

Лекция 2. Базовые сетевые протоколы и их безопасность

Сеть Интернет строится на основе стека TCP/IP — базового набора протоколов, который представляет собой реализацию модели OSI — разбиения сложной задачи передачи данных на подзадачи (уровни модели).

В компьютерных системах все данные (изображения, текст, звук и др.) представляются в виде последовательности байтов. При передаче данных через сеть они разбиваются на кусочки (сегменты, кадры, пакеты — в зависимости от уровня) и передаются по очереди.

Стек протоколов TCP/IP делится на уровни, которые вкладываются друг в друга:

1. Канальный уровень - физическая передача данных (Ethernet, Wi-Fi, L2TP)
2. Сетевой уровень - передача данных между физ. сетями (IPv4, IPv6, ICMP)
3. Транспортный уровень - установление базовых каналов передачи данных (TCP, UDP)
4. Прикладной уровень - протоколы приложений и сервисов (DNS, HTTP, SSH)

Протоколы каждого последующего уровня вкладываются в поле с данными протоколов низших уровней по принципу “матрёшки”.

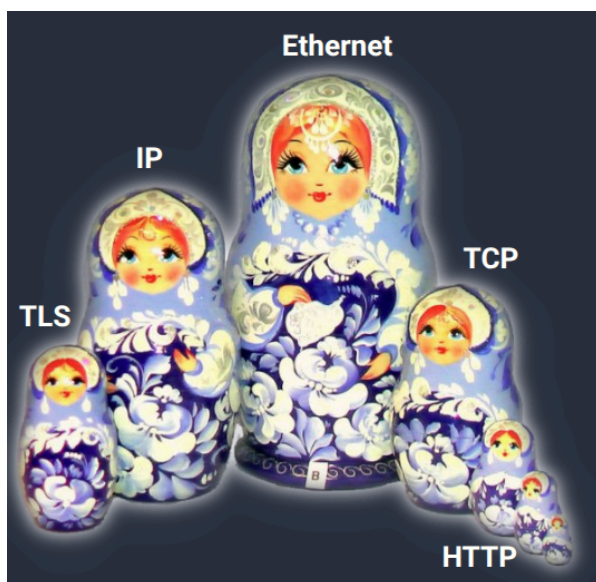


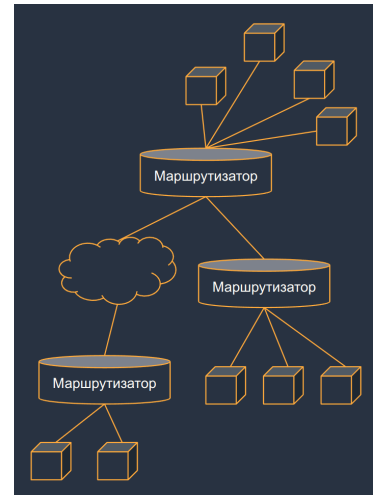
Рис. 1. Пример разбиения на уровни для протокола HTTPS

При желании можно даже устроить “рекурсию” и вложить весь стек протоколов TCP/IP в данные других протоколов, используя их как транспортные. Такая технология называется “туннелированием”. Используется в различных ситуациях, например при организации сетей VPN.

Протоколы канального уровня обеспечивают передачу данных, разделенных на кусочки-кадры, между физическими устройствами.

Протокол IP

Протоколы физического уровня позволяют передавать данные в пределах одной физической сети. Но как обеспечить передачу данных между несколькими сетями? Протокол IP (Internet Protocol) описывает порядок передачи пакетов (маршрутизацию) между физическими сетями. Для этого каждому узлу сети назначается адрес (IP-адрес, 32 бита в IPv4, 128 бит в IPv6), который для целей глобальной маршрутизации должен быть уникален (совпадение локальных адресов возможно при использовании технологии NAT) в пределах глобальной сети. Для отправки пакетов необходимо указывать адрес отправителя и адрес получателя. Пакеты передаются между физически связанными между собой устройствами (кабелем, беспроводным каналом, лазером, т.п.). Получив пакет, маршрутизатор, по таблице маршрутизации определяет какому устройству переслать пакет и этот пакет будет передаваться по цепочке маршрутизаторов пока не дойдёт до целевого IP-адреса.

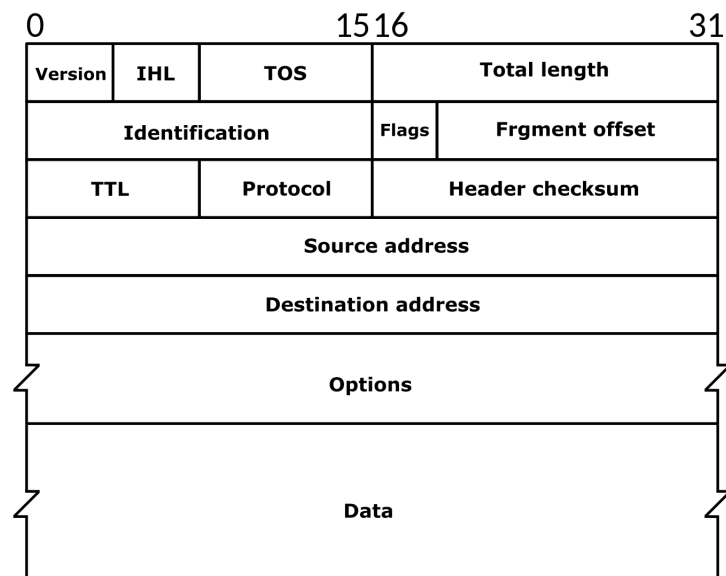


Преобразование текстовых строк, удобных для восприятия человеком (например, сайтов), в IP-адреса производится с помощью протокола более высокого уровня DNS.

Чтобы обеспечить передачу пакетов по физически сетям с разной пропускной способностью используется **фрагментация** - разбиение пакета на несколько более малых пакетов-фрагментов.

Протокол не гарантирует доставку пакетов, этим занимаются протоколы более высокого уровня.

Чтобы передать пакет необходимо сформировать заголовок с IP-адресом отправителя и получателя и данными.



Утилита Traceroute

Формат пакета IP.

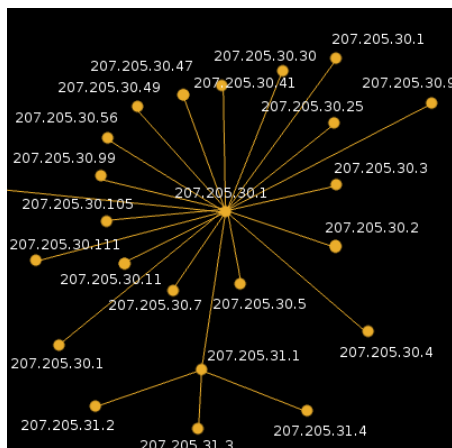
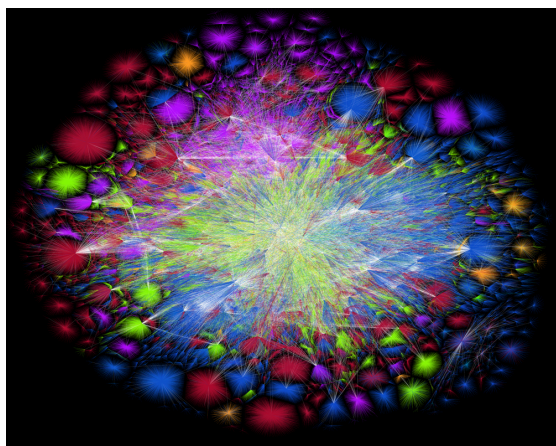
Проследить путь пакета до заданного узла позволяют инструменты traceroute/tracepath/tracert.

Они используют поле “время жизни пакета” (TTL) - число промежуточных хостов. Данное поле в протоколе IP предназначено для предотвращения бесконечной пересылки пакетов в случае, если в сети появится петля или цикл. TTL уменьшается на единицу при прохождении каждого узла. Если TTL = 0, то хост отправляет сообщение ICMP Time Exceeded и не пересылает дальше пакет.

Утилита traceroute использует данную особенность протокола для того, чтобы получить IP-адреса промежуточных узлов сети. Утилита отправляет последовательно пакеты с TTL=1,2,3,

```
n7v@n7v-laptop:~$ tracepath -b yandex.ru
1?: [LOCALHOST] pmtu 1400
1: _gateway (192.168.1.1) 1.343ms
1: _gateway (192.168.1.1) 0.780ms
2: 37-147-176-1.broadband.corbina.ru (37.147.176.1) 1.460ms
3: 81.211.111.30 (81.211.111.30) 1.861ms
4: spb-62-141-124-161.sovintel.spb.ru (62.141.124.161) 4.811ms
5: pe16.Moscow.gldn.net (79.104.235.207) 36.149ms
6: no reply
```

Промежуточные узлы возвращают пакет ICMP Time Exceeded с IP-адресом узла в адресе отправителя. Это работает только если промежуточный узел отправляет пакеты ICMP.



Протокол IP. Безопасность.

Можно в целом сказать, что функции для обеспечения безопасности, такие как шифрование, в протоколе IP отсутствуют. Безопасность обеспечивается уже на вышестоящих уровнях.

1. IP-спуфинг

В протоколе не проверяется корректность адреса отправителя/получателя, поэтому возможна отправка пакета от имени любого адреса, любому другому узлу сети (подмена адреса называется **IP-спуфингом**). При этом ответные пакеты от целевого узла будут отправляться на поддельный адрес.

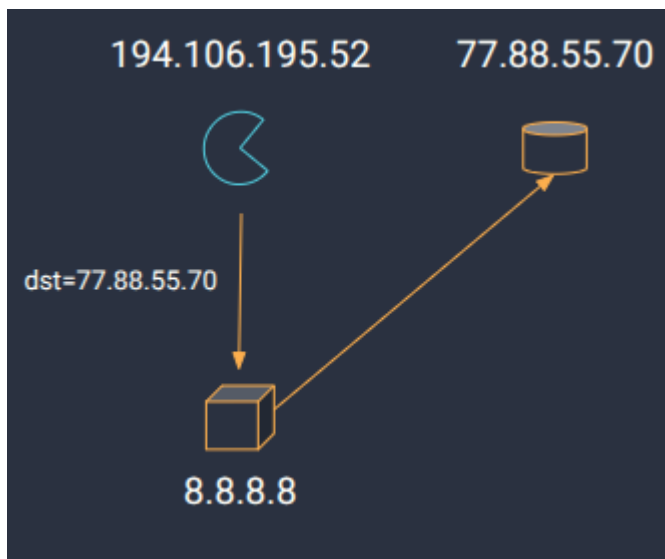


Рис. 2. Пример IP-спуфинга

Такая техника атаки может использоваться при проведении DoS-атак с целью сокрытия источника атаки и усиления атаки за счёт провоцирования отсылки больших ответных пакетов (например, ответов на DNS-запросы).

2. Фрагментация

В протоколе IP она предназначена для осуществления передачи через физические сети с разной пропускной способностью. Если требуется передать большой пакет и физическая сеть не поддерживает такие размеры пакетов, то пакет разбивается на несколько пакетов-фрагментов.

С точки зрения безопасности фрагментация может использоваться для обхода систем защиты, так как для анализа пакета его нужно собрать. Для этого требуется хранить фрагментированные пакеты в буфере, который имеет ограниченный размер. Хакер может посылать пакеты с задержками по времени или в разном порядке, что в нагруженных системах может привести к

переполнению буфера пакетов и пропуску непроверенных пакетов в сети (обход антивирусов, COB и фаерволов), либо блокировке легитимных пакетов (провоцирование DoS).

3. Ping-сканирование

В протоколе IPv4 сравнительно небольшое число адресов в подсетях. Поэтому перебор адресов получателя и отправка ping-запросов по протоколу ICMP позволяет идентифицировать узлы сети. В версии протокола IPv6 это уже сделать сложнее.

Порты

Протокол IP позволяет передать данные между узлами сети, но не позволяет распределять данные по сервисам (например, разным программам на одном компьютере). Для решения данной задачи служат **порты** — номера сервисов. Программы предварительно согласовывают номера портов, по которым будут передавать данные и указывают их в заголовках протоколов. Есть стандартные порты, по которым висит тот или иной сервис.

80	HTTP/Skype
443	HTTPS
22	SSH

Рис. 3. Примеры стандартных портов

Кроме того, один из портов может согласовываться в процессе установления соединения (например, в протоколе TCP).

Транспортный уровень

Протокол UDP

Протокол UDP по сути своей добавляет концепцию портов к протоколу IP и обеспечивает передачу данных между сервисами без каких-либо дополнительных функции и со всеми недостатками протокола IP, таких как отсутствие гарантии доставки пакетов. Используется так, где потери пакетов не критичны и важна скорость передачи данных.

Протокол TCP

Основной транспортный протокол сети Интернет. Поддерживает передачу данных между сервисами и создание соединений с гарантией доставки пакетов в правильном порядке.

Основной принцип, на котором работает гарантия доставки: если не получено подтверждение доставки от получателя (ответный пакет с флагом ACK) в течении некоторого времени, то отослать пакет заново.

Утилита netcat

Для изучения устройства сети Интернет будем использовать консольную утилиту netcat, которая реализует подключение по TCP/UDP и отсылку данных, введенных в консоли. В различных ОС имеет разные названия и реализации:

- nc в Linux
- ncat из состава Nmap в Windows/Linux

Позволяет подключиться к любому компьютеру/серверу в сети Интернет.

Пример. Чат по TCP/IP между двумя компьютерами

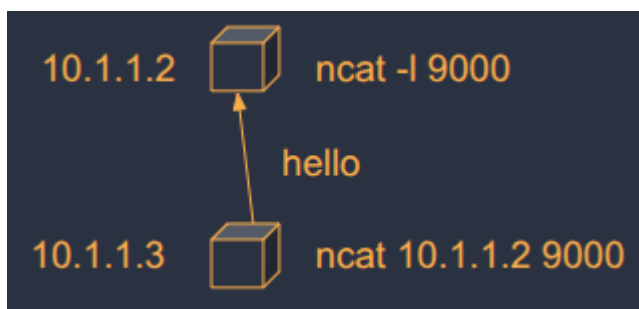


Рис. 4. Чат по netcat. Схема

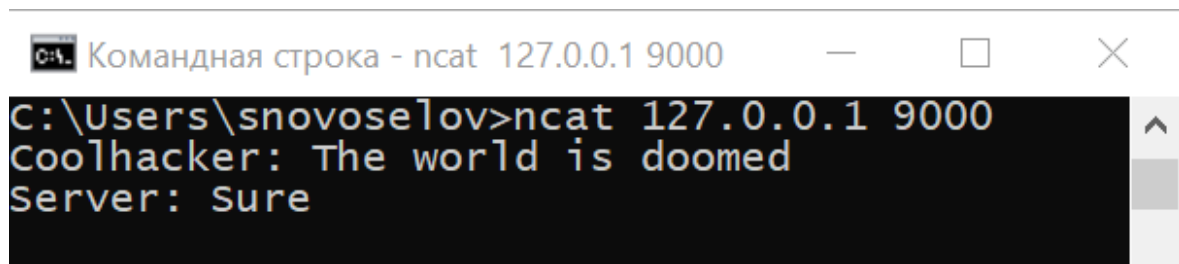
1. Для создания соединения следует один из компьютеров выбрать в качестве сервера:

```
Командная строка - ncat -l 9000
C:\Users\snoovse1ov>ncat -l 9000
Coolhacker: The world is doomed
Server: Sure
```

Рис. 5. Чат по netcat. Консоль сервера

На рисунке выбран порт 9000. Допустимо выбирать любое значение.

2. Второй компьютер к нему подключается:



```
C:\Users\snovoselov>ncat 127.0.0.1 9000
Coolhacker: The world is doomed
Server: Sure
```

Рис. 6. Чат по netcat. Консоль клиента

IP-адрес сервера (на рисунке 127.0.0.1), может быть любой адрес в Интернете.

3. После подключения все введенные данные отправляются в обе стороны

Связь работает в рамках локальной сети. Например, если компьютеры подключены к одной Wi-Fi точке доступа и нет изоляции клиентов. Для связи по сети Интернет необходимо, чтобы один из компьютеров имел внешний (не локальный) IP-адрес.

Утилита netcat может использоваться в качестве полезной нагрузки эксплойтов. В этом случае, к ней подключается ввод/вывод командного интерпретатора (sh, bash, dash и т.п.) и хакер может подключиться для ввода команд.

Анализатор сетевого трафика Wireshark

Для просмотра пакетов используются анализаторы сетевого трафика.



Wireshark - наиболее известный и продвинутый анализатор (www.wireshark.org).

Данный анализатор поддерживает почти все известные протоколы.

Захват из Adapter for loopback traffic capture

Файл Редактирование Просмотр Запуск Захват Анализ Статистика Телефония Беспроводной Инструменты Помощь

tcp.port == 9000 && ip.addr == 127.0.0.1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	1571 → 9000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=25
2	0.000041	127.0.0.1	127.0.0.1	TCP	56	9000 → 1571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=
3	0.000065	127.0.0.1	127.0.0.1	TCP	44	1571 → 9000 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
4	12.720132	127.0.0.1	127.0.0.1	S101	76	1571 → 9000 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=32
5	12.720197	127.0.0.1	127.0.0.1	TCP	44	9000 → 1571 [ACK] Seq=1 Ack=33 Win=2619648 Len=0
6	19.551216	127.0.0.1	127.0.0.1	S101	57	9000 → 1571 [PSH, ACK] Seq=1 Ack=33 Win=2619648 Len=13
7	19.551305	127.0.0.1	127.0.0.1	TCP	44	1571 → 9000 [ACK] Seq=33 Ack=14 Win=2619648 Len=0
12	56.725784	127.0.0.1	127.0.0.1	TCP	44	9000 → 1571 [RST, ACK] Seq=14 Ack=33 Win=0 Len=0

> Frame 1: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{Loopback}, id 0

Null/Loopback

Family: IP (2)

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 1571, Dst Port: 9000, Seq: 0, Len: 0

```

0000  02 00 00 00 45 00 00 34 84 b0 40 00 80 06 00 00  ...E..4..@....
0010  7f 00 00 01 7f 00 00 01 06 23 23 28 52 d6 bc da  ....##(R...
0020  00 00 00 00 80 02 ff ff 3d ee 00 00 02 04 ff d7  ....=.....
0030  01 03 03 08 01 01 04 02  ....

```

Family (null.family), 4 байт

Пакеты: 24 · Показаны: 8 (33.3%)

Профиль: Default

Рис. 7. Пакеты TCP для примера с чатом.

Так как в протоколе TCP не предусмотрено шифрование, то как видно из рисунка все сообщения чата можно прочесть имея возможность прослушивать данные (как это происходит, например, в открытой сети Wi-Fi).

4 12.720132 127.0.0.1 127.0.0.1 S101 76 1571 → 9000 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=32

> Frame 4: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface \Device\NPF_{Loopback}, id 0

Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 1571, Dst Port: 9000, Seq: 1, Ack: 1, Len: 32

> Data (32 bytes)

Data: 436f666c6861636b65723a2054686520776f726c642069732064666f6d65640a

[Length: 32]

```

0000  02 00 00 00 45 00 00 48 84 b3 40 00 80 06 00 00  ...E..H..@....
0010  7f 00 00 01 7f 00 00 01 06 23 23 28 52 d6 bc db  ....##(R...
0020  e8 78 3a a6 50 18 27 f9 3d 1e 00 00 43 6f 6f 6c  ...x:P.'.'...Cool
0030  68 61 63 6b 65 72 3a 20 54 68 65 20 77 6f 72 6c  hacker: The worl
0040  64 20 69 73 20 64 6f 6f 6d 65 64 0a  ....d is doo med.

```

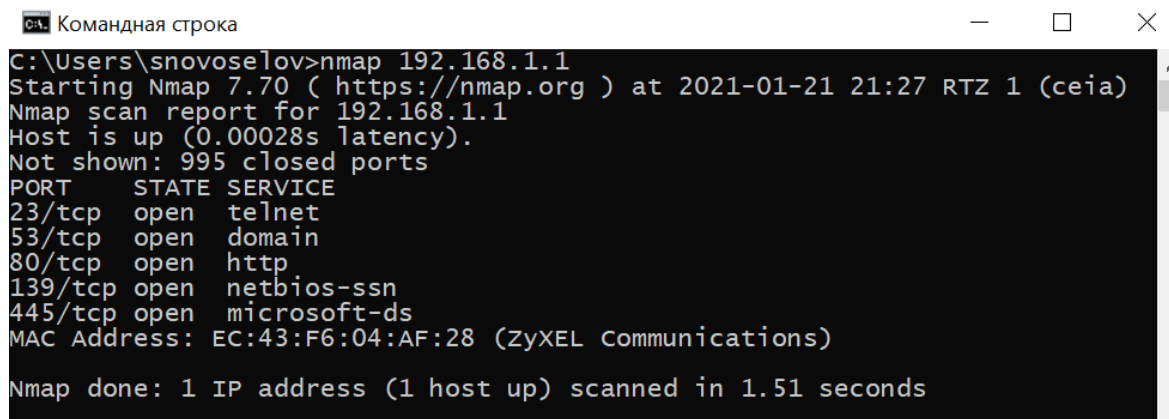
Рис. 8. Пакет TCP сообщением от компьютера-клиента.

Для шифрования можно использовать TLS-протокол, который вкладывается в TCP. Есть версии netcat с поддержкой данной функции.

Сканирование портов

В протоколе TCP/UDP предусмотрено всего 65535 портов. Так как число портов небольшое, то для нахождения сервисов и программ доступных на целевой системе можно использовать перебор. Для сканирования есть множество различных техник с целью добиться ответа от системы в том числе при наличии средств защиты. В основном техники основаны на манипуляциях

с полями в заголовке протокола.



```
C:\Users\snoyoselov>nmap 192.168.1.1
Starting Nmap 7.70 ( https://nmap.org ) at 2021-01-21 21:27 RTZ 1 (ceia)
Nmap scan report for 192.168.1.1
Host is up (0.00028s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
23/tcp    open  telnet
53/tcp    open  domain
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
MAC Address: EC:43:F6:04:AF:28 (ZyXEL Communications)

Nmap done: 1 IP address (1 host up) scanned in 1.51 seconds
```

Рис. 9. Сканирование портов. Пример для роутера

Как видно из примера здесь есть telnet. Можно попытаться подобрать пароль:
`nmap -p 23 --script telnet-brute 192.168.1.1`

Определение версий ПО

Определение версии ОС может осуществляться на основе:

- анализа ответов хоста на нестандартные IP/UDP/TCP пакеты
- значений по-умолчанию для заголовков пакетов (например, размер окна TCP)
- используемых начальных значений последовательностей
- статистических характеристик генераторов случайных чисел в ответах узла

Определение версий сервисов и программ как правило выполняется по баннеру — часто при подключении сервер возвращает строку-баннер с версией пример: “Server: nginx/1.4.2”, либо по “отпечаткам” — ответам на специально составленные запросы.

Сканер nmap содержит реализацию большинства техник и вместе с базами запросов (nmap-service-probes). После определения версий ПО, их можно проверить по базам уязвимостей и эксплойтов (NVD, exploit-db.com).

На следующем рисунке представлен пример определения версий сервисов на машине с Metasploitable 2 с помощью команды:

```
nmap -sV 192.168.1.45
```

```

Nmap scan report for 192.168.1.45
Host is up (0.000015s latency).
Not shown: 977 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  tcpwrapped
1099/tcp  open  rmiregistry  GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  x11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
Service Info: Hosts: metasploitable.localdomain, localhost, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

```

Рис. 10. Определение версий с помощью Nmap

Прикладные протоколы

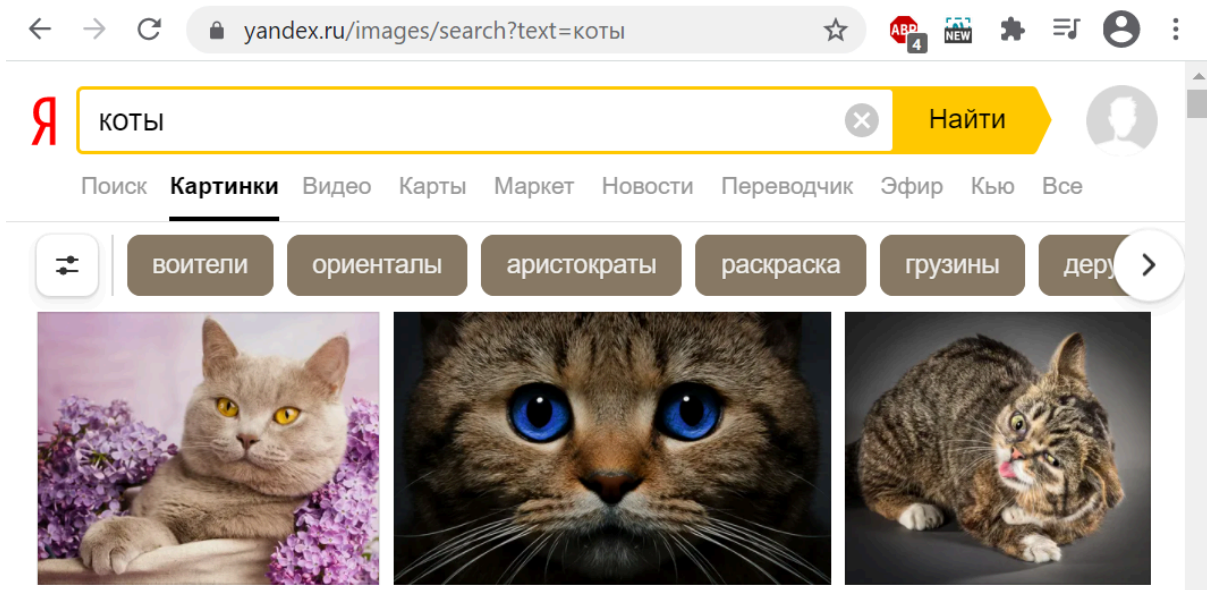
Основные прикладные протоколы, используемые в сети Интернет:

- DNS (Domain Name System) — преобразование удобных для человека строк (доменных имён) в IP-адреса.
- HTTP — протокол лежащий в основе сети World Wide Web (WWW)
- HTTPS — протокол HTTP, вложенный в протокол TLS

Для примера, рассмотрим что происходит при открытии ссылки

yandex.ru/images/search?text=коты

в браузере. Для отрисовки страницы как на следующем рисунке:



браузер выполняет большое количество действий. Сначала браузер разбивает ссылку (URL) на части:

- yandex.ru - имя хоста
- /images/search?text=коты - путь и параметры запроса

Затем браузер преобразует yandex.ru в IP-адрес 77.88.55.50 с помощью DNS-протокола. После этого подключается по протоколу TCP к IP-адресу 77.88.55.50 с портом 80 (HTTP) или 443 (HTTPS, в этом случае дополнительно используется TLS) и посылает строку (HTTP-запрос) вида:

```
GET /images/search?text=%D0%BA%D0%BE%D1%82%D1%8B HTTP/1.1
Host: yandex.ru
```

В ответ получает HTTP-заголовки со служебной информацией:

```
HTTP/1.1 200 OK
Cache-Control: private, max-age=3600
Content-Type: text/html; charset=utf-8
...
Date: Thu, 21 Jan 2021 15:35:47 GMT
```

и данные ответа (html-код):

```
<!DOCTYPE html><html class="i-ua_js_no i-ua_css_standard"
lang="ru">
<head><meta charset="utf-8"><meta http-equiv="X-UA-Compatible"
content="IE=edge"><title>Яндекс.Картинки</title>
```

...

Получив html-код, браузер разбирает и отрисовывает его, подгружая аналогичным образом картинки (GET-запросы HTTP) и другие ресурсы.

Протокол HTTP

Протокол HTTP вкладывается в протоколы TCP (порт 80) или TLS (порт 443). После подключения к серверу по одному из этих протоколов клиент передает текстовые строки вида:

МЕТОД ПАРАМЕТРЫ

ЗАГОЛОВКИ

ПУСТАЯ СТРОКА

ДАННЫЕ

Заголовки передаются в формате:

КЛЮЧ: ЗНАЧЕНИЕ

и содержат в себе дополнительные параметры запроса или настройки протокола.

Основные HTTP-методы: GET/POST/HEAD.

- GET — запрос данных
- HEAD — запрос только заголовков, без тела ответа
- POST — отправка данных на сервер (пример: формы, файлы)

Сервер отвечает в ответ такой же текстовой строкой, только вместо строки с методом передается строка статуса.

Протокол HTTP не хранит состояние, т.е. сервер не хранит информацию о прошлых HTTP-запросах. Информацию нужно передавать в заголовках запроса (Cookie) и/или её запоминают в базе скрипты, вызываемые сервером (пример: интерпретаторы PHP, Ruby, Python могут записывать данные в базу MySQL).

Протокол HTTP. Безопасность

Шифрование

Протокол не включает в себя шифрование. Поэтому например, в публичной Wi-Fi сети все данные передаваемые по чистому протоколу HTTP можно перехватить и это один из каналов утечки паролей/логинов пользователей при заходе на сайт.

Для защиты необходимо использовать протокол HTTPS, который представляет собой протокол HTTP, вложенный в протокол TLS.

Параметры скриптов

Для обработки запроса на сервере используются скрипты (PHP/Python/Ruby), куда передаются параметры запроса (пример, text=коты). Ошибки в работе скриптов и обработке передаваемых данных — основной источник уязвимостей на сайтах. Как правило, уязвимости находятся подстановкой нестандартных значений в параметры HTTP-запросов.

Литература и ссылки

- Дуглас Э. Камер - Сети TCP IP. Принципы, протоколы и структура (2003)
- Metasploitable 2 и 3 - системы для тренировки
- <https://sourceforge.net/projects/metasploitable/>
- <https://github.com/brimstone/metasploitable3/releases>
- Gordon Lyon. Nmap Network Scanning. The Official Nmap Project Guide to Network Discovery and Security Scanning (2011) (<http://nmap.org/book/>)
- Список скриптов Nmap:
<https://nmap.org/nsedoc/>