP
  - Имейте в виду UPnP может быть опасен (например, китайская уязвимая камера может пробить доступ к себе из интернета, имея пароль по умолчанию)
- Роутер может быть сам спрятан за следующим NAT (например если у провайдера нет в распоряжении нужного числа IP адресов). В случае если он не спрятан и имеет общедоступный интернет-адрес, такой адрес называется «белым»

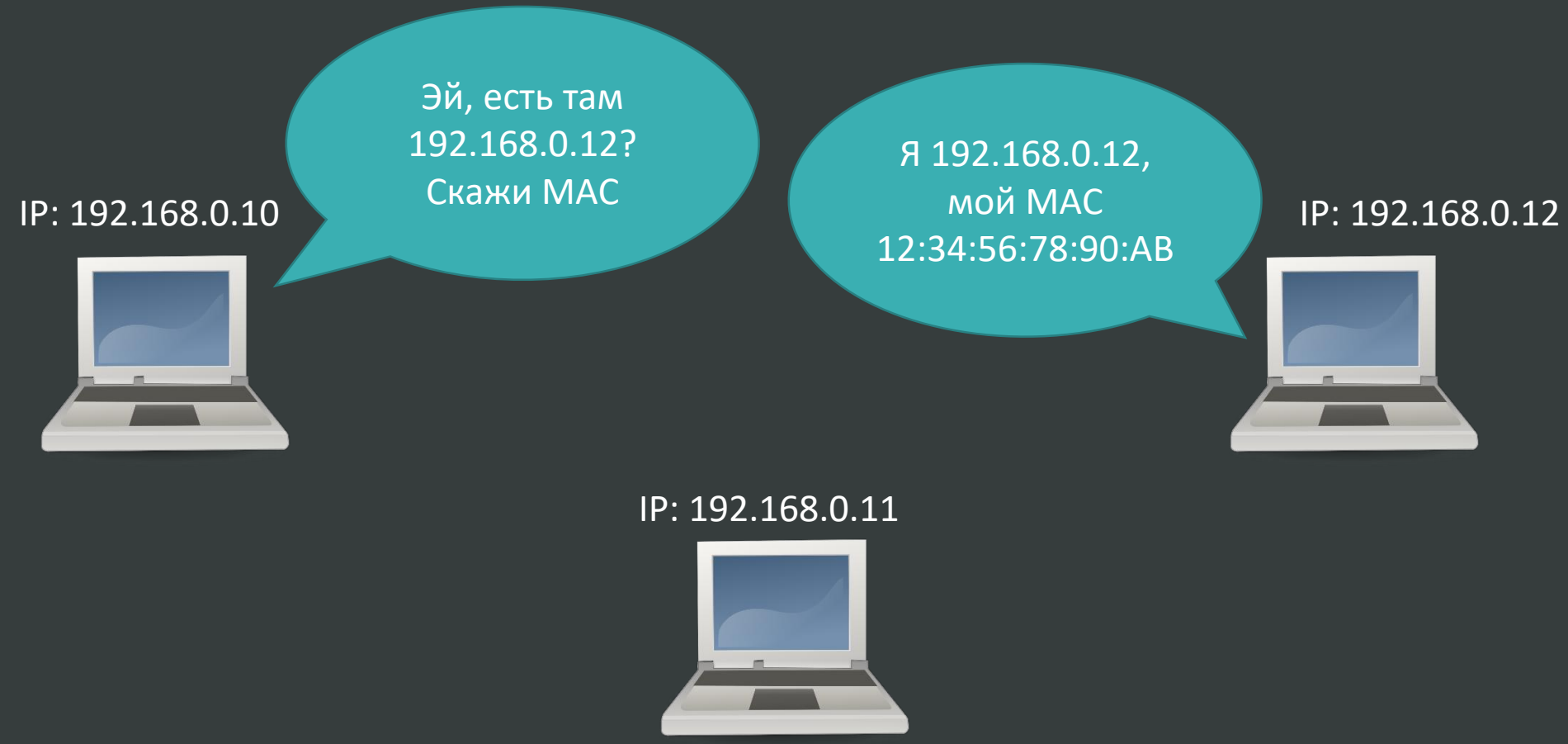
NAT (network address translation)



ARP (address resolution protocol)

- Данный протокол предназначен для определения MAC-адреса по IP-адресу компьютера
- Он полезен только для пересылки данных в одном фрагменте сети
- Существует атака на данный протокол, называемая ARP-spoofing

ARP (address resolution protocol)



ARP-spoofing

ARP-spoofing

IP: 192.168.0.10

Ммм, ладно

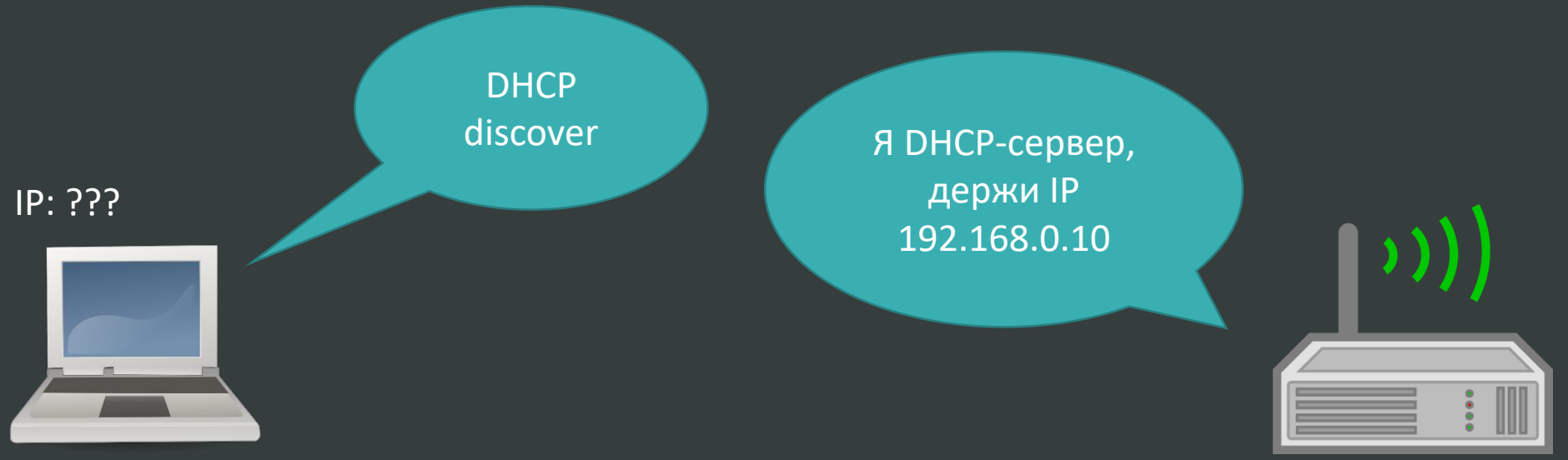
Кстати, 192.168.0.12, я 192.168.0.10 и мой MAC теперь AB:AD:D0:0D:13:37 [самопроизвольный ARP]

IP: 192.168.0.12

IP: 192.168.0.11

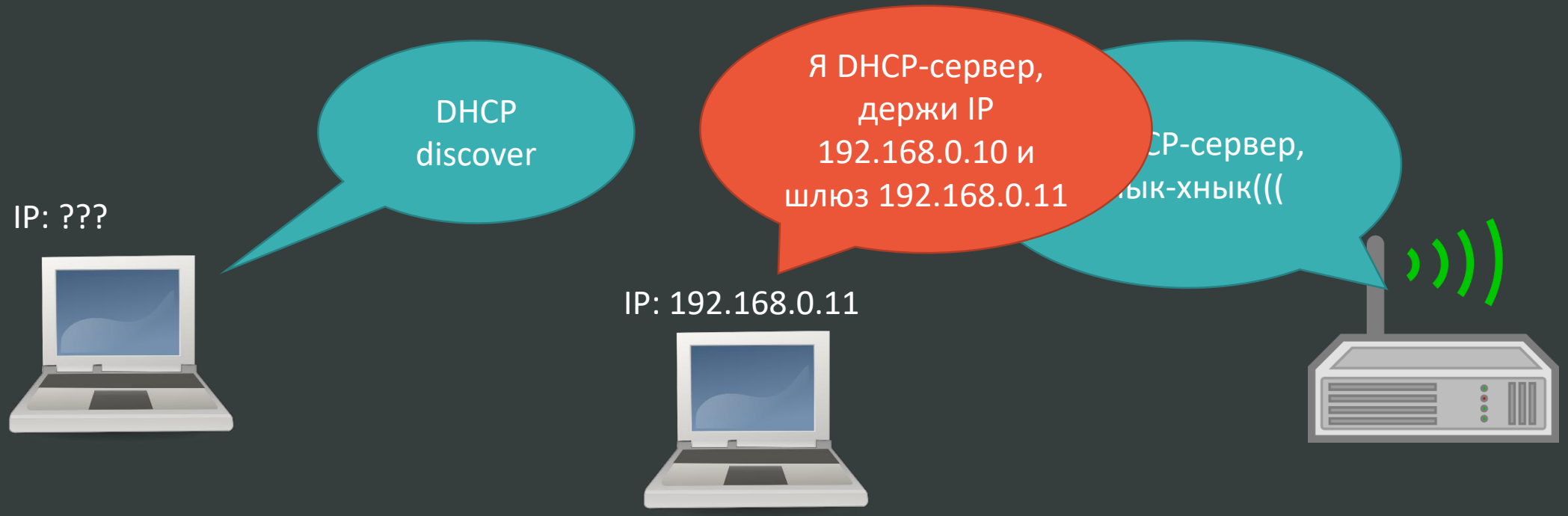
DHCP (dynamic host configuration protocol)

- Данный протокол предназначен для определения IP-адресов компьютеров, подключающихся к сети
  - В сетях, где нет DHCP-сервера, необходимо задавать IP-адрес вручную



DHCP-spoofing

- Понятно, что в сети может (намеренно или случайно) оказаться больше одного DHCP-сервера и тогда начнется неразбериха
  - DHCP может выдавать не только IP, но и шлюз, адреса DNS-серверов и многое другое



DNS (domain name system)

- Данный протокол предназначен в основном для определения IP-адресов компьютеров по их доменному имени
  - Доменное имя состоит из имени домена и далее имён всех доменов, в которые он входит, разделённых точками
  - Пример доменного имени: example.somedomain.ru:
    - example – поддомен домена somedomain.ru
    - somedomain принадлежит доменной зоне .ru
    - ru – домен верхнего уровня
  - Согласно RFC в доменном имени могут содержаться только буквы, цифры и дефисы, причем начинаться и заканчиваться на дефис доменное имя не может
- Может использоваться также для определения других параметров, связанных с доменным именем, например серверов для обмена почтой

Некоторые виды DNS записей

- A – содержит IP(v4) адрес(а), привязанный к данному доменному имени
- AAAA – то же самое, но для IPv6
- CNAME – используется для перенаправления на другое доменное имя
- MX – указывает почтовый сервер для данного домена
- NS – указывает на DNS-сервер для данного домена
  - Данная запись позволяет передать поддомены на управление вашему собственному DNS-серверу
  - Это открывает различные сценарии использования: подробнее можно почитать про iodine на хабре, например https://habr.com/ru/post/129097/
- TXT – произвольные данные, текст

Как посмотреть DNS записи

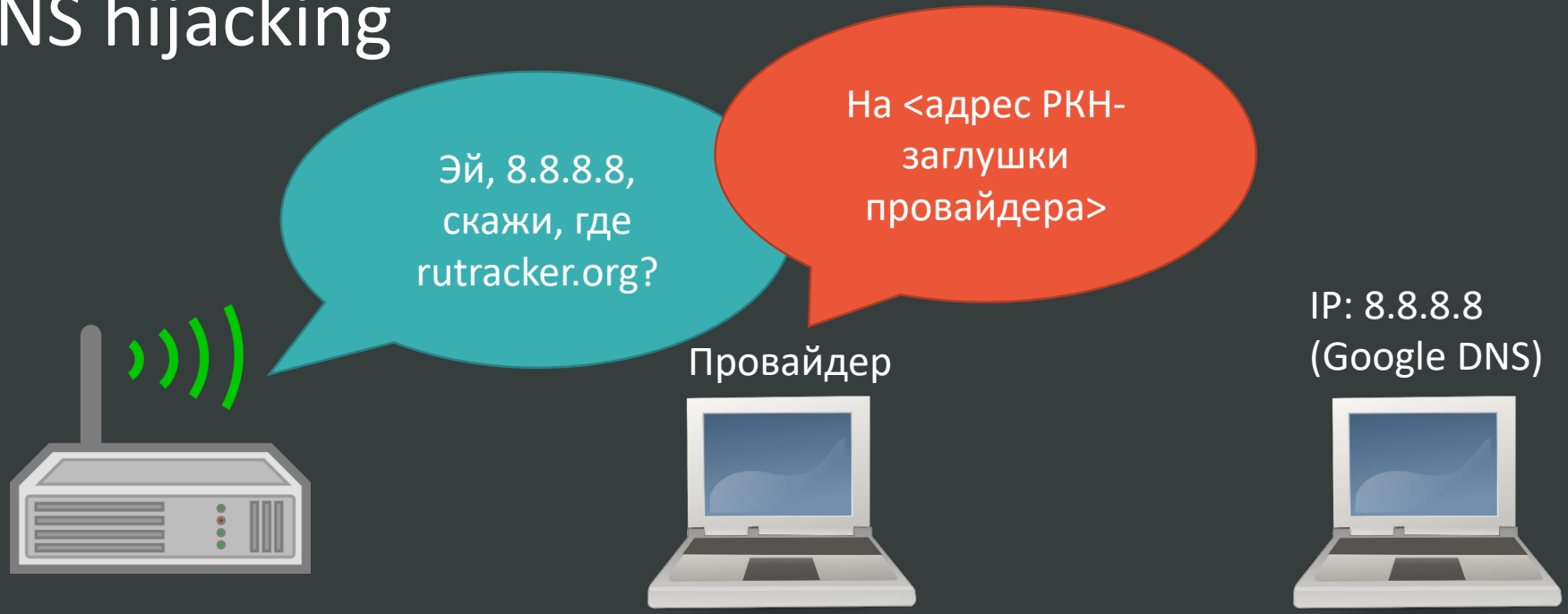
- Для просмотра DNS записей есть инструменты dig (Linux) и nslookup (Windows)
  - Впрочем, если вам нужен только IP-адрес, можно и просто пинганути сервер
- Об их аргументах можно почитать в документации, рассмотрим самый типичный сценарий – получение записи определенного вида по доменному имени:

```
n0n3m4@local:~$ dig @1.1.1.1 google.com AAAA
...
;; ANSWER SECTION:
google.com.             262     IN      AAAA
2a00:1450:4001:82b::200e
...
```

```
C:\> nslookup -type=AAAA google.com 1.1.1.1
...
Не заслуживающий доверия ответ:
Name:    google.com
Address:  2a00:1450:4001:831::200e
...
```

- Имя DNS-сервера здесь можно не указывать, тогда будет взят сервер по умолчанию
- Если не указывать тип записи, по умолчанию будет взята запись A (IPv4-адрес)
- Чтобы посмотреть все записи, можно использовать тип ANY (но он работает не со всеми DNS-серверами, с 8.8.8.8 работает, а с 1.1.1.1 – нет)

DNS hijacking



Листинг поддоменов

- Иногда может возникнуть необходимость посмотреть список поддоменов, которые есть у домена
  - Например, это может быть полезно во время багхантинга, чтобы найти скрытые поддомены, где может оказаться какая-нибудь отладочная консоль или что-нибудь еще
- Хотя посмотреть всю информацию о домене, когда вы знаете его имя, довольно просто, выкачать список поддоменов DNS не позволяет (если только не включен DNS Zone Transfer)
- Поэтому для решения этой задачи существует несколько разных способов:
  - Перебор, например при помощи KnockPy https://github.com/guelfoweb/knock
  - Использование истории выпуска HTTPS-сертификатов (Certificate Transparency) https://github.com/eslam3kl/crtfinder
  - Использование поисковиков, можно как вручную при помощи запросов вида site:domain, так и при помощи инструментов вроде https://github.com/OWASP/Amass/

Время задач

DNS

Категория: Lesson 1 / Network

Решивших: 23

Время: 00:10:15

- Доступ к задачам можно получить на nsuctf.ru
- Для различных вопросов рекомендую присоединиться к @NSUCTF в Telegram
- Для решения этой задачи вам могут пригодиться инструменты dig и nslookup
  - Примеры использования: dig @1.1.1.1 google.com A и nslookup -type=A google.com 1.1.1.1

L4: Транспортный уровень

- Отвечает за доставку сообщений до адресатов, обеспечивает возможность создания соединения
  - Основные протоколы (TCP и UDP) вводят понятие «порт» – число от 0 до 65535, ассоциированное с конкретной программой, использующей сеть на данном ПК
- Список протоколов не ограничен TCP и UDP, существует, например, и SCTP

L4: Транспортный уровень

Параметр	TCP	UDP
Назначение	Надежная передача данных	Быстрая передача данных, может выступать в качестве основы для других протоколов (как будто он L2)
Установление соединения	Требуется	Не требуется
Вид соединения для программы	Непрерывный поток	Множество одиночных пакетов
Повторная отправка потерянных сообщений	Обеспечивается	Отсутствует
Сохранение порядка сообщений	Обеспечивается	¯_(ツ)_/¯
Подтверждение получения	Обеспечивается	¯_(ツ)_/¯
Скорость передачи данных	Зависит от сетевой задержки	Максимальная

Понятие порта

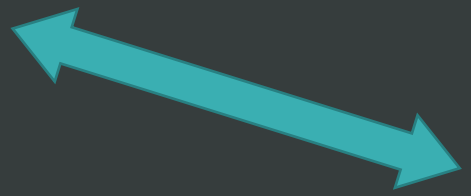
- Наличие такого понятия как «порт» позволяет иметь на одном компьютере (с одним IP адресом) несколько программ, работающих с сетью
- Порты используются как для входящих, так и для исходящих соединений
  - Для исходящих соединений порт выбирается из числа «эфемерных» или «динамических» портов, в Linux они лежат в диапазоне от 32768 до 61000
  - IANA рекомендует для эфемерных портов диапазон 49152 – 65535
- Порты TCP и UDP не зависят друг от друга: один и тот же номер порта может использоваться одновременно и для TCP-соединений и для UDP

Понятие порта

Клиент

22/tcp, SSH сервер

41234/tcp, Firefox



22/tcp, SSH сервер

67/udp, DHCP сервер

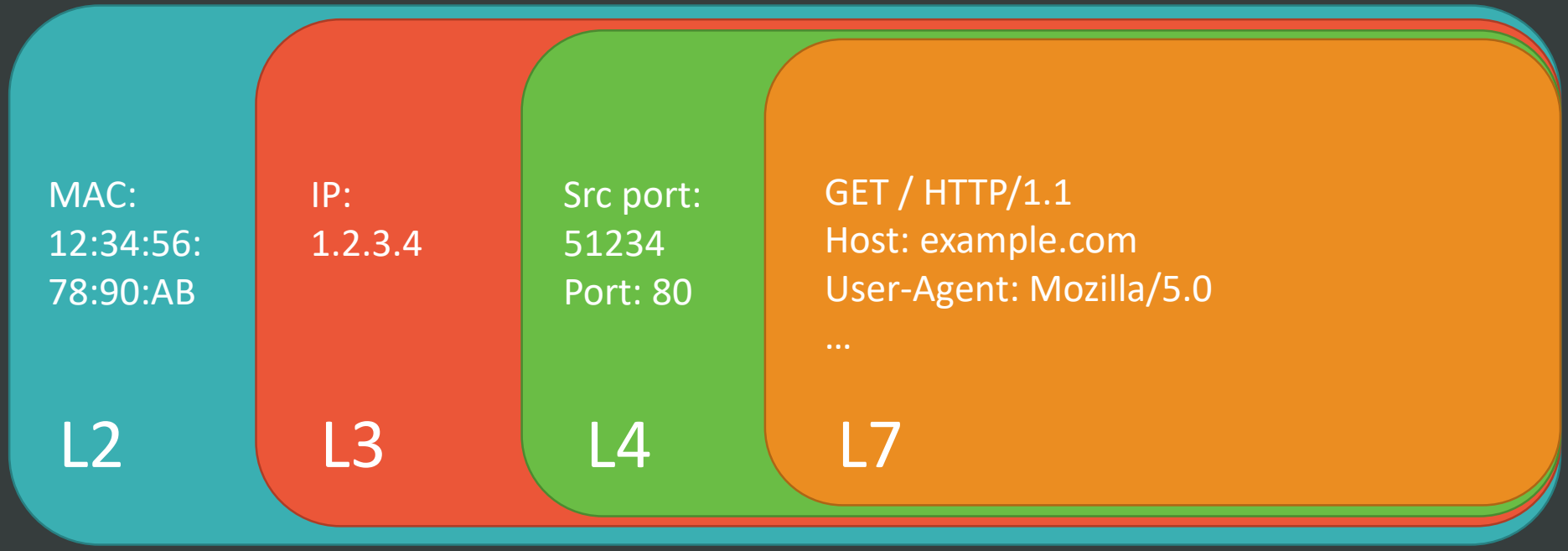
80/tcp, веб-сервер

Сервер

L7: Прикладной уровень

- Обеспечивает взаимодействие между программами по сети
  - Программы реализующие протокол согласно его стандарту всегда смогут понять друг друга даже если они написаны совершенно различными разработчиками
- Примерами протоколов данного уровня являются:
  - HTTP / HTTPS
  - FTP
  - DNS
  - SSH

Общая схема отношений между уровнями



Полезные программы

- Wireshark – программа, позволяющая просматривать записи сетевого трафика. Также позволяет осуществлять захват трафика
  - Кстати, в списке данных о пакете можно разглядеть соответствующие заголовки L2, L3, L4 и L7 (если Wireshark угадал протокол) и посмотреть на их поля
- nc (netcat) – программа, позволяющая устанавливать TCP и UDP соединения, соединяя ввод-вывод терминала с сетевым подключением (сокетом)
- Пример использования nc для подключения к TCP серверу на определенный порт:

```
n0n3m4@localhost:~$ nc nsuctf.ru 30022
SSH-2.0-paramiko_2.7.2
```

- nc – традиционный для хакера инструмент, при помощи него можно делать много интересного, например быстро кидать файлы по сети и не только
- Также nc может выступать в роли сервера при помощи флага -l

Interactive_Window:
C:\Program Files\Wireshark\Wireshark.exe
The Wireshark Network Analyzer
[OSI]

Interactive_Window:
C:\Program Files\ConEmu\ConEmu64.exe -run bash
Bash
[nc, ping]

Время задач

TCP

Категория: Lesson 1 / Network

Решивших: 17

Время: 00:00:02

- Доступ к задачам можно получить на nsuctf.ru
- Для различных вопросов рекомендую присоединиться к @NSUCTF в Telegram
- Для решения этой задачи вам может пригодиться инструмент nc (netcat)
  - Пример использования: nc nsuctf.ru 30022

HTTP

HTTP (hypertext transfer protocol)

- Это текстовый протокол, работающий поверх протокола передачи данных TCP
- Традиционный порт для HTTP-сервисов: 80/tcp
- Запрос состоит из:
  - Стартовой строки
  - Заголовков
  - Тела сообщения
- Ответ состоит из:
  - Стартовой строки ответа
  - Заголовков
  - Тела сообщения

Стартовая строка

- Стартовая строка состоит из:
  - Метода (GET, POST, OPTIONS и т.д.)
  - URL (путь к документу)
  - Версии протокола
- Пример стартовой строки:

```
GET /example HTTP/1.1
```

Основные методы HTTP

- GET – наиболее часто используемый метод, позволяет получить содержимое указанного ресурса. По RFC не должен использоваться для модификации состояния сервера
- POST – метод, позволяющий клиенту отправлять на сервер данные. Ответ, так же, как и в GET, может содержать данные. По RFC может использоваться для модификации состояния сервера
- HEAD – аналогичен GET за исключением того, что сервер в ответ на этот запрос посылает только заголовки, не тело ответа. Используется для кэширования
- OPTIONS – используется для определения возможностей сервера, например для получения списка доступных методов для страницы, сервер включает этот список в заголовок ответа Allow

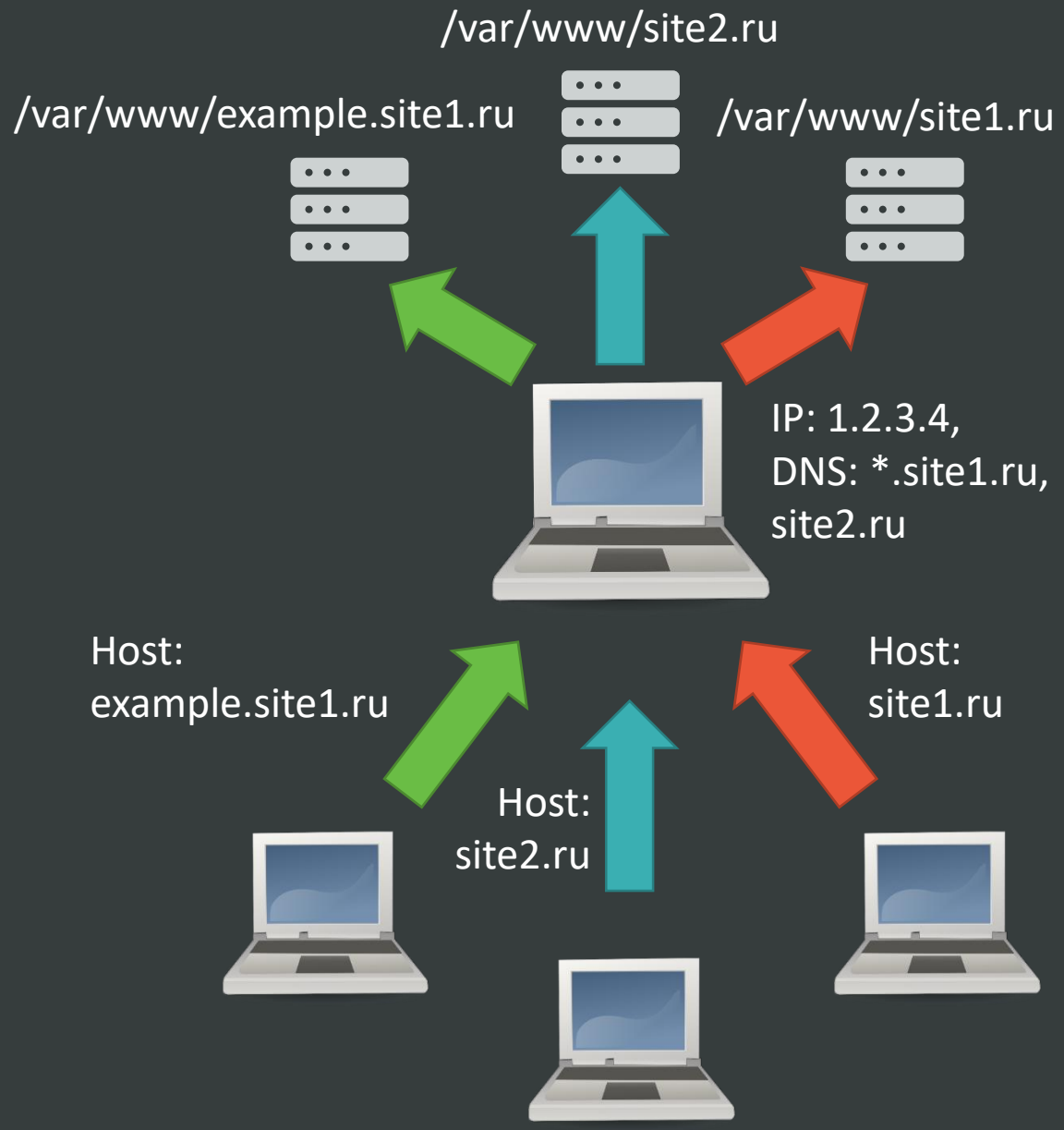
Заголовки

- Строки, содержащие разделенные двоеточиями пары параметров и их значений
- Популярными примерами заголовков бывают:
  - Host – параметр, указывающий, к какому серверу на данном IP адресе осуществляется обращение
  - User-Agent – параметр, указывающий, каким браузером пользуется клиент
  - Referer – адрес предыдущей страницы, с которой перешел пользователь
  - Cookie – параметр, содержащий cookies: пары имен и значений, которые сайт может сохранить у пользователя для его аутентификации
- Пример заголовков:

  Host: example.com
  User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0
  Cookie: login=vasya; password=vasya1337

Host

- Данный параметр позволяет веб-серверу различать, к какому из адресов пользователь запрашивает доступ
  - В протоколе IP, как было показано ранее, имя хоста не фигурирует
- В веб-серверах возможность иметь несколько HTTP-серверов на одном IP называется «Virtual hosts»
- На хосте «по умолчанию» (он открывается, если зайти на сервер по IP) могут оказаться различные чувствительные файлы, которые разработчик сайта забыл спрятать
  - Или ошибка 503, как на nsuctf.ru, если зайти на него по IP :)



User-Agent

- Заголовок, позволяющий определить, каким браузером пользуется пользователь при посещении сайта
  - У поисковых ботов тоже есть свои значения User-Agent
  - Некоторые сайты предоставляют поисковым ботам доступ для индексации платного контента, при помощи подмены User-Agent иногда можно читать платные статьи (нужно признать, в настоящее время такое бывает достаточно редко)
- Значение данного заголовка может использоваться в различной аналитике, которая, в свою очередь, может иметь определенные ошибки, позволяющие использовать это поле для атак
- Google решил объявить User-Agent "устаревшим" заменив его некими "Client Hints", а сам User-Agent заморозить, но это все еще не внедрено
- Пример:

  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0) Gecko/20100101 Firefox/92.0

Cookies

- Небольшие фрагменты данных, хранимые сервером на пользовательском ПК
  - Эти данные могут использоваться для авторизации и отслеживания пользователя
- Данные хранятся в виде ключ-значение
- Имеют срок действия, после которого удаляются браузером
- Бывают сессионными: данные cookies удаляются после закрытия страницы
- При наличии флага secure cookies будут отправляться только при HTTPS запросах
- При наличии флага httponly cookies будут недоступны для чтения на стороне веб-страницы (через, например, document.cookie в JavaScript)
- Пример:

login=vasya; password=vasya1337

Передача параметров

- Через URL:
  - В форме GET-параметров, после вопросительного знака. Пары параметр-значение разделяются амперсандами, параметр и значение разделяются символом равенства
  - Через сам адрес (например, имя файла в директории)
- Через тело сообщения:
  - Данные параметры используются в POST-запросах, могут содержать данные форм в различных форматах или файлы, загружаемые пользователем
- Пример:

```
POST /get-by-id/123?username=admin&nonce=1293 HTTP/1.1
Content-Length: 21
                                (пустая строка, это важно)
token=verysecrettoken
```

Формат данных POST-запроса

Формат тела сообщения задается заголовком Content-Type, примеры его популярных значений:

- application/x-www-form-urlencoded – традиционный формат по умолчанию, по способу кодирования совпадает с используемым в GET-параметрах
- application/json – формат JSON, который часто используется современными API. Напоминает объекты, используемые в языке программирования JavaScript
- multipart/form-data – формат, часто используемый при отправке файлов и прочих двоичных данных (остальные поля тогда тоже становятся в этом формате), к счастью нужен редко

```
…
Content-Type: application/x-www-form-urlencoded

login=vasya&pass=123

…
Content-Type: application/json

{"login":"vasya","pass":"123"}

…
----------------------------327027987976977044247870…
Content-Disposition: form-data; name="login"

vasya
----------------------------327027987976977044247870…
```

URL (unified resource locator)

- Стандарт записи ссылок на сетевые ресурсы. Состоит из схемы / протокола, логина, пароля, хоста, порта, пути с параметрами и идентификатора фрагмента (в простонародье "хэша")
  - Хэш в отличие от всего остального на сервер не передается и используется только клиентским ПО (браузер, JavaScript и так далее)
- Пример: http://vasya:qwerty@host.com:1337/getbyid/123?user=admin&nonce=1293#hash
- Не все параметры обязательны: например, порт, имя и пароль обычно отсутствуют
- Такое сложное устройство URL иногда используется в атаках для обхода фильтров ссылок (например, по хосту), поэтому важно о нем помнить
- Попробуйте угадать, каким будет хост следующего URL:

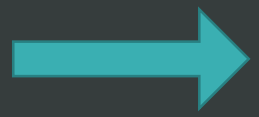
http://1.2.3.4:1337#@example.com:80

URL Encoding

- Что если мы захотим передать амперсанд или пробел в качестве адреса или параметра?
- URL (percent) encoding – способ записи специальных символов в URL или значениях параметров
- Символы записываются в виде %xx, где xx – шестнадцатеричное представление символа в кодировке ASCII / UTF-8. Например %20 это пробел
- Этот способ записи можно применять и к обычным символам, которые не требуют такой кодировки (например, буквам), что позволяет обходить некоторые фильтры, направленные на защиту от уязвимостей
  - Правда некоторые особо умные инструменты (requests, Chrome) могут сворачивать кодирование обратно в символы, так что лучше взять что-нибудь другое (curl, Firefox)
- Кириллица (и прочие символы UTF) всегда URL-кодируется

URL Encoding

0123456789abcdefghijklmnopqrstuv
wxyzABCDEFGHIJKLMNOPQRSTUVW
XYZ!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~



0123456789abcdefghijklmnopqrstuv
wxyzABCDEFGHIJKLMNOPQRSTUVW
XYZ%21%22%23%24%25%26%27%28
%29%2A%2B%2C-
./%3A%3B%3C%3D%3E%3F%40%5B%
5C%5D%5E_%60%7B%7C%7D%7E

Стартовая строка ответа

- Строка ответа состоит из:
  - Версии протокола
  - Кода состояния (трех цифр)
  - Пояснения (текстового пояснения к коду состояния, данное поле необязательно)
- Пример стартовой строки ответа:

```
HTTP/1.1 200 OK
```

Основные коды состояния

- 2xx (обычно 200) – успех, обычно за ним следует тело ответа (например страница)
- 3xx (обычно 302) – перенаправление, адрес перенаправления пишется в заголовке Location
- 4xx (обычно 403 или 404) – ошибка клиента, сюда относятся все ошибки, связанные с некорректным поведением клиента: например попытка запросить страницу, к которой у него нет доступа, или которой не существует
- 5xx (обычно 500) – ошибка сервера, сюда относятся, например, все падения сервера при попытке обработать запрос

Interactive_Window:
C:\Program Files\Wireshark\Wireshark.exe
The Wireshark Network Analyzer

Interactive_Window:
C:\Program Files\ConEmu\ConEmu64.exe -run bash -c "bash --rcfile <(echo '. ~/.bashrc; echo Running curl http://example.com --data username=vasya after keypress; read; curl http://example.com --data username=vasya')"
Bash

HTTPS

- HTTP + SSL, зашифрованный протокол
  - Можно рассматривать его как обычный HTTP, у которого обычные сокеты превратились в сокеты SSL
- Использует порт 443/tcp
- Мало чем отличается от HTTP с точки зрения уязвимостей, за исключением того, что HTTPS нельзя прослушивать, соответственно данные, переданные через него в общественной сети остаются защищенными (по крайней мере так заявлено)

Прокси

- Промежуточный сервер, позволяющий делать запросы через себя
- Бывают следующих видов:
  - HTTP – прокси, зачастую не поддерживают HTTPS и могут видеть весь пересылаемый через них трафик
    - Иногда дописывают заголовок X-Forwarded-For, выдавая IP клиента
    - Позволяют использовать только протокол HTTP (если не поддерживают HTTPS, иначе обычно все TCP-протоколы через метод CONNECT)
  - SOCKS – прокси, поддерживают все протоколы, иногда даже UDP
- Мы повстречаемся с прокси, когда будем знакомиться с программой Burp Suite

Чем работать с HTTP

- Curl – позволяет осуществлять различные запросы к HTTP серверам из командной строки (еще есть wget, но он не так удобен)
  - Этот способ удобен только для простых запросов, так как все данные запроса нужно писать в терминале как параметры командной строки
  - Если вы хотите посмотреть, как выглядит HTTP на уровне текстовых данных, очень рекомендую флаг -v
- Python Requests – библиотека для удобного осуществления запросов к HTTP-серверам из Python
  - Очень удобный способ работы с HTTP, если нужна автоматизация
- REST-клиенты – тип программ, позволяющий делать HTTP запросы вручную
  - Удобны если вы не знаете Python или хотите редактировать запросы через удобный пользовательский интерфейс, а не через код

Python Requests

- Как и другие библиотеки языка Python, эта может быть установлена через пакетный менеджер pip
- Пример GET-запроса:

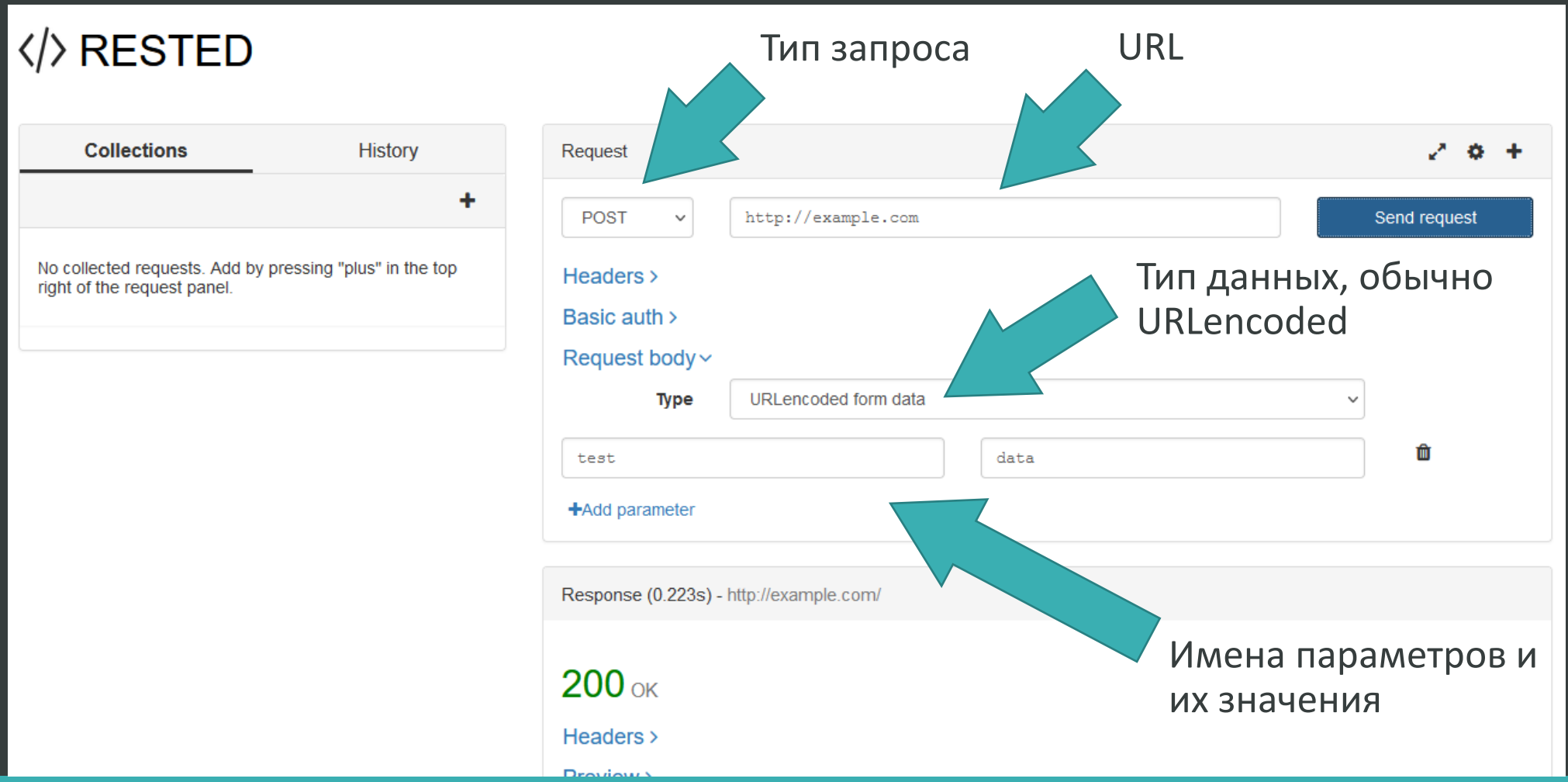
```
import requests
print(requests.get('http://example.com').text)
```

- Пример POST-запроса:

```
import requests
print(requests.post('http://example.com', data={'test': 'data'}).text)
```

- Конечно, возможности библиотеки requests этим не ограничиваются, поэтому рекомендую почитать документацию

REST-клиент RESTED (Firefox)



Особенности RESTED

- Можно установить из магазина расширений Firefox
- RESTED разделяет куки с основным браузером Firefox
  - Это означает, что вы можете сделать какой-нибудь запрос в RESTED, а его результат использовать в обычной сессии браузера
- RESTED использует по умолчанию тип данных JSON, не забудьте сменить его на URLencoded form data
  - Если, конечно, не работаете с JSON
- RESTED не очень хорош в отправке пустых JSON и вместо {} отправляет пустоту, что приводит к ошибкам
- Также в RESTED можно импортировать запросы формата HAR, в который можно экспортировать запросы из инструментов разработчика
  - Это делается при помощи ПКМ – «Копировать» – «Копировать все как HAR»

Interactive_Window:
C:\Program Files\Mozilla Firefox\firefox.exe "moz-extension://22d67434-c548-465d-ab90-384baf929c3d/dist/index.html"
"https://requestbin.com/r"
Mozilla Firefox

Спасибо за внимание!
Задачи доступны на

nsuctf.ru

- Пожалуйста, используйте имя пользователя формата «Фамилия Имя»
  - e-mail можно забить любой, сервером он не проверяется
- Для вопросов по задачам рекомендую присоединиться к @NSUCTF в Telegram
  - Только, пожалуйста, без спойлеров