Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №3

Атаки при установке TCP-соединения и протоколов прикладного уровня.

Выполнил:
Брычиков Д.Д.
Проверил:
Чернявский Ю. А.

Минск 2018

Введение

Адресация в сети Internet.

Типы адресов.

Типы адресов:

1. **Физический (MAC-адрес)**
2. **Сетевой (IP-адрес)**
3. **Символьный (DNS-имя)**

Компьютер в сети TCP/IP может иметь адреса трех уровней (но не менее двух):

- Локальный адрес компьютера. Для узлов, входящих в локальные сети - это MAC-адрес сетевого адаптера. Эти адреса назначаются производителями оборудования и являются уникальными адресами.
- IP-адрес, состоящий из 4 байт, например, 109.26.17.100. Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов.
- Символьный идентификатор-имя (DNS), например, www.kstu.ru.

IP-адреса

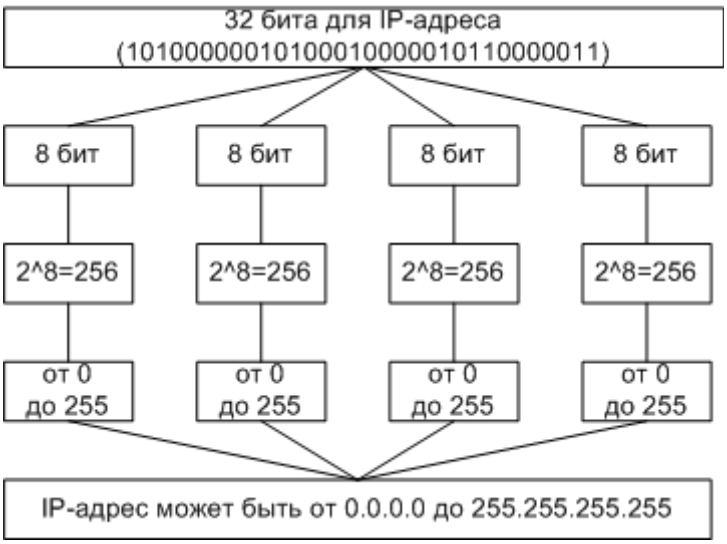
IPv4 - адрес является уникальным 32-битным идентификатором IP-интерфейса в Интернет.

IPv6 - адрес является уникальным 128-битным идентификатором IP-интерфейса в Интернет, иногда называют **Internet-2,** адресного пространства IPv4 уже стало не хватать, поэтому постепенно вводят новый стандарт.

IP-адреса принято записывать разбивкой всего адреса по октетам (8), каждый октет записывается в виде десятичного числа, числа разделяются точками. Например, адрес

10100000010100010000010110000011
записывается как

10100000.01010001.00000101.10000011 = 160.81.5.131



Перевод адреса из двоичной системы в десятичную

IP-адрес хоста состоит из номера IP-сети, который занимает старшую область адреса, и номера хоста в этой сети, который занимает младшую часть.

160.81.5.131 - IP-адрес

160.81.5. - номер сети

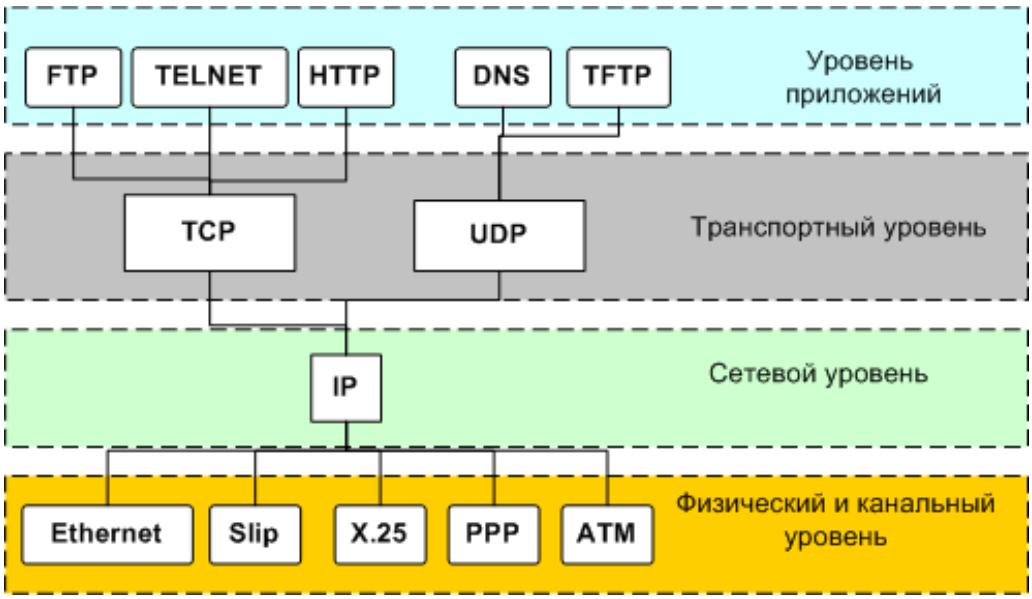
131 - номер хоста

Базовые протоколы (IP, TCP)

Стек протоколов TCP/IP

TCP/IP - собирательное название для набора (стека) сетевых протоколов разных уровней, используемых в Интернет. Особенности TCP/IP:

- Открытые стандарты протоколов, разрабатываемые независимо от программного и аппаратного обеспечения;
- Независимость от физической среды передачи;
- Система уникальной адресации;
- Стандартизованные протоколы высокого уровня для распространенных пользовательских сервисов.



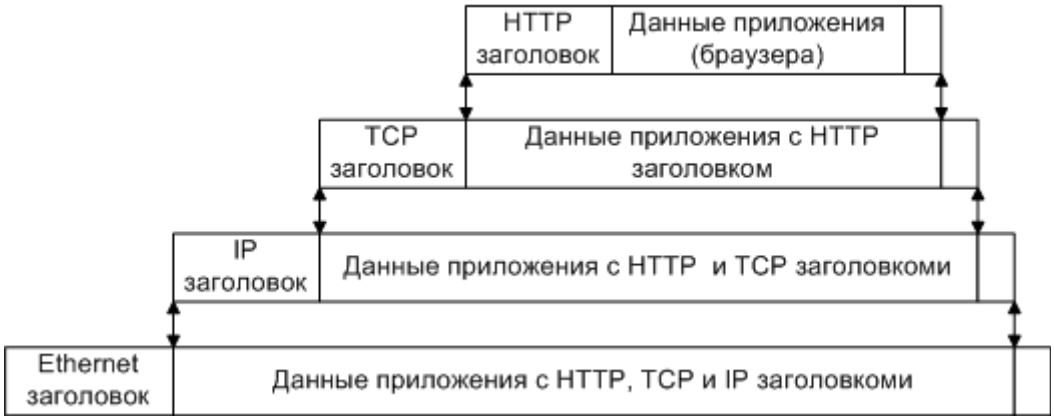
Стек протоколов TCP/IP

Стек протоколов TCP/IP делится на 4 уровня:

- Прикладной,
- Транспортный,
- Межсетевой,
- Физический и канальный.

Позже была принята 7-ми уровневая модель ISO.

Данные передаются в пакетах. Пакеты имеют заголовок и окончание, которые содержат служебную информацию. Данные, более верхних уровней вставляются, в пакеты нижних уровней.



Пример инкапсуляции пакетов в стеке TCP/IP

Физический и канальный уровень.

Стек TCP/IP не подразумевает использования каких-либо определенных протоколов уровня доступа к среде передачи и физических сред передачи данных. От уровня доступа к среде передачи требуется наличие интерфейса с модулем IP, обеспечивающего передачу IP-пакетов. Также требуется обеспечить преобразование IP-адреса узла сети, на который передается IP-пакет, в MAC-адрес. Часто в качестве уровня доступа к среде передачи могут выступать целые протокольные стеки, тогда говорят об IP поверх ATM, IP поверх IPX, IP поверх X.25 и т.п.

Межсетевой уровень и протокол IP.

Основу этого уровня составляет IP-протокол.

IP (Internet Protocol) – интернет протокол.

Первый стандарт IPv4 определен в RFC-760 (DoD standard Internet Protocol J. Postel Jan-01-1980)

Последняя версия IPv6 - RFC-2460 (Internet Protocol, Version 6 (IPv6) Specification S. Deering, R. Hinden December 1998).

Основные задачи:

- Адресация
- Маршрутизация
- Фрагментация датаграмм
- Передача данных

Протокол IP доставляет блоки данных от одного IP-адреса к другому.

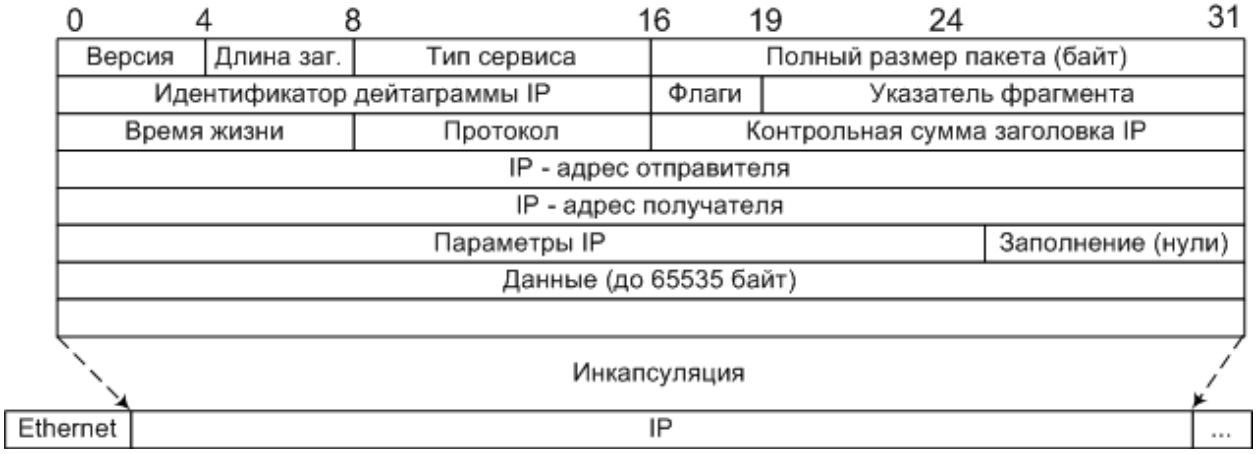
Программа, реализующая функции того или иного протокола, часто называется модулем, например, “IP-модуль”, “модуль TCP”.

Когда модуль IP получает IP-пакет с нижнего уровня, он проверяет IP-адрес назначения.

- Если IP-пакет адресован данному компьютеру, то данные из него передаются на обработку модулю вышестоящего уровня (какому конкретно - указано в заголовке IP-пакета).
- Если же адрес назначения IP-пакета - чужой, то модуль IP может принять два решения: первое - уничтожить IP-пакет, второе - отправить его дальше к месту назначения, определив маршрут следования - так поступают маршрутизаторы.

Также может потребоваться, на границе сетей с различными характеристиками, разбить IP-пакет на фрагменты (**фрагментация**), а потом собрать в единое целое на компьютере-получателе.

Если модуль IP по какой-либо причине не может доставить IP-пакет, он уничтожается. При этом модуль IP может отправить компьютеру-источнику этого IP-пакета уведомление об ошибке; такие уведомления отправляются с помощью протокола **ICMP**, являющегося неотъемлемой частью модуля IP. Более никаких средств контроля корректности данных, подтверждения их доставки, обеспечения правильного порядка следования IP-пакетов, предварительного установления соединения между компьютерами протокол IP не имеет. Эта задача возложена на транспортный уровень.



Структура дейтограммы IP. Слова по 32 бита.

Версия - версия протокола IP (например, 4 или 6)

Длина заг. - длина заголовка IP-пакета.

Тип сервиса (TOS - type of service) - Тип сервиса (подробнее рассмотрен в лекции 8).

TOS играет важную роль в маршрутизации пакетов. Интернет не гарантирует запрашиваемый TOS, но многие маршрутизаторы учитывают эти запросы при выборе маршрута (протоколы OSPF и IGRP).

Идентификатор дейтаграммы, флаги (3 бита) и указатель фрагмента - используются для распознавания пакетов, образовавшихся путем фрагментации исходного пакета.

Время жизни (TTL - time to live) - каждый маршрутизатор уменьшает его на 1, что бы пакеты не блуждали вечно.

Протокол - Идентификатор протокола верхнего уровня указывает, какому протоколу верхнего уровня принадлежит пакет (например: TCP, UDP).

Коды некоторые протоколов RFC-1700 (1994)

Код	Протокол	Описание
0	-	Зарезервировано
1	ICMP	Протокол контрольных сообщений
2	IGMP	Групповой протокол управления
4	IP	IP-поверх-IP (туннели)
6	TCP	Протокол управления передачей
8	EGP	Протокол внешней маршрутизации

9	IGP	Протокол внутренней маршрутизации
17	UDP	Протокол дейтограмм пользователя
35	IDRP	Междоменный протокол маршрутизации
36	XTP	Xpress транспортный протокол
46	RSVP	Протокол резервирования ресурсов канала
88	IGRP	внутренний протокол маршрутизации
89	OSPFIGP	внутренний протокол маршрутизации
97	ETHERIP	Ethernet-поверх-IP
101-254	-	не определены
255	-	зарезервировано

Маршрутизация.

Протокол IP является маршрутизируемый, для его маршрутизации нужна маршрутная информация.

Маршрутная информация, может быть:

- Статической (маршрутные таблицы прописываются вручную)
- Динамической (маршрутную информацию распространяют специальные протоколы)

Транспортный уровень

Протоколы транспортного уровня обеспечивают прозрачную доставку данных между двумя прикладными процессами. Процесс, получающий или отправляющий данные с помощью транспортного уровня, идентифицируется на этом уровне номером, который называется номером порта. Таким образом, роль адреса отправителя и получателя на транспортном уровне выполняет номер порта (или проще - порт).

Анализируя заголовок своего пакета, полученного от межсетевого уровня, транспортный модуль определяет по номеру порта получателя, какому из прикладных процессов направлены данные, и передает эти данные соответствующему прикладному процессу. Номера портов получателя и отправителя записываются в заголовок транспортным модулем, отправляющим данные; заголовок транспортного уровня содержит также и другую служебную информацию; формат заголовка зависит от используемого транспортного протокола.

На транспортном уровне работают два основных протокола: UDP и TCP.

Протокол надежной доставки сообщений TCP

TCP (Transfer Control Protocol) – протокол контроля передачи, протокол TCP применяется в тех случаях, когда требуется гарантированная доставка сообщений.

Первая и последняя версия TCP - RFC-793 (Transmission Control Protocol J. Postel Sep-01-1981).

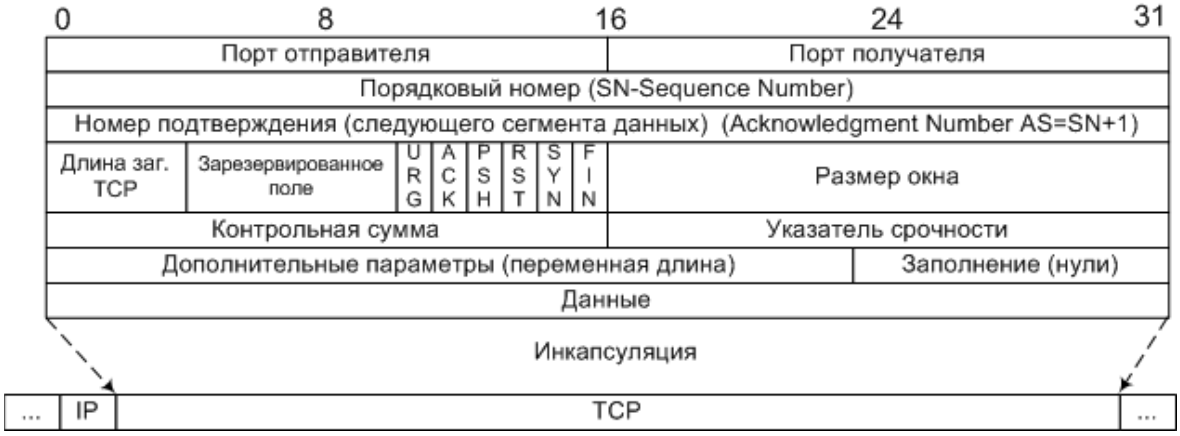
Основные особенности:

- Устанавливается соединение.

- Данные передаются **сегментами**. Модуль TCP нарезает большие сообщения (файлы) на пакеты, каждый из которых передается отдельно, на приемнике наоборот файлы собираются. Для этого нужен **порядковый номер (Sequence Number - SN)** пакета.

- Посылает запрос на следующий пакет, указывая его номер в поле **"Номер подтверждения" (AS).** Тем самым, подтверждая получение предыдущего пакета.

- Делает проверку целостности данных, если пакет битый посылает повторный запрос.



Структура дейтограммы TCP. Слова по 32 бита.

Длина заголовка - задается словами по 32бита.

Размер окна - количество байт, которые готов принять получатель без подтверждения.

Контрольная сумма - включает псевдо заголовок, заголовок и данные.

Указатель срочности - указывает последний байт срочных данных, на которые надо немедленно реагировать.

URG - флаг срочности, включает поле "Указатель срочности", если =0 то поле игнорируется.

ACK - флаг подтверждение, включает поле "Номер подтверждения, если =0 то поле игнорируется.

PSH - флаг требует выполнения операции push, модуль TCP должен срочно передать пакет программе.

RST - флаг прерывания соединения, используется для отказа в соединении

SYN - флаг синхронизация порядковых номеров, используется при установлении соединения.

FIN - флаг окончание передачи со стороны отправителя

Назначение портов

По номеру порта транспортные протоколы определяют, какому приложению передать содержимое пакетов.

Порты могут принимать значение от 0-65535 (два байта 2^16).

Номера портам присваиваются таким образом: имеются стандартные номера (например, номер 21 закреплен за сервисом FTP, 23 - за telnet, 80 - за HTTP), а менее известные приложения пользуются произвольно выбранными локальными номерами (как правило, больше>1024), некоторые из них также зарезервированы.

Программа Ping

Программа для проверки соединения и работы с удаленным хостом.

Программа TraceRoute - позволяет проверить маршрут до удаленного хоста.

Программа nmap - позволяет сканировать порты.

Работу порта, также можно проверить с помощью telnet.

Некоторые заданные порты RFC-1700 (1994) 43%

Порт	Служба	Описание
0	-	Зарезервировано
13	Daytime	Синхронизация времени

20	ftp-data	Канал передачи данных для FTP
21	ftp	Передача файлов
23	telnet	Сетевой терминал
25	SMTP	Передача почты
37	time	Синхронизация времени
43	Whois	Служба Whois
53	DNS	Доменные имена
67	bootps	BOOTP и DHCP - сервер
68	bootps	BOOTP и DHCP - клиент
69	tftp	Упрощенная передача почты
80	HTTP	Передача гипертекста
109	POP2	Получение почты
110	POP3	Получение почты
119	NNTP	Конференции
123	NTP	Синхронизация времени
137	netbios-ns	NETBIOS - имена
138	netbios-dgm	NETBIOS Datagram Service
139	netbios-ssn	NETBIOS Session Service
143	imap2	Получение почты
161	SNMP	Протокол управления
210	z39.50	Библиотечный протокол
213	IPX	IPX - протокол
220	imap3	Получение почты
443	HTTPs	HTTP с шифрованием
520	RIP	Динамическая маршрутизация
Диапазон 1024-65535		
1024	-	Зарезервировано
6000-6063	X11	Графический сетевой терминал

Постановка задачи

1) Изучить теоретические сведения.

2) Создать приложение, реализующее атаки на протокол при установке TCP-соединения и в рамках заданного протокола прикладного уровня.

В интерфейсе приложения должны быть наглядно представлены:

- Исходные данные протокола (модули, ключи, флаги, иные данные);
- Данные, передаваемые по сети каждой из сторон;
- Проверки, выполняемые каждым из участников.
- Процесс взаимодействия между сторонами протокола может быть реализован при помощи буферных переменных.
- Также необходимо выделить каждый из этапов атаки для того, чтобы его можно было отделить от остальных.

WIRESHARK ДЛЯ АНАЛИЗА ТРАФИКА

Wireshark — это мощный сетевой анализатор, который может использоваться для анализа трафика, проходящего через сетевой интерфейс вашего компьютера. Он может понадобиться для обнаружения и решения проблем с сетью, отладки ваших веб-приложений, сетевых программ или сайтов. Wireshark позволяет полностью просматривать содержимое пакета на всех уровнях: так вы сможете лучше понять как работает сеть на низком уровне.

Все пакеты перехватываются в реальном времени и предоставляются в удобном для чтения формате. Программа поддерживает очень мощную систему фильтрации, подсветку цветом, и другие особенности, которые помогут найти нужные пакеты. В этой инструкции мы рассмотрим, как пользоваться Wireshark для анализа трафика. Недавно разработчики перешли к работе над второй веткой программы Wireshark 2.0, в неё было внесено множество изменений и улучшений, особенно для интерфейса. Именно её мы будем использовать в этой статье.

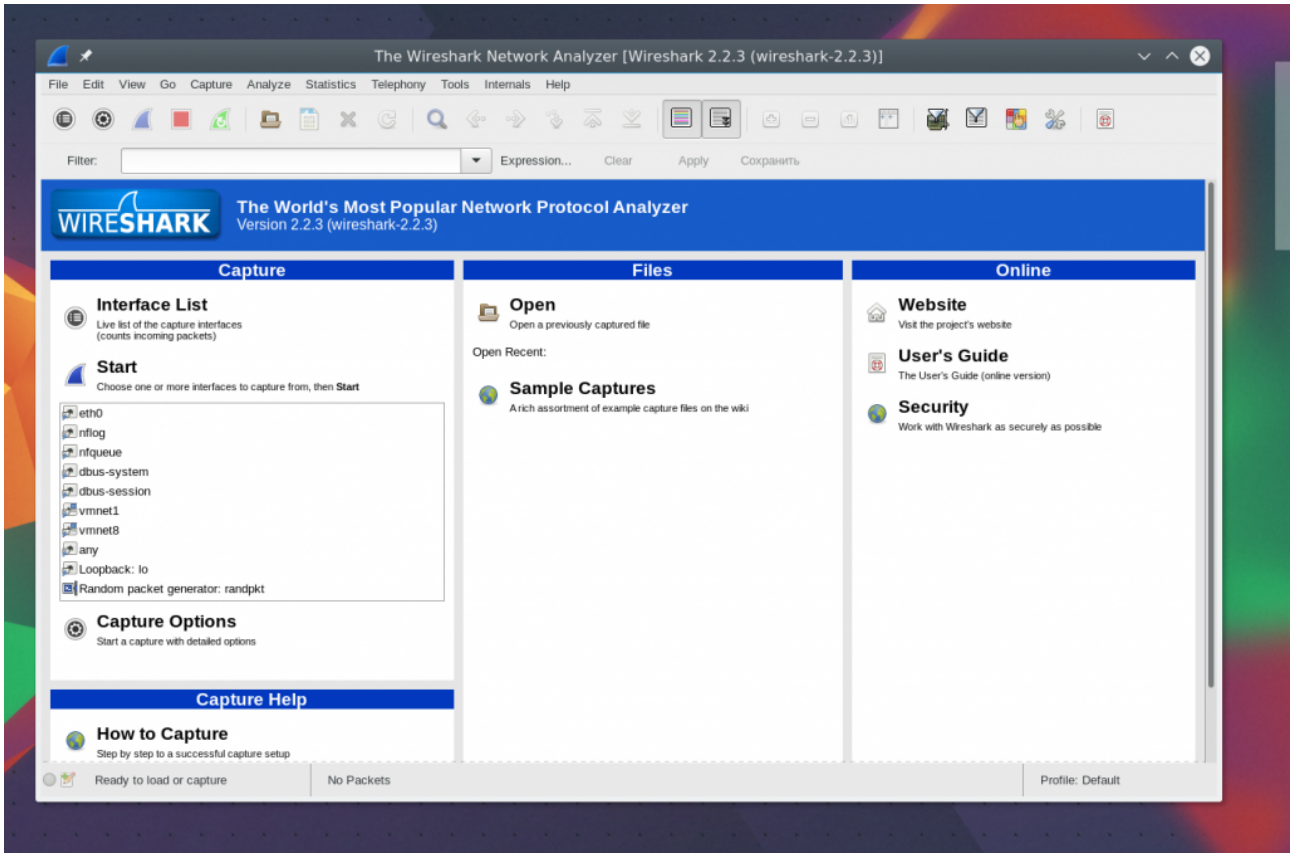
ОСНОВНЫЕ ВОЗМОЖНОСТИ WIRESHARK

Перед тем, как переходить к рассмотрению способов анализа трафика, нужно рассмотреть, какие возможности поддерживает программа более подробно, с какими протоколами она может работать и что делать. Вот основные возможности программы:

- Захват пакетов в реальном времени из проводного или любого другого типа сетевых интерфейсов, а также чтение из файла;
- Поддерживаются такие интерфейсы захвата: Ethernet, IEEE 802.11, PPP и локальные виртуальные интерфейсы;
- Пакеты можно отсеивать по множеству параметров с помощью фильтров;
- Все известные протоколы подсвечиваются в списке разными цветами, например TCP, HTTP, FTP, DNS, ICMP и так далее;
- Поддержка захвата трафика VoIP-звонков;
- Поддерживается расшифровка HTTPS-трафика при наличии сертификата;
- Расшифровка WEP-, WPA-трафика беспроводных сетей при наличии ключа и handshake;
- Отображение статистики нагрузки на сеть;
- Просмотр содержимого пакетов для всех сетевых уровней;

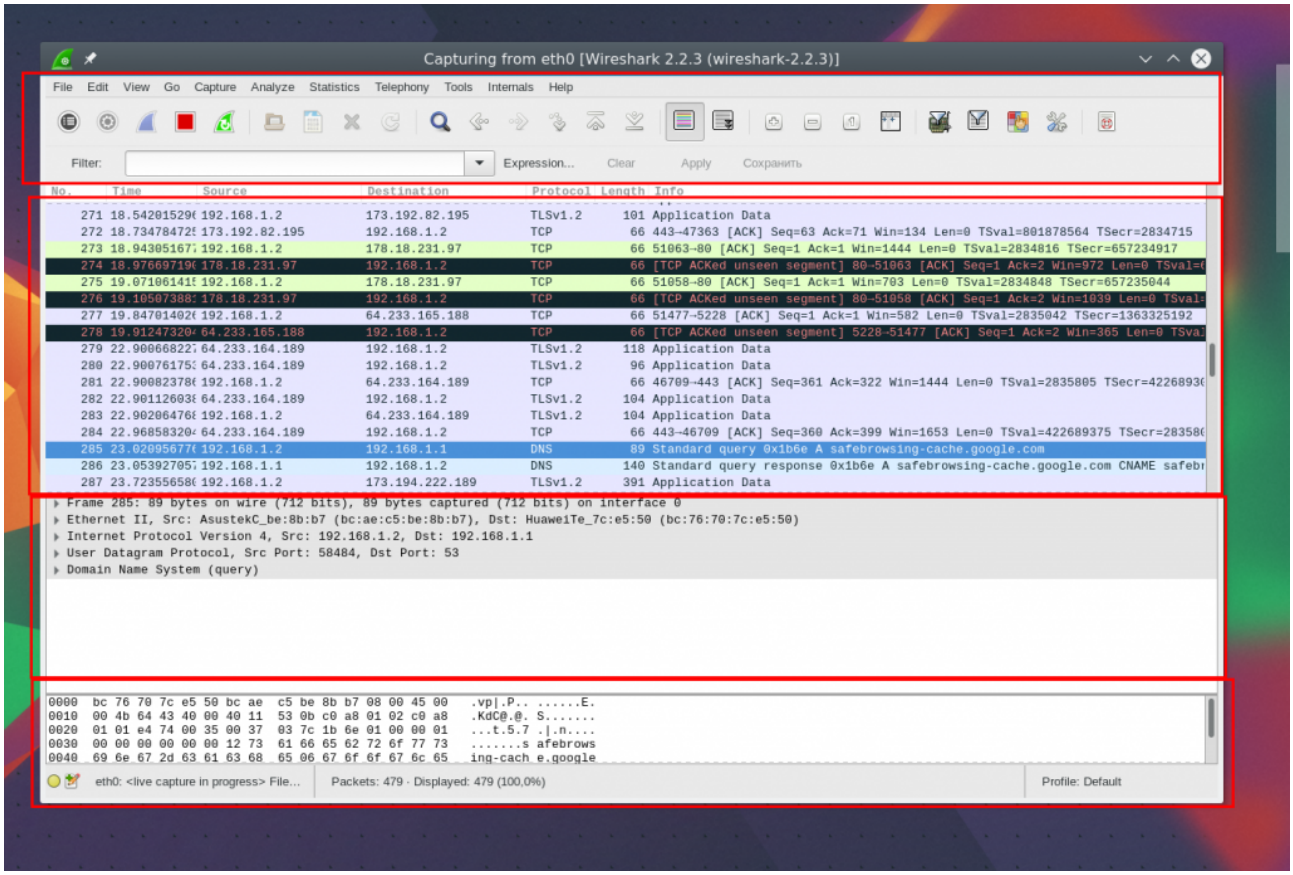
- Отображение времени отправки и получения пакетов.
- Программа имеет множество других функций, но это были те основные, которые могут вас заинтересовать.

Главное окно программы разделено на три части: первая колонка содержит список доступных для анализа сетевых интерфейсов, вторая — опции для открытия файлов, а третья — помощь.



АНАЛИЗ СЕТЕВОГО ТРАФИКА

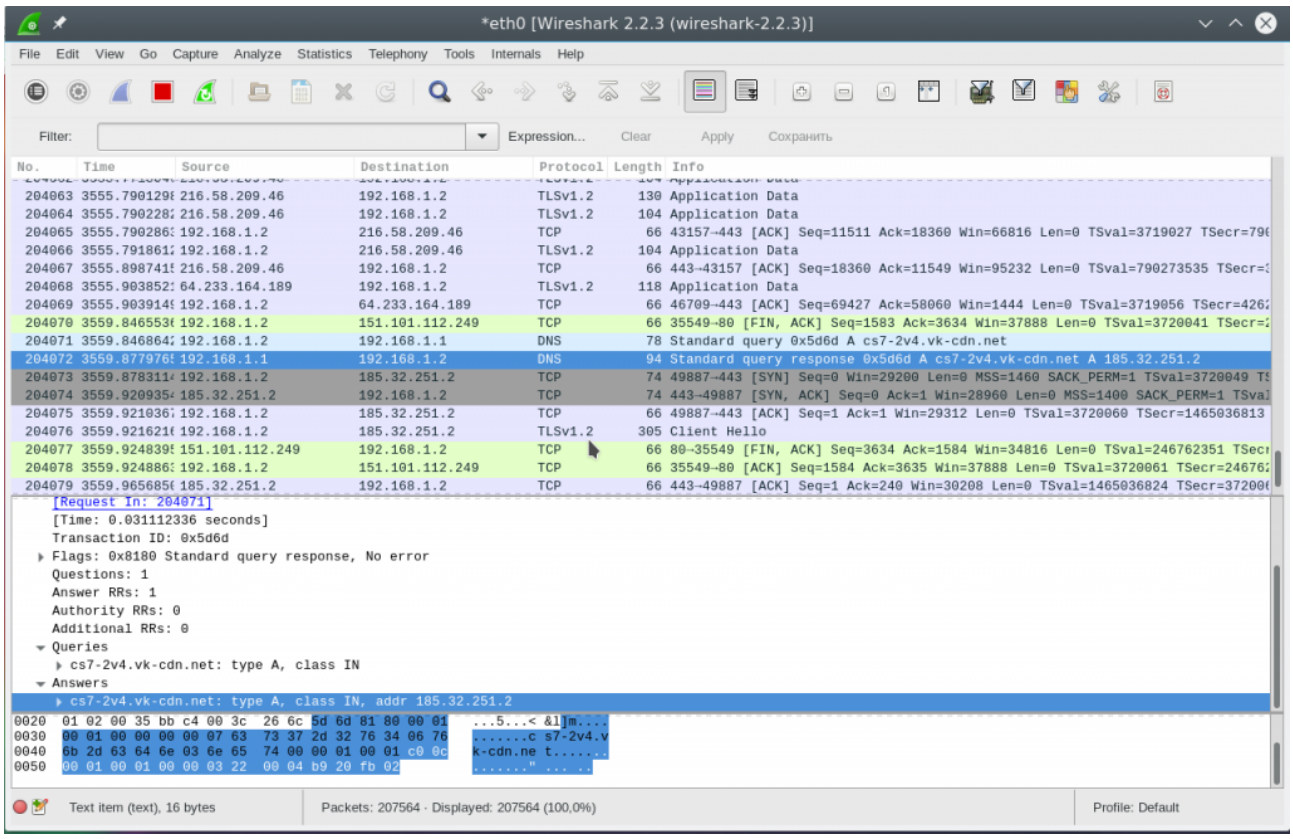
Для начала анализа выберите сетевой интерфейс, например eth0, и нажмите кнопку **Start.**



После этого откроется следующее окно, уже с потоком пакетов, которые проходят через интерфейс. Это окно тоже разделено на несколько частей:

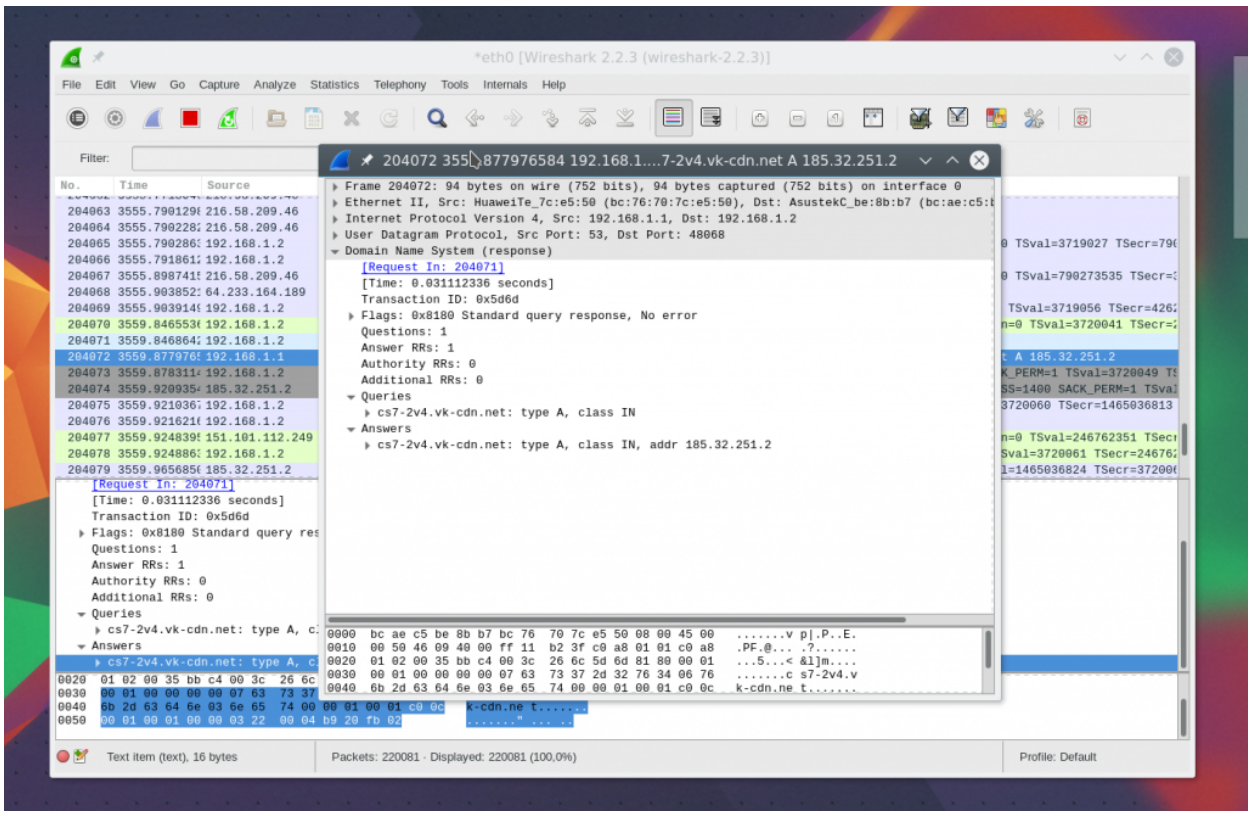
- **Верхняя часть** — это меню и панели с различными кнопками;
- **Список пакетов** — дальше отображается поток сетевых пакетов, которые вы будете анализировать;
- **Содержимое пакета** — чуть ниже расположено содержимое выбранного пакета, оно разбито по категориям в зависимости от транспортного уровня;
- **Реальное представление** — в самом низу отображается содержимое пакета в реальном виде, а также в виде HEX.

Вы можете кликнуть по любому пакету, чтобы проанализировать его содержимое:



Здесь мы видим пакет запроса к DNS, чтобы получить IP-адрес сайта, в самом запросе отправляется домен, а в пакете ответа мы получаем наш вопрос, а также ответ.

Для более удобного просмотра можно открыть пакет в новом окне, выполнив двойной клик по записи:



ФИЛЬТРЫ WIRESHARK

Перебирать пакеты вручную, чтобы найти нужные, очень неудобно, особенно при активном потоке. Поэтому для такой задачи лучше использовать фильтры. Для ввода фильтров под меню есть специальная строка. Вы можете нажать **Expression**, чтобы открыть конструктор фильтров, но там их очень много, поэтому мы рассмотрим самые основные:

- **ip.dst** — целевой IP-адрес;
- **ip.src** — IP-адрес отправителя;
- **ip.addr** — IP отправителя или получателя;
- **ip.proto** — протокол;
- **tcp.dstport** — порт назначения;
- **tcp.srcport** — порт отправителя;
- **ip.ttl** — фильтр по ttl, определяет сетевое расстояние;
- **http.request_uri** — запрашиваемый адрес сайта.

Для указания отношения между полем и значением в фильтре можно использовать такие операторы:

- **==** — равно;
- **!=** — не равно;
- **<** — меньше;
- **>** — больше;

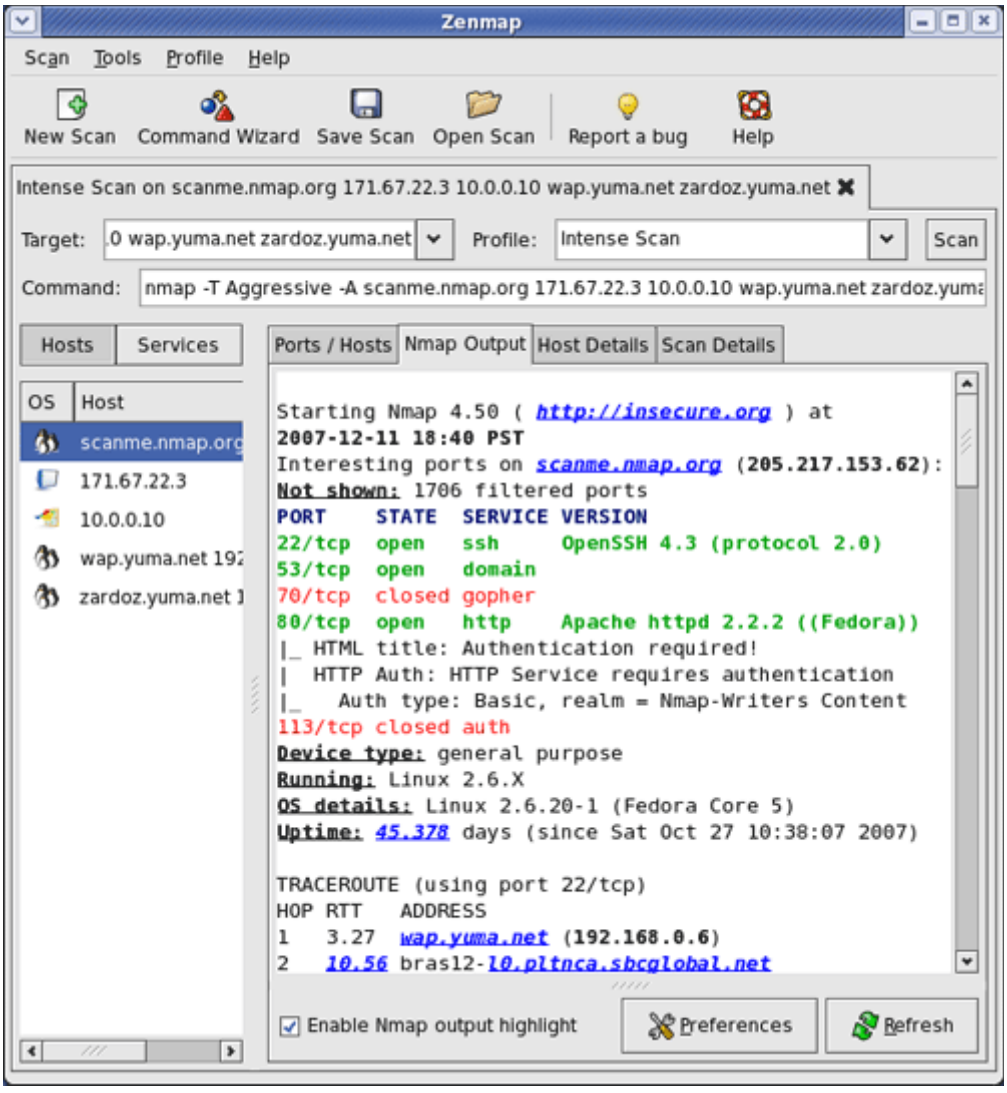
- **<=** — меньше или равно;
- **>=** — больше или равно;
- **matches** — регулярное выражение;
- **contains** — содержит.

Для объединения нескольких выражений можно применять:

- **&&** — оба выражения должны быть верными для пакета;
- **||** — может быть верным одно из выражений.

NMAP ДЛЯ СКАНИРОВАНИЯ ПОРТОВ

Одним из известных и достаточно популярных сканеров безопасности является программа Nmap (Network Mapper). Это абсолютно бесплатный продукт, распространяемый в соответствии с лицензией GNU. Главным его отличием являются очень широкие функциональные возможности. Nmap может просканировать как отдельный хост, так и целую сеть, найти открытые порты, попытаться установить тип операционной системы, запущенные службы и т.п. Однако есть у этой программы и еще одна особенность. Дело в том, что работа с ней осуществляется в командной строке. Что, конечно же, не очень удобно. Исправить эту ситуацию поможет утилита Zenmap. Она выпущена непосредственно разработчиками Nmap и является официальной графической оболочкой этого сканера.



Благодаря Zenmap работать с Nmap становится очень просто. Сначала создается задание, в котором задается объект и все опции сканирования.

Для этого используется специальное окно, состоящее аж семи вкладок. Впрочем, в этом нет абсолютно ничего удивительного, если вспомнить о богатейших возможностях Nmap. Созданный таким образом профиль можно сохранить в виде файла специального формата и в будущем быстро загружать его буквально несколькими кликами мышки. Результаты выполнения задания также отображаются в окне Zenmap. Для удобства просмотра они разделены на несколько вкладок. Эти результаты также можно сохранять в виде файлов, а в будущем открывать для просмотра.

Большинство типов сканирования доступны только привилегированным пользователям, потому что посылаются и принимаются сырые пакеты, что требует прав пользователя root на Unix системах. Под Windows рекомендуется работать с учетной записью администратора, хотя иногда Nmap работает и с непривилегированными пользователя, когда в ОС уже загружена утилита WinPcap. Требование root привилегий было серьезным ограничением, когда Nmap была выпущена в свет в 1997, т.к. многие пользователи имели доступ только к разделяемым аккаунтам. Сейчас мир изменился. Компьютеры стали дешевле, многие пользователи имеют постоянный доступ в Интернет, а Unix системы для домашних компьютеров (включая Linux и Mac OS X) теперь широко распространены. Также теперь доступна Windows версия Nmap, что позволяет запускать ее на еще большем количестве компьютеров. По этим причинам, пользователям нет необходимости запускать Nmap с разделяемых аккаунтов. Это большая удача, т.к. функции требующие привилегированного доступа делают Nmap намного более мощной и гибкой.

Когда Nmap предпринимает попытку выдать правильные результаты, надо иметь ввиду, что вся информация базируется на пакетах, возвращенных целевыми машинами (или брандмауэром перед ними). Такие хосты могут быть ненадежными и посылать ответы с целью ввести Nmap в забдуждение. Намного более распространным случаем являются не совместимые с RFC хосты, которые отвечают на запросы Nmap не так, как должны. Сканирования типа FIN, NULL и Xmas наиболее восприимчивы к такого рода проблемам. Такие сложности специфичны только для определенных типов сканирования, и поэтому обсуждаются в посвященных им разделах.

В этом разделе описываются около дюжины способов сканирования портов поддерживаемых Nmap. В любой момент времени вы можете использовать только один метод; исключение составляет UDP сканирование (-sU), которое

может быть скомбинировано с любым типом TCP сканирования. В качестве памятки имейте ввиду, что различные опции сканирования портов задаются в форме -s<C>, где <C> это символ из названия какого-либо типа сканирования, обычно первый. Единственное исключение это FTP bounce сканирование (-b). По умолчанию Nmap осуществляет SYN сканирование; этот тип сканирования заменяет сканирование с использованием соединения для пользователей не имеющих достаточных привилегий для отправки сырых пакетов (требует root доступа в Unix), или если были заданы цели в формате IPv6. Среди описанных ниже типов сканирования, непривилегированные пользователи могут осуществлять только сканирование с использованием соединения и FTP bounce сканирование.

-sS (TCP SYN сканирование)

> SYN это используемый по умолчанию и наиболее популярный тип сканирования. На то есть несколько причин. Он может быть быстро запущен, он способен сканировать тысячи портов в секунду при быстром соединении, его работе не препятствуют ограничивающие бранмауэры. Этот тип сканирования относительно ненавящив и незаметен, т.к. при таком сканировании TCP соединение никогда не устанавливается до конца. Он работает с любым TCP стеком, не завися от каки-либо особенностей специфичной платформы, как это происходит при сканированиях типа FIN/NULL/Xmas, Maimon и idle сканировании. Он также предоставляет ясную и достоверную дифференциацию между состояниями открыт, закрыт и фильтруется.
>
> Эту технику часто называют сканированием с использованием полуотрытых соединений, т.к. вы не открываете полного TCP соединения. Вы посылаете SYN пакет, как если бы вы хотели установить реальное соединение и ждете. Ответы SYN/ACK указывают на то, что порт прослушивается (открыт), а RST (сброс) на то, что не прослушивается. Если после нескольких запросов не приходит никакого ответа, то порт помечается как фильтруемый. Порт также помечается как фильтруемый, если в ответ приходит ICMP сообщение об ошибке недостижимости (тип 3, код 1,2, 3, 9, 10 или 13).

-sT (TCP сканирование с использованием системного вызова connect)

> Это используемый по умолчанию тип TCP сканирования, когда недоступно SYN сканирование. Это происходит в случае, когда у пользователя нет привилегий для использования сырых пакетов или при сканировании IPv6 сетей. Вместо того, чтобы использовать сырые пакеты, как это происходит при большинстве других типов сканирования, Nmap "просит" операционную систему установить соединение с целевой машиной по

указанному порту путем системного вызова connect. Это такой же высокоуровневый системный вызов, используемый браузерами, P2P клиентами и другими приложениями для установки соединения. Этот вызов является частью программируемого интерфейса, известного как Berkeley Sockets API. Вместо того, чтобы считывать ответы в форме сырых пакетов, Nmap использует этот API для получения информации о статусе каждой попытки соединения.

При доступности SYN сканирования, оно, безусловно, будет являться лучшм выбором. У Nmap имеется меньше возможностей контролирования высокоуровнего вызова connect по сравнению с сырыми пакетами, что делает его менее эффективным. Системный вызов завершает соединения по открытым портам, вместо того, чтобы использовать полуоткрытые соединения, как в случае с SYN сканированием. Таким образом на получение той же самой информации потребуется больше времени и пакетов, да к тому же целевые машины скорее всего запишут это соединение в свои логи. То же самое сделает и порядочная IDS, хотя большинство машин не имеют такой системы защиты. Многие службы на вашей Unix системе будут добавлять запись в системный лог (syslog), а также сообщение об ошибке, когда Nmap будет устанавливать и закрывать соединение без отправления данных. Некоторые службы даже аварийно завершают свою работу, когда это происходит, хотя это не является обычной ситуацией. Администратор, который увидит в логах группу записей о попытке установки соединения от одной и той же системы, должен знать, что его машина подверглась такому типу сканирования.

-sU (Различные типы UDP сканирования)

В то время как большинство сервисов Интернета используют TCP протокол, UDP службы также широко распространены. Тремя наиболее популярными являются DNS, SNMP и DHCP (используют порты 53, 161/162 и 67/68). Т.к. UDP сканирование в общем случае медленнее и сложнее TCP, то многие специалисты по безопасности игнорируют эти порты. Это является ошибкой, т.к. существуют UDP службы, которые используются атакующими. К счастью, Nmap позволяет инвентаризировать UDP порты.

UDP сканирование запускается опцией -sU. Оно может быть скомбинировано с каким-либо типом TCP сканирования, например SYN сканирование (-sS), чтобы использовать оба протокола за один проход.

UDP сканирование работает путем посылки пустого (без данных) UDP заголовка на каждый целевой порт. Если в ответ приходит ICMP ошибка о недостижимости порта (тип 3, код 3), значит порт закрыт. Другие ICMP ошибки недостижимости (тип 3, коды 1, 2, 9, 10 или 13) указывают на то,

что порт фильтруется. Иногда, служба будет отвечать UDP пакетом, указывая на то, что порт открыт. Если после нескольких попыток не было получено никакого ответа, то порт классифицируется как открыт| фильтруется. Это означает, что порт может быть открыт, или, возможно, пакетный фильтр блокирует его. Функция определения версии (-sV) может быть полезна для дифференциации действительно открытых портов и фильтруемых.

Большой проблемой при UDP сканировании является его медленная скорость работы. Открытые и фильтруемые порты редко посылают какие-либо ответы, заставляя Nmap отправлять повторные запросы, на случай если пакеты были утеряны. Закрытые порты часто оказываются еще большей проблемой. Обычно они в ответ возвращают ICMP ошибку о недостижимости порта. Но в отличии от RST пакетов отсылаемых закрытыми TCP портами в ответ на SYN или сканирование с установкой соединения, многие хосты ограничивают лимит ICMP сообщений о недостижимости порта по умолчанию. Linux и Solaris особенно строги в этом плане. Например, ядро Linux 2.4.20 огранивает количество таких сообщений до одного в секунду (в net/ipv4/icmp.c).

Nmap обнаруживает такого рода ограничения и соответственно сокращает количество запросов, чтобы не забивать сеть бесполезными пакетами, которые все равно будут отброшены целевой машиной. К сожалению, при ограничении в стиле Linux (один пакет в секунду) сканирование 65,536 портов займет более 18 часов. К способам увеличения скорости UDP сканирования относятся: параллельное сканирование нескольких хостов, сканирование в первую очередь только наиболее популярных портов, сканирование из-за брандмауэра и использование --host-timeout дял пропуска медленных хостов.

-sN; -sF; -sX (TCP NULL, FIN и Xmas сканирования)

Эти три типа сканирования используют (другие типы сканирования доступны с использованием опции --scanflags описанной в другой секции) незаметную лазейку в TCP RFC, чтобы разделять порты на открытые и закрытые. На странице 65 RFC 793 говорится, что «если порт назначения ЗАКРЫТ .... входящий сегмент не содержащий RST повлечет за собой отправку RST в ответ.» На следующей странице, где обсуждается отправка пакетов без установленных битов SYN, RST или ACK, утверждается что: «вы вряд ли с этим столкнетесь, но если столкнетесь, то сбросьте сегменти и вернитесь к исходному состоянию.»

Когда сканируется система отвечающая требованиям RFC, любой пакет, не содержащий установленного бита SYN, RST или ACK, повлечет за собой отправку RST в ответ в случае, если порт закрыт, или не повлечет никакого

ответа, если порт открыт. Т.к. ни один из этих битов не установлен, то любая комбинация трех оставшихся (FIN, PSH и URG) будет являться правильной. Nmap использует это в трех типах сканирования:

Null сканирование (-sN)

Не устанавливаются никакие биты (Флагов в TCP заголовоке 0)

FIN сканирование (-sF)

Устанавливается только TCP FIN бит.

Xmas сканирование (-sX)

Устанавливаются FIN, PSH и URG флаги.

Эти три типа сканирования работают по одной схеме, различия только в TCP флагах установленных в пакетах запросов. Если в ответ приходит RST пакет, то порт считается закрытым, отсутствие ответа означает, что порт открыт|фильтруется. Порт помечается как фильтруется, если в ответ приходит ICMP ошибка о недостижимости (тип 3, код 1, 2, 3, 9, 10 или 13).

Ключевой особенностью этих типов сканирования является их способность незаметно обойти некоторые не учитывающие состояние (non-stateful) брандмауэры и роутеры с функцией пакетной фильтрации. Еще одним преимуществом является то, что они даже чуть более незаметны, чем SYN сканирование. Все же не надо на это полагаться - большинство современных IDS могут быть сконфигурированы на их обнаружение. Большим недостатком является то, что не все системы следуют RFC 793 дословно. Некоторые системы посылают RST ответы на запросы не зависимо от того, открыт порт или закрыт. Это приводит к тому, что все порты помечаются как закрытые. Основными системами ведущими себя подобным образом являются Microsoft Windows, многие устройства Cisco, BSDI и IBM OS/400. Хотя такое сканирование применимо к большинству систем, основанных на Unix. Еще одним недостатком этих видов сканирования является их неспособность разделять порты на открытые и фильтруемые, т.к. порт помечается как открыт|фильтруется.

-sA (TCP ACK сканирование)

Этот тип сканирования сильно отличается от всех других тем, что он не способен определить открый порт open (или даже открытый|фильтруемый). Он используются для выявления правил брандмауэров, определения учитывают ли он состояние или нет, а также для определения фильтруемых ими портов.

Пакет запроса при таком типе сканирования содержит установленным только ACK флаг (если не используется --scanflags). При сканировании нефильтруемых систем, открытые и закрытые порты оба будут возвращать в ответ RST пакет. Nmap помечает их как не фильтруемые, имея ввиду, что они достижимы для ACK пакетов, но неизвестно открыты они или закрыты. Порты, которые не отвечают или посылают в ответ ICMP сообщение об ошибке (тип 3, код 1, 2, 3, 9, 10 или 13), помечаются как фильтруемые.

-sW (TCP Window сканирование)

Этот тип сканирования практически то же самое, что и ACK сканирование, за исключением того, что он использует особенности реализации различных систем для разделения портов на открытые и закрытые, вместо того, чтобы всегда при получении RST пакета выводить не фильтруется. Это осуществляется путем анализа TCP Window поля полученного в ответ RST пакета. В некоторых системах открытые порты используют положительное значение этого поля (даже в RST пакетах), а закрытые - нулевое. Поэтому вместо того, что все время при получении RST пакета в ответ помечать порты как не фильтруемые, при Window сканировании порты помечаются как открытые или закрытые, если значение поля TCP Window положительно или равно нулю соответственно.

Этот тип сканирования основывается на особенностях реализации меньшинства систем в Интернете, поэтому вы не можете все время доверять ему. В общем случае в системах, не имеющих таких особенностей, все порты будут помечаться как закрытые. Конечно, это возможно, что у машины действительно нет открытых портов. Если большинство просканированных портов закрыты, и лишь несколько распространненых портов (таких как 22, 25, 53) фильтруются, то скорее всего результатам сканирования можно доверять. Иногда, системы будут вести себя прямо противоположным образом. Если в результате сканирования будет найдено 1000 открытых портов и 3 закрытых или фильтруемых, то как раз эти 3 могут оказаться действительно открытыми.

-sM (TCP сканирование Мэймона (Maimon))

Этот тип сканирования носит имя своего первооткрывателя, Уриела Мэймона (Uriel Maimon). Он описал эту технику в журнале Phrack Magazine, выпуск #49 (Ноябрь 1996). Версия Nmap с поддержкой этого типа сканирования была выпущена через два номера. Техника практически такая же как и при NULL, FIN и Xmas сканированиях, только в качестве запросов используются запросы FIN/ACK. Согласно RFC 793 (TCP), в ответ на такой запрос должен быть сгенерирован RST пакет, если порт открыт или закрыт. Тем не менее, Уриел

заметил, что многие BSD системы просто отбрасывают пакет, если порт открыт.

--scanflags (Заказное TCP сканирование)

Действительно продвинутым пользователям Nmap не нужды ограничивать себя заранее приготовленными типами сканирования. С помощью опции --scanflags вы можете разработать свой тип сканирования путем задания специфичных TCP флагов. Используйте свое воображение, обходя системы обнаружения вторжений, чьи производители просто просмотрели справочное руководство Nmap, путем задания собственных правил!

Аргументом опции --scanflags может быть числовое значение, например, 9 (PSH и FIN флаги), но использование символьных имен намного проще. Используйте любые комбинации URG, ACK, PSH, RST, SYN и FIN. Например, опцией --scanflags URGACKPSHRSTSYNFIN будут установлены все флаги, хотя это и не очень полезно для сканирования. Порядок задания флагов не имеет значения.

В добавлении к заданию желаемых флагов, вы также можете задать тип TCP сканирования (например, -sA или -sF). Это укажет Nmap на то, как необходимо интерпретировать ответы. Например, при SYN сканировании отсутствие ответа указывает на фильтруемый порт, тогда как при FIN сканировании - на открытый|фильтруемый. Nmap будет осуществлять заданный тип сканирования, но используя указанные вами TCP флаги вместо стандартных. Если вы не указываете тип сканирования, то по умолчанию будет использоваться SYN.

-sI <зомби_хост>[:<порт>] ("ленивое" idle сканирование)

Этот продвинутый метод сканирования позволяет осуществить действительно незаметное TCP сканирование портов цели (имеется ввиду, что никакие пакеты не отсылаются на целевую машину с вашего реального IP адерса). Вместо этого, на зомби машине используется предсказуемая последовательность генерации ID IP фрагментов для сбора информации об открытых портах цели. Системы IDS будут считать, что сканирование производится с заданной вами зомби машины (которая должна работать и удовлетворять определенным критериям). Этот тип сканирования слишком сложен для описания в этом справочном руководстве, поэтому я написал и выложил подробное описание на https://nmap.org/book/idlescan.html.

Помимо его незаметности (в силу своей природы), этот тип сканирования также позволяет определять основанные на IP доверительные отношения между машинами. Список открытых портов показывает открытые порты с точки зрения зомби машины. Поэтому вы можете попробовать просканировать цель используя различные зомби машины, которым, вы

считаете, возможно будут доверять (посредством правил роутера/пакетного фильтра).

Вы можете добавить номер порта после двоеточия к зомби хосту, если хотите использовать конкретный порт. По умолчанию будет использоваться порт 80.

Порты также могут быть заданы именами, которым они соответствуют в файле nmap-services. Вы даже можете использовать шаблоны * и ? в именах. Например, чтобы просканировать ftp и все порты начинающиеся с http используйте -p ftp,http*. В таких случаях лучше брать аргументы -p в кавычки.

Диапазоны портов заключаются в квадратные скобки; будут просканированы порты из этого диапазона, встречающиеся в nmap-services. Например, с помощью следующей опции будут просканированы все порты из nmap-services равные или меньше 1024: -p [-1024]. В таких случаях лучше брать аргументы -p в кавычки.

-sO (Сканирование IP протокола)

Сканирование такого типа позволяет определить, какие IP протоколы (TCP, ICMP, IGMP и т.д.) поддерживаются целевыми машинами. Технически такое сканирование не является разновидностью сканирования портов, т.к. при нем циклически перебираются номера IP протоколов вместо номеров TCP или UDP портов. Хотя здесь все же используется опция -p для выбора номеров протоколов для сканирования, результаты выдаются в формате таблицы портов, и даже используется тот же механизм сканирования, что и при различных вариантах сканирования портов. Поэтому он достаточно близок к сканированию портов и описывается здесь.

Помимо полезности непосредственно в своей сфере применения, этот тип сканирования также демонстрирует всю мощь открытого программного обеспечения (open-source software). Хотя основная идея довольна проста, я никогда не думал включить такую функцию в Nmap, и не получал запросов на это. Затем, летом 2000-го, Джерард Риджер (Gerhard Rieger) развил эту идею, написал превосходный патч воплощающий ее и отослал его на nmap-hackers рассылку. Я включил этот патч в Nmap и на следующий день выпустил новую версию. Лишь единицы комерческого программного обеспечения могут похвастаться пользователями, достаточно полными энтузиазма для разработки и предоставления своих улучшений!

Способ работы этого типа сканирования очень похож на реализованный в UDP сканировании. Вместо того, чтобы изменять в UDP пакете поле, содержащее номер порта, отсылаются заголовки IP пакета, и изменяется 8 битное поле IP протокола. Заголовки обычно пустые, не содержащие

никаких данных и даже правильного заголовка для требуемого протокола. Исключениями являются TCP, UDP и ICMP. Включение правильного заголовка для этих протоколов необходимо, т.к. в обратном случае некоторые системы не будут их отсылать, да и у Nmap есть все необходимые функции для их создания. Вместо того, чтобы ожидать в ответ ICMP сообщение о недостижимости порта, этот тип сканирования ожидает ICMP сообщение о недостижимости протокола. Если Nmap получает любой ответ по любому протоколу, то протокол помечается как открытый. ICMP ошибка о неостижимости протокола (тип 3, код 2) помечает протокол как закрытый. Другие ICMP ошибки недостижимости (тип 3, код 1, 3, 9, 10 или 13) помечают протокол как фильтруемый (в то же время они показывают, что протокол ICMP открыт). Если не приходит никакого ответа после нескольких запросов, то протокол помечается как открыт|фильтруется

.

-b <FTP хост> (FTP bounce сканирование)

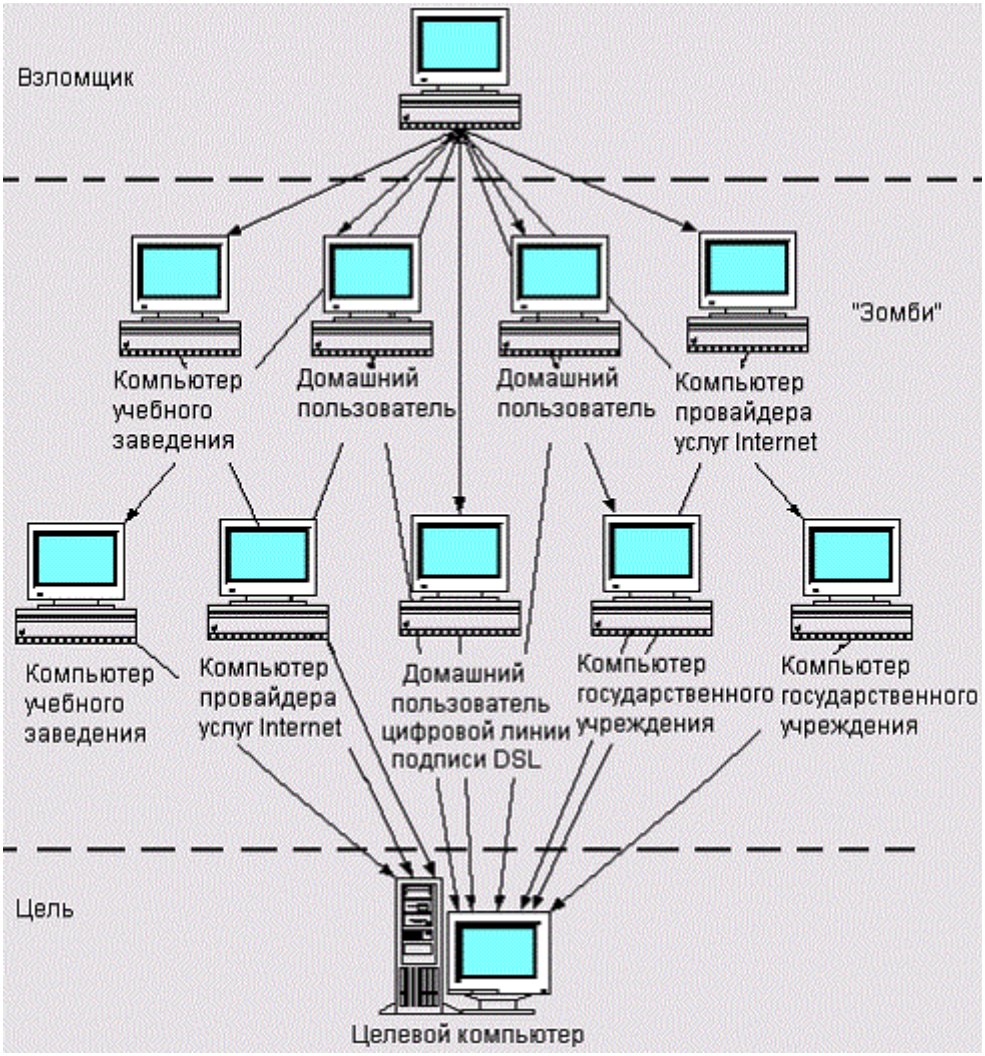
Интересной возможностью FTP протокола (RFC 959) является поддержка так называемых прокси FTP соединений. Это позволяет пользователю подключиться к одному FTP серверу, а затем попросить его передать файлы другому. Это является грубым нарушением, поэтому многие сервера прекратили поддерживать эту функцию. Используя эту функцию, можно осуществить с помощью данного FTP сервера сканирование портов других хостов. Просто попросите FTP сервер переслать файл на каждый интересующий вас порт целевой машины по очереди. Сообщение об ошибке укажет: открыт порт или нет. Это хороший способ обхода брандмауэров, т.к. организационные FTP сервера обычно имеют больше доступа к другим внутренним хостам, чем какие-либо другие машины. В Nmap такой тип сканирования задается опцией -b. В качестве аргумента ей передается <имя_пользователя>:<пароль>@<сервер>:<порт>. <Сервер> - это сетевое имя или IP адрес FTP сервера. Как и в случае в обычными URL, вы можете опустить <имя_пользователя>:<пароль>, тогда будут использованы анонимные данные (пользователь: anonymous пароль:-wwwuser@). Номер порта (и предшествующее ему двоеточие) также можно не указывать; тогда будет использован FTP порт по умолчанию (21) для подключения к <серверу>.

Эта уязвимость была широко распространена в 1997, когда была выпущена Nmap, но теперь почти везде исправлена. Уязвимые сервера по-прежнему есть, так что, если ничего другое не помогает, то стоит попробовать. Если вашей целью является обход бранмауэра, то просканируйте целевую сеть на наличие открытого порта 21 (или даже на наличие любых FTP служб, если вы используете определение версии), а затем попробуйте данный тип сканирования с каждым из найденных. Nmap скажет вам, уязвим хост или

нет. Если вы просто пытаетесь замести следы, то вам нет необходимости (и, фактически, не следует) ограничивать себя только хостами целевой сети. Перед тем как вы начнете сканировать произвольные Интернет адреса на наличие уязвимого FTP сервера, имейте ввиду, что многим системным администраторам это не понравится.

DoS-атака

DoS (аббр. Англ. Denial of Service «отказ в обслуживании») — хакерская атака на вычислительную систему с целью довести её до отказа, то есть создание таких условий, при которых добросовестные пользователи системы не могут получить доступ к предоставляемым системным ресурсам (серверам), либо этот доступ затруднён. Отказ «вражеской» системы может быть и шагом к овладению системой (если в нештатной ситуации ПО выдаёт какую-либо критическую информацию — например, версию, часть программного кода и т. д.). Но чаще это мера экономического давления: потеря простой службы, приносящей доход, счета от провайдера и меры по уходу от атаки ощутимо бьют «цель» по карману. В настоящее время DoS и DDoS-атаки наиболее популярны, так как позволяют довести до отказа практически любую систему, не оставляя юридически значимых улик.



Распределённая DoS-атака

Если атака выполняется одновременно с большого числа компьютеров, говорят о **DDoS-атаке** (от англ. *Distributed Denial of Service*, *распределённая атака типа «отказ в обслуживании»*). Такая атака проводится в том случае, если требуется

вызвать отказ в обслуживании хорошо защищённой крупной компании или правительственной организации.

Первым делом злоумышленник сканирует крупную сеть с помощью специально подготовленных сценариев, которые выявляют потенциально слабые узлы. Выбранные узлы подвергаются нападению, и злоумышленник получает на них права администратора. На захваченные узлы устанавливаются троянские программы, которые работают в фоновом режиме. Теперь эти компьютеры называются компьютерами-зомби, их пользователи даже не подозревают, что являются потенциальными участниками DDoS-атаки. Далее злоумышленник отправляет определенные команды захваченным компьютерам и те, в свою очередь осуществляют мощную DoS-атаку на целевой компьютер.

Существуют также программы для добровольного участия в DDoS-атаках.

В некоторых случаях к фактической DDoS-атаке приводит непреднамеренное действие, например, размещение на популярном интернет-ресурсе ссылки на сайт, размещённый на не очень производительном сервере (слэшдот-эффект). Большой наплыв пользователей приводит к превышению допустимой нагрузки на сервер и, следовательно, отказу в обслуживании части из них.

Защита

Для защиты от сетевых атак применяется ряд фильтров, подключенных к интернет-каналу с большой пропускной способностью. Фильтры действуют таким образом, что последовательно анализируют проходящий трафик, выявляя нестандартную сетевую активность и ошибки. В число анализируемых шаблонов нестандартного трафика входят все известные на сегодняшний день методы атак, в том числе реализуемые и при помощи распределённых бот-сетей.

Классификация DoS-атак

Хакерам гораздо легче осуществить DoS-атаку на систему, чем получить полный доступ к ней. Существуют различные причины, из-за которых может возникнуть DoS-условие, то есть такая ситуация, при которой пользователи не могут получить доступ к ресурсам, которые предоставляет сервер, либо доступ к ним существенно затруднен:

Насыщение полосы пропускания

В настоящее время практически каждый компьютер подключён к сети Internet либо к локальной сети. Это служит отличным поводом для осуществления DoS-атаки за счет переполнения полосы пропускания. Обычно злоумышленники пользуются флудом(англ. *flood* — «наводнение», «переполнение») — атака, связанная с большим количеством обычно бессмысленных или сформированных в неправильном формате запросов к компьютерной системе или сетевому оборудованию, имеющая своей целью или приведшая к отказу в работе системы из-за исчерпания системных ресурсов — процессора, памяти или каналов связи. Есть несколько разновидностей флуда.

HTTP-флуд и ping-флуд

Это самый примитивный вид DoS-атаки. Насыщение полосы пропускания можно осуществить с помощью обычных ping-запросов только в том случае, если канал атакующего (например 1,544 Мбит/с) намного шире канала компьютера-жертвы, скорость в котором 128 Кбит/с. Но такая атака бесполезна против сервера, так как тот, в свою очередь, обладает довольно широкой полосой пропускания. Для атаки на сервер обычно применяется HTTP-флуд. Атакующий шлёт маленький по объёму HTTP-пакет, но такой, чтобы сервер ответил на него пакетом, размер которого в сотни раз больше. Даже если канал сервера в десять раз шире канала атакующего, то все равно есть большой шанс насытить полосу пропускания жертвы. А для того, чтобы ответные HTTP-пакеты не вызвали отказ в обслуживании у злоумышленника, он каждый раз подменяет свой ip-адрес на ip-адреса узлов в сети.

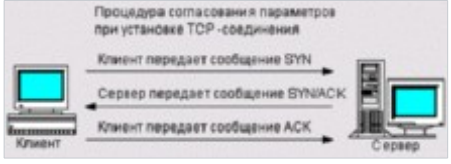
Smurf-атака (ICMP-флуд)

Атака Smurf или ICMP-флуд — один из самых опасных видов DoS-атак, так как у компьютера-жертвы после такой атаки произойдет отказ в обслуживании практически со 100 % гарантией. Злоумышленник использует широковещательную рассылку для проверки работающих узлов в системе, отправляя ping-запрос. Очевидно, атакующий в одиночку не сможет вывести из строя компьютер-жертву, поэтому требуется ещё один участник — это усиливающая сеть. В ней по широковещательному адресу злоумышленник отправляет поддельный ICMP пакет. Затем адрес атакующего меняется на адрес жертвы. Все узлы пришлют ей ответ на ping-запрос. Поэтому ICMP-пакет, отправленный злоумышленником через усиливающую сеть, содержащую 200 узлов, будет усилен в 200 раз. Поэтому для такой атаки обычно выбирается большая сеть, чтобы у компьютера-жертвы не было никаких шансов.

Атака Fraggle (UDP-флуд)

Атака Fraggle (осколочная граната)(от англ. *Fraggle attack*) является полным аналогом Smurf-атаки, где вместо ICMP пакетов используются пакеты UDP, поэтому её ещё называют UDP-флуд. Принцип действия этой атаки простой: на седьмой порт жертвы отправляются echo-команды по широковещательному запросу. Затем подменяется ip-адрес злоумышленника на ip-адрес жертвы, которая вскоре получает множество ответных сообщений. Их количество зависит от числа узлов в сети. Эта атака приводит к насыщению полосы пропускания и полному отказу в обслуживании жертвы. Если все же служба echo отключена, то будут сгенерированы ICMP-сообщения, что также приведёт к насыщению полосы.

Атака с помощью переполнения пакетами SYN (SYN-флуд)



Установка TCP — соединения

До появления атаки Smurf была широко распространена атака с помощью переполнения пакетами SYN, также известная под названием SYN-флуд. Для описания её действия можно остановиться на рассмотрении двух систем A и B, которые хотят установить между собой TCP соединение, после которого они смогут обмениваться между собой данными. На установку соединения выделяется

некоторое количество ресурсов, этим и пользуются DoS-атаки. Отправив несколько ложных запросов, можно израсходовать все ресурсы системы, отведённые на установление соединения. Рассмотрим подробнее, как это происходит. Хакер с системы A отправляет пакет SYN системе B, но предварительно поменяв свой IP-адрес на несуществующий. Затем, ничего не подозревая, компьютер B отправляет ответ SYN/ACK на несуществующий IP-адрес и переходит в состояние SYN-RECEIVED. Так как сообщение SYN/ACK не дойдет до системы A, то компьютер B никогда не получит пакет с флагом ACK.Данное потенциальное соединение будет помещено в очередь. Из очереди оно выйдет только по истечении 75 секунд. Этим пользуются злоумышленники и отправляют сразу несколько пакетов SYN на компьютер жертвы с интервалом в 10 секунд, чтобы полностью исчерпать ресурсы системы. Определить источник нападения очень непросто, так как злоумышленник постоянно меняет исходный IP-адрес.

Схема алгоритма

Рис 1. Схема реализации DoS

Вывод

В результате работы были рассмотрены основные виды сетевых атак. Были изучены популярные утилиты, позволяющие реализовывать атаке на задан ный протокол прикладного уровня. Также было реализовано программное средство, осуществляющее DoS атаку.

Сетевой стэк TCP/IP имеет ряд уязвимостей. Как правило причиной возможности атак на данное семейство протоколов является предположение, что пользователи будут вести себя честно.

Сканирование портов само по себе не является критической уязвимостью, однако может предоставить злоумышленнику информацию об запущенных на удаленном компьютере сервисах. Данная информация может быть в дальнейшем использована для атаки на уязвимые сервисы.

Сетевой анализ является серьезной угрозой конфиденциальности информации. Следует всегда помнить, что сам по себе протокол TCP не шифрует данные и злоумышленник, проанализировав содержимое передаваемых пакетов, всегда может похитить содержимое. Поэтому конфиденциальные данные необходимо шифровать.

DoS атаки являются угрозой доступности информации. Существует несколько видов DoS атак. Защита от угроз данного типа является одной из самых сложных, поскольку для повышения уровня защиты, необходимо закупать новое оборудование. Однако также и сами атаки являются очень затратными.

Тем не менее от многих типов атак существует способ защиты. Работа по защите приложений от атак на TCP/IP обычно возлагается на плечи программиста.

Исходный код

```
#!/usr/bin/env python3

from socket import socket, AF_INET, SOCK_STREAM

host = 'localhost'
port = 27015
addr = (host,port)

cnt = 0



while True:

    tcp_socket = socket(AF_INET, SOCK_STREAM)
    tcp_socket.connect(addr)
    #print('connected')

from socket import socket, AF_INET, SOCK_STREAM
from time import sleep

host = 'localhost'
port = 27015
addr = (host,port)

i = 0
while True:
    i = i + 1
    tcp_socket = socket(AF_INET, SOCK_STREAM)
    tcp_socket.connect(addr)
    print('Send: ' + "Some confidential info #{}".format(i))
    tcp_socket.send("Some confidential info #{}".format(i).encode())
    data = tcp_socket.recv(1024)
    print('Recv: ' + data.decode())
    tcp_socket.close()
    sleep(5)
from socket import socket, AF_INET, SOCK_STREAM
from threading import Thread


def proc(conn):
    while True:
        data = conn.recv(1024)
        if not data:
            break
        print(data.decode())
        conn.send(data)
    conn.close()


host = 'localhost'
port = 27015
addr = (host,port)

tcp_socket = socket(AF_INET, SOCK_STREAM)
tcp_socket.bind(addr)

tcp_socket.listen(100)


while True:
    conn, addr = tcp_socket.accept()
    th = Thread(target=proc, args=(conn,))
    th.start()

tcp_socket.close()
import subprocess
import time

for i in range(100):
    subprocess.Popen(["python3", "attack.py"])


time.sleep(60)
```