электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.7

АТАКИ НА СЕТЕВЫЕ КОММУТАТОРЫ

Спиридонов А.А.*,студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Теоретическая информатика и компьютерные технологии»

Научный руководитель:
Старчак С.Л., д.т.н., доцент, профессор отдела № 1 УВЦ ВИ,
МГТУ им. Н.Э. Баумана

Земляков В.Н.,
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
Начальник цикла отдела № 1 УВЦ ВИ

Ямщиков А.А.,
Заместитель начальника ВК № 2 ФВО ВИ
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана

Научный консультант: Шаменков Н.А.,
к.т.н., начальник лаборатории, НИЦ РКО 4 ЦНИИ МО РФ
bauman@bmstu.ru

Введение

Общеизвестный факт, что в настоящее время активно используются локальные сети.

Локальная вычислительная сеть (ЛВС, локальная сеть; англ. Local Area Network, LAN) — компьютерная сеть, покрывающая обычно относительно небольшую территорию или небольшую группу зданий. Также существуют локальные сети, узлы которых разнесены географически на расстояния более 12 500 км (космические станции и орбитальные центры). Несмотря на такие расстояния, подобные сети всё равно относят к локальным.

Существует множество способов классификации сетей. Основным критерием классификации принято считать способ администрирования. То есть в зависимости от того, как организована сеть и как она управляется, её можно отнести к локальной, распределённой, городской или глобальной сети. Управляет сетью или её сегментом сетевой администратор. В случае сложных сетей их права и обязанности строго распределены, ведётся документация и журналирование действий команды

администраторов. Чаще всего локальные сети построены на технологии Ethernet. Для построения простой локальной сети используются маршрутизаторы и коммутаторы. Даже если грамотно настроено администрирование конкретной подсети – сетевые коммутаторы могут быть слабым звеном в информационной безопасности, т.к. известно и доказано, что сетевые коммутаторы подвержены атакам и имеют уязвимые места.

Необходимо понять физику их работы, выявить способы программного «взлома» коммутаторов, иначе говоря – найти эти слабые места. В рамках поставленной задачи требуется произвести атаку на сетевой неуправляемый коммутатор в сети IPv4. Атаку требуется произвести с UNIX – подобной системы. Для осуществления атаки решено использовать arp – пакеты.

В качестве операционной системы для реализации атаки была выбрана BSD – подобная система Mac OSX. В качестве языка программирования выступил C.

Постановка задачи

Конечная цель рассматриваемой задачи – повышение информационной безопасности в узлах ЛВС за счёт оптимизации ТТХ используемых сетевых коммутаторов с учётом существующих и потенциально возможных способов взлома.

Предварительный анализ задачи позволил определить этапы, которые необходимо реализовать для приближения к конечной цели:

- ✓ Требуется провести предварительный анализ сетевых коммутаторов. Разобраться в логике их работы.
- ✓ Изучить природу взлома сетевых коммутаторов.
- ✓ Разработать методику проведения эксперимента, провести эксперимент.
- ✓ Сделать выводы и наметить дальний план работы.

1. Предварительный анализ работы сетевых коммутаторов

Сетевой коммутатор (switch) – является многопортовой разновидностью моста.

Мосты – это устройства, призванные физически разделить сеть , построенную на репитерах (повторителях) или хабах (Hub или концентратор — это многопортовый репитер), на несколько доменов коллизий.

Коллизия это конфликт передачи, т.е. попытка одновременной передачи данных несколькими устройствами, что невозможно в рамках одного домена (грубо говоря – внутри одного канала, шины), так как все слышат всех и наложение сигналов от

нескольких устройств искажают информацию, поэтому передаваемый пакет данных теряется.

Повторитель копирует (повторяет) полученный пакет на все свои порты, кроме того, откуда пакет был получен. Мост (как и коммутатор) поступает более умно. Он сначала выясняет, для кого предназначен полученный им пакет и передаёт его лишь на тот порт, за которым и находится получатель пакета. Если же получатель пакета находится в том же сегменте (за тем же портом) что и отправитель, то пакет уничтожается (отправитель его и так получил), тем самым остальные сегменты не засоряются паразитным трафиком. Также мосты и коммутаторы могут объединять разнородные сегменты сети.

Важными следствиями из физики работы коммутатора являются:

- Устройства, подключённые к коммутатору могут работать в режиме полного дуплекса (одновременная приём и передача данных), тем самым общая скорость работы между коммутатором и подключённым устройством возрастает в два раза;
- Устройства, подключённые к разным портам, могут работать на разных скоростях (например 100 Мбит и 1 Гбит) и режимах дуплекса;
- На разных портах коммутатора могут быть разные среды передачи, например витая пара и оптика, хотя последнее, конечно, реализуется через отдельные мосты
- Коммутатор является обучающимся устройством и содержит в своей памяти таблицу соответствий MAC адресов хостов (компьютеров) и портов, за которыми эти хосты находятся (Рис. **1**)

MAC address	Порт коммутатора
AA:23:C4:10:22:1F	1
11:22:33:44:55:66	1
99:98:12:43:22:BF	3
19:5F:1D:DD:12:B9	4
13:9B:CC:23:00:78	4

Рис. 1. Таблица соответствий

Логика работы сетевого коммутатора.

Коммутатор хранит в памяти таблицу коммутации (см. рисунок 1), в которой указывается соответствие MAC-адреса узла порту коммутатора. При включении коммутатора эта таблица пуста, и он работает в режиме обучения. В этом режиме

поступающие на какой-либо порт данные передаются на все остальные порты коммутатора. При этом коммутатор анализирует кадры (фреймы) и, определив MAC-адрес хоста-отправителя, заносит его в таблицу коммутации. Впоследствии, если на один из портов коммутатора поступит кадр, предназначенный для хоста, MAC-адрес которого уже есть в таблице, то этот кадр будет передан только через порт, указанный в таблице. Если MAC-адрес хоста-получателя не ассоциирован с каким-либо портом коммутатора, то кадр будет отправлен на все порты, за исключением того порта, с которого он был получен. Со временем коммутатор строит таблицу для всех активных MAC-адресов, в результате трафик локализуется. Стоит отметить малую латентность (задержку) и высокую скорость пересылки на каждом порту интерфейса.

Какие бывают режимы коммутации

Существует три способа коммутации. Каждый из них — это комбинация таких параметров, как время ожидания и надёжность передачи.

1. С промежуточным хранением (Store and Forward). Коммутатор читает всю информацию в кадре, проверяет его на отсутствие ошибок, выбирает порт коммутации и после этого посылает в него кадр.
2. Сквозной (cut-through). Коммутатор считывает в кадре только адрес назначения и после выполняет коммутацию. Этот режим уменьшает задержки при передаче, но в нём нет метода обнаружения ошибок.
3. Бесфрагментный (fragment-free) или гибридный. Этот режим является модификацией сквозного режима. Передача осуществляется после фильтрации фрагментов коллизий (кадры размером 64 байта обрабатываются по технологии store-and-forward, остальные — по технологии cut-through).

Задержка, связанная с «принятием коммутатором решения», добавляется к времени, которое требуется кадру для входа на порт коммутатора и выхода с него, и вместе с ним определяет общую задержку коммутатора.

Симметричная и асимметричная коммутация

Свойство симметрии при коммутации позволяет дать характеристику коммутатора с точки зрения ширины полосы пропускания для каждого его порта. Симметричный коммутатор обеспечивает коммутируемые соединения между портами с одинаковой шириной полосы пропускания, например, когда все порты имеют ширину пропускания 10 Мб/с или 100 Мб/с.

Асимметричный коммутатор обеспечивает коммутируемые соединения между портами с различной шириной полосы пропускания, например, в случаях комбинации портов с шириной полосы пропускания 10 Мб/с и 100 Мб/с или 100 Мб/с и 1000 Мб/с.

Асимметричная коммутация используется в случае наличия больших сетевых потоков типа клиент-сервер, когда многочисленные пользователи обмениваются информацией с сервером одновременно, что требует большей ширины пропускания для того порта коммутатора, к которому подсоединён сервер, с целью предотвращения переполнения на этом порте. Для того чтобы направить поток данных с порта 100 Мб/с на порт 10 Мб/с без опасности переполнения на последнем, асимметричный коммутатор должен иметь буфер памяти.

Асимметричный коммутатор также необходим для обеспечения большей ширины полосы пропускания каналов между коммутаторами, осуществляемых через вертикальные кросс-соединения, или каналов между сегментами магистрали.

Буфер памяти

Для временного хранения пакетов и последующей их отправки по нужному адресу коммутатор может использовать буферизацию. Буферизация может быть также использована в том случае, когда порт пункта назначения занят. Буфером называется область памяти, в которой коммутатор хранит передаваемые данные.

Буфер памяти может использовать два метода хранения и отправки пакетов: буферизация по портам и буферизация с общей памятью. При буферизации по портам пакеты хранятся в очередях (queue), которые связаны с отдельными входными портами. Пакет передаётся на выходной порт только тогда, когда все пакеты, находившиеся впереди него в очереди, были успешно переданы. При этом возможна ситуация, когда один пакет задерживает всю очередь из-за занятости порта его пункта назначения. Эта задержка может происходить даже в том случае, когда остальные пакеты могут быть переданы на открытые порты их пунктов назначения.

При буферизации в общей памяти все пакеты хранятся в общем буфере памяти, который используется всеми портами коммутатора. Количество памяти, отводимой порту, определяется требуемым ему количеством. Такой метод называется динамическим распределением буферной памяти. После этого пакеты, находившиеся в буфере, динамически распределяются по выходным портам. Это позволяет получить пакет на одном порте и отправить его с другого порта, не устанавливая его в очередь.

Коммутатор поддерживает карту портов, в которые требуется отправить пакеты. Очистка этой карты происходит только после того, как пакет успешно отправлен.

Поскольку память буфера является общей, размер пакета ограничивается всем размером буфера, а не долей, предназначенной для конкретного порта. Это означает, что крупные пакеты могут быть переданы с меньшими потерями, что особенно важно при асимметричной коммутации, то есть когда порт с шириной полосы пропускания 100 Мб/с должен отправлять пакеты на порт 10 Мб/с.

Коммутатор работает на канальном (втором) уровне модели OSI. Коммутаторы подразделяются на **управляемые** и **неуправляемые** (наиболее простые). Более сложные коммутаторы позволяют управлять коммутацией на сетевом (третьем) уровне модели OSI. Обычно их именуют соответственно, например «Layer 3 Switch» или сокращённо «L3 Switch». L3 Switch, по своей логике работы очень близок к маршрутизатору (Router).

Далее в данной работе будут рассматриваться только неуправляемые сетевые коммутаторы.

2. Природа взлома сетевого коммутатора

Объект атаки для получения информации:

- Сетевые узлы
- Сетевые службы
- Сетевые протоколы
- Операционные системы
- Средства защиты
- Пользователи сети

Цели атаки:

- Раскрытие информации
- Искажение информации
- Нарушение доступности информации
- Захват сетевого узла или службы
- Расширение прав доступа на узле
- Модификация поведения узла или службы
- Другие цели (дерево целей)

Этапы атаки:

- Разведка
- Реализация
- Скрытие факта атаки

Информация об объекте атаки:

- Сбор публичной информации об объекте Изучение окружения и связей объекта
- Определение топологии сети (TTL, record route)
- Определение доступности, типа и роли сетевого узла (sweep, sniff, os fingerprint, dns)
- Определение доступности и типов служб на узле (port scan, service fingerprint)
- Поиск уязвимостей сетевых узлов и служб (vuln scanners)

Инструменты для атаки

- Социальная психология и социальная инженерия
- Информационные ресурсы об уязвимостях
- Сканеры сетей и уязвимостей (nmap, nessus)
- Анализаторы и генераторы пакетов (Wireshark,..)
- Программы эксплуатации уязвимостей (exploit, shellcode, Metasploit Framework)
- Программы подбора паролей и крипто-анализа, сетевые черви и трояны
- Автономные агенты (botnets)
- Комплекты инструментов (rootkit, BackTrack,SET) Средства разработки ПО для атак

Сложность атаки:

- уровень1 - инструментов и знаний не надо. может произойти случайно
- уровень2 - минимальные знания. универсальные инструменты
- уровень3 - техническая подкованность требуется. инструменты можно найти в Интернет
- уровень4 - инженерная подготовка. редко используемые или специфичные инструменты
- уровень5 - специально разработанные инструменты. академическая подготовка
- уровень6 - моделирование в лаборатории

3. Классификация последствий атак:

По уровням модели TCP/IP

1)**Физический**: повреждение кабеля, электромагнитные помехи

2)**Канальный**: переполнение таблицы коммутации, прослушивание, несанкционированная смена MAC-адреса, подмена записей в ARP-таблице, подмена DHCP-сервера в сети.

3)**Сетевой**: Подмена IP-адресов, фрагментация, инкапсуляция данных в ICMP пакеты, модификация маршрутной таблицы, использование широковещательных адресов для усиления атаки

4)**Транспортный:** подмена порта отправителя, манипуляции или переполнение таблицы состояния TCP соединений, использование нестандартных комбинаций флагов и параметров TCP, перехват TCP соединения

5)**Прикладной:** отравление кеша DNS, подмена идентификатора пользователя, использование ошибок ПО, подбор пароля, инъекции, и.т.д.

По типу уязвимостей:

- проектирования
- реализации
- конфигурации

4. Методика проведения эксперимента

В данной работе требуется провести эксперимент, результатом которого будет иллюстрация наличия проблемы (уязвимость в сетевых коммутаторах).

В рамках проводимого эксперимента требуется: работая на канальном уровне (уровень №2 модели OSI), имея заданную простую топологию сети, зная характеристики сетевого коммутатора произвести атаку arp-flooding на этот сетевой коммутатор и проанализировать исход проведённой атаки. Также требуется разработать и использовать собственный боевой софт для выявления уязвимости. Производить атаку требуется с машины, на которой установлена unix – подобная система.

В данном эксперименте для атак использовалась машина с операционной системой MAC OS X Lion.

Топология сети.

Топология сети в данной работе имеет следующий вид, представленный на Рисунок 2Рис**.** 2**.** Топология сети. В приведённой упрощённой схеме ПК№1, ПК№2 и «Атакующий» находятся в одной подсети и имеют выход во внешнюю сеть через коммутатор.

Атакуемый сетевой коммутатор. Характеристики.

В работе атакуемым устройством являлся EZ Switch 10/100 1023DT (

Рис. 3. EZ Switch 10/100 1023DT), характеристики устройства приведены в Таблица 1 «EZ Switch 10/100 1023DT».

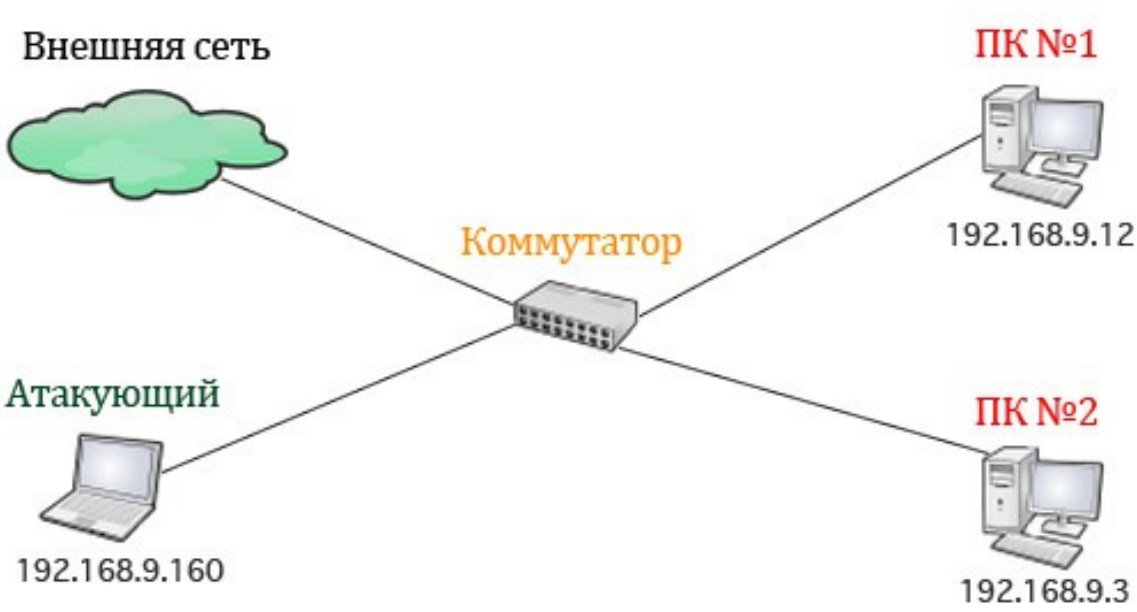

Рис. 2. Топология сети

Рис. 3. EZ Switch 10/100 1023DT

Таблица 1 «EZ Switch 10/100 1023DT»

Число портов коммутатора	24
Размер таблицы MAC адресов	8 192
Внутренняя пропускная способность	4.8 Гбит/сек
Объем оперативной памяти	3 Мб

Данный свитч работает на втором уровне модели OSI и относится к классу неуправляемых сетевых коммутаторов.

Выбранный алгоритм атаки: mac-flooding

В рамках выбранного алгоритма, требуется перевести устройство – коммутатор подсети, в которой находятся машины атакующего, №1 и №2, в широковещательный

режим. Т.е. ARP запросами с машины хакера заполнить таблицу маршрутов у коммутатора, чтобы тот перешёл в широковещательный режим, после чего отправлял все проходящие через него пакеты на все свои порты. Хакеру в этом случае нужно перевести свою сетевую карту в неразборчивый режим и производить анализ всего трафика.

Описание структуры ARP - пакета в ethernet

Что такое сетевой пакет.

В компьютерных сетях пакет — это определённым образом оформленный блок данных, передаваемый по сети. Пакет состоит из двух частей: управляющей информации и данных для передачи (называемых также полезной нагрузкой). Управляющая информация содержит данные, необходимые для доставки полезной нагрузки: адреса отправителя и получателя, коды обнаружения ошибок (например, контрольные суммы) и информацию об очерёдности.

ARP протокол.

ARP (**A**ddress **R**esolution **P**rotocol) — протокол определения адреса, работает на канальном уровне (уровень 2 модели OSI) и предназначен для определения MAC-адреса по известному IP-адресу. Наибольшее распространение этот протокол получил благодаря повсеместности сетей IP, построенных поверх Ethernet, поскольку практически в 100 % случаев при таком сочетании используется ARP.

Существуют следующие типы сообщений ARP: запрос ARP (ARP request) и ответ ARP (ARP reply). Система-отправитель при помощи запроса ARP запрашивает физический адрес системы-получателя. Ответ (физический адрес узла-получателя) приходит в виде ответа ARP.

Перед тем как передать пакет сетевого уровня через сегмент Ethernet, сетевой стек проверяет кэш ARP, чтобы выяснить, не зарегистрирована ли в нём уже нужная информация об узле-получателе. Если такой записи в кэше ARP нет, то выполняется широковещательный запрос ARP. Этот запрос для устройств в сети имеет следующий смысл: «Кто-нибудь знает физический адрес устройства, обладающего следующим IP-адресом?» Когда получатель с этим IP-адресом примет этот пакет, то должен будет ответить: «Да, это мой IP-адрес. Мой физический адрес следующий: …» После этого отправитель обновит свой кэш ARP и будет способен передать информацию получателю. Ниже приведён пример запроса и ответа ARP.

Записи в кэше ARP могут быть статическими и динамическими. Пример, данный выше, описывает динамическую запись кэша. Можно также создавать статические записи в таблице ARP. Это можно сделать при помощи команды:

arp -s <IP-адрес> <MAC-адрес>

Записи в таблице ARP, созданные динамически, остаются в кэше в течение 2-х минут. Если в течение этих двух минут произошла повторная передача данных по этому адресу, то время хранения записи в кэше продлевается ещё на 2 минуты. Эта процедура может повторяться до тех пор, пока запись в кэше просуществует до 10 минут. После этого запись будет удалена из кэша, и будет отправлен повторный запрос ARP.

Структура ARP – пакета в ethernet сети

Ниже представлена структура ARP - пакета в ethernet сети (см. Таблица 2 «структура ARP - пакета в ethernet»). Данная структура включается в себя сразу ethernet (красный) и arp (жёлтый) заголовок. Так удобнее отсылать пакет с точки зрения программиста.

Таблица 2 «структура ARP - пакета в ethernet»

[1][ethernet] **Мак адрес назначения** [6 байт] unsigned char **destination_hardware_address[ETHER_ADDR_LEN];**
[2] [ethernet] **Мак адрес отправителя** [6 байт] unsigned char **source_hardware_address[ETHER_ADDR_LEN];**
[3] [ethernet] **Длина** [2 байта] **или тип** [1 байт] unsigned short **protocol_id;**
[4] [arp] **Тип канального протокола для передачи** [2 байта] unsigned short **hardware_type;**
[5] [arp] **Код сетевого протокола.** [2 байта] unsigned short **protocol_type**;
[6] [arp] **Длина физического адреса** [1 байт] unsigned char **hardware_length**;
[7] [arp] **Длина логического адреса в байтах.** [1 байт] unsigned char **protocol_length**;
[8] [arp] **Код операции отправителя** [2 байта] unsigned short **operation**;
[9] [arp] **Мак адрес назначения** [6 байт] unsigned char **target_hardware_address[ETHER_ADDR_LEN];**
[10] [arp] **Логический адрес назначения** [4 байта] unsigned char **target_ip_address[4];**
[11] [arp] **Мак адрес отправителя** [6 байт]

unsigned char **sender_hardware_address[ETHER_ADDR_LEN];**
[12] [arp] **Логический адрес отправителя** [4 байта] unsigned char **sender_ip_address[4];**
[14] [ethernet] **Контрольная сумма** [4 байта] unsigned char **target_ip_address[4];**

Поле [1] «Мак адрес назначения» состоит из 6 байт и содержит физический адрес устройства в сети, которому адресован данный пакет. Значения этого и следующего поля являются уникальными. Каждому производителю адаптеров Ethernet назначаются первые три байта адреса, а оставшиеся три байта определяются непосредственно самим производителем. Например, для адаптеров фирмы 3Com физические адреса будут начинаться с 0020AF. Первый бит адреса получателя имеет специальное значение. Если он равен 0, то это адрес конкретного устройства (только в этом случае первые три байта служат для идентификации производителя сетевой платы), а если 1 - широковещательный. Обычно в широковещательном адресе все оставшиеся биты тоже устанавливаются равными единице (FF FF FF FF FF FF).

Поле [2] «Мак адрес отправителя» состоит из 6 байт и содержит физический адрес устройства в сети, которое отправило данный кадр. Первый бит адреса отправителя всегда равен нулю.

Поле [3] **«Длина/тип»** может содержать длину или тип кадра в зависимости от используемого кадра Ethernet. Если поле задаёт длину, она указывается в двух байтах. Если тип - то содержимое поля указывает на тип протокола верхнего уровня, которому принадлежит данный кадр.

Поле [4] **«Тип канального протокола для передачи»** содержит номер канального протокола передачи данных. Например, Ethernet имеет номер 0x0001.

Поле [5] **«Код сетевого протокола»** содержит код сетевого протокола. Например, для IPv4 будет записано 0x0800.

Поле [6] **«Длина физического адреса»** содержит длину МАК – адреса. Например, для сетей ethernet, эта длина будет равна 6 байтам.

Поле [7] **«Длина логического адреса в байтах»** содержит длину логического (IP) адреса в байтах.

Поле [8] **«Код операции отправителя»** содержит код операции: 1 в случае запроса и 2 в случае ответа.

Поле [9] **«Мак адрес назначения»** содержит физический адрес получателя.

Поле [10] **«Логический адрес назначения»** содержит IP - адрес получателя.

Поле [11] **«Мак адрес отправителя»** содержит физический адрес отправителя.

Поле [12] **«Логический адрес отправителя»** содержит IP - адрес отправителя.

Поле [13] **«Контрольная сумма»** содержит результат вычисления контрольной суммы всех полей, за исключением преамбулы, признака начала кадра и самой контрольной суммы. Вычисление выполняется отправителем и добавляется в кадр. Аналогичная процедура вычисления выполняется и на устройстве получателя. В случае, если результат вычисления не совпадает со значением данного поля, предполагается, что произошла ошибка при передаче. В этом случае кадр считается испорченным и игнорируется.

Предположения касательно реализации алгоритма

Как уже указывалось в разделе №1 – сетевой коммутатор работает на 2м уровне модели OSI и содержит в своей памяти таблицу соответствий MAC - адресов хостов (компьютеров) и портов, за которыми эти хосты находятся.

Первоначально было предположение, что если заполнить данную таблицу соответствий фальшивыми MAC - адресами («зафлудить» память коммутатора) - коммутатор перестанет контролировать передачу пакетов и будет рассылать их на все свои порты, подобно устаревшим хабам. Благодаря этому возникнет возможность анализировать весь трафик подсети, и задача данного эксперимента будет реализована.

Программная реализация первоначального алгоритма.

Общая логика работы программы.

1) Подключить необходимые заголовочные файлы
2) Определить необходимые для работы программы переменные, константы и структуры.
3) Создать низкоуровневый (сырой) сокет для отправки пакетов.
4) Связать низкоуровневый сокет с сетевой картой, через которую будет происходить отправка пакетов.
5) В цикле заполнять поля заголовка arp – пакета и отправлять в сеть.
6) Как только все пакеты отправлены, завершить работу программы и приступить к анализу работы атакуемой сети, например, используя утилиту wireshark.

Примечание.

Текст в этом разделе, помеченный *курсивом*, означает, что приводятся различия программной реализации тех или иных моментов в разных операционных системах (например, различия в Linux – подобных OS и BSD – подобных).

Необходимые заголовочные файлы и их короткое описание.

В языках программирования Си и C++, заголовочные файлы — основной способ подключить к программе типы данных, структуры, прототипы функций, перечислимые типы, и макросы, используемые в другом модуле.

1) **stdio.h** - заголовочный файл стандартной библиотеки языка Си, содержащий определения макросов, константы и объявления функций и типов, используемых для различных операций стандартного ввода и вывода.
2) **stdlib.h** - заголовочный файл стандартной библиотеки языка Си, который содержит в себе функции, занимающиеся выделением памяти, контроль процесса выполнения программы, преобразования типов и другие.
3) **string.h** - заголовочный файл стандартной библиотеки языка Си, содержащий функции для работы с нуль-терминированными строками и различными функциями работы с памятью.
4) **time.h** - заголовочный файл стандартной библиотеки языка программирования СИ, содержащий типы и функции для работы с датой и временем.
5) **errno.h** - заголовочный файл стандартной библиотеки языка программирования С, содержащий объявление макроса для идентификации ошибок через их код
6) **sys/socket.h** - заголовочный файл содержит базовые функции сокетов BSD и структуры данных.
7) **netinet/in.h** - заголовочный файл содержит семейства адресов/протоколов PF_INET и PF_INET6. Широко используются в сети Интернет, включают в себя IP-адреса, а также номера портов TCP и UDP.
8) **arpa/inet.h** - заголовочный файл содержит функции для работы с числовыми IP-адресами.
9) **netdb.h** - заголовочный файл содержит функции для преобразования протокольных имён и имён хостов в числовые адреса.
10) **netinet/if_ether.h**, **net/ethernet.h** – заголовочные файлы содержат константы (например, размер мак-адреса) и структуры для работы с канальным уровнем (уровнем 2 модели OSI).
11) **net/if.h**, **net/ndrv.h** – заголовочный файлы содержат структуры и константы для удобной работы с сетевыми интерфейсами.

*Заголовочный файл **net/ndrv.h** встречается только в OSX – подобных системах. (В linux - подобных системах, используется, например, заголовочный файл if_packet.h). Все*

остальные заголовочные файлы обычно входят в стандартный дистрибутив UNIX – подобной операционной системы, однако могут располагаться в разных каталогах.

Константы и структуры, необходимые для работы

1) unsigned char DEVICE[] = "en0";

   Переменная содержит имя сетевой карты в системе.

2) struct arp_packet packet;

   Структура для отправки пакета. См. п. 3.3. для подробного описания.

3) struct sockaddr_ndrv s_ndrv;

   Структура для работы с сетевой картой.

4) char random_hardware_address[19];

   Переменная для временного хранения случайного mac – адреса, записанного в обычной нотации (в виде букв и двоеточий).

5) char random_ip_address[16];

   Переменная для временного хранения случайного ip – адреса, записанного в обычной нотации (в виде чисел и точек).

6) int times_repeated;

   Переменная для хранения числа повторений отправки пакетов.

7) struct in_addr tmp_ip_addr_struct;

   Структура для хранения случайного ip - адреса в сетевом виде

8) struct ether_addr *tmp_ether_addr_struct;

   Структура для хранения случайного mac – адреса в сетевом виде

9) struct timespec tw;

   Структура для работы со временем.

Структура sockaddr_ndrv присуща только BSD – подобной операционной системе вроде OSX. В Linux – подобных OS для работы с сетевыми интерфейсами стоит использовать структуры вроде sockaddr_ll

Создание сырого сокета

С точки зрения программиста **сокет** это объект, похожий на файл, с некоторыми дополнительными функциями.

С точки зрения сетевой подсистемы сокет это некоторый объект, модифицирующий точку доступа.

В программе сокет идентифицируется дескриптором - это просто переменная типа int. Программа получает дескриптор от операционной системы при создании сокета, а затем передаёт его сервисам socket API для указания сокета, над которым необходимо выполнить то или иное действие.

С каждым сокетом связываются три атрибута: домен, тип и протокол. Эти атрибуты задаются при создании сокета и остаются неизменными на протяжении всего времени его существования. Для создания сокета используется функция socket();

Создание сырого сокета в программе:

```
int sock_sender;
sock_sender = socket(AF_NDRV, SOCK_RAW, 0);
if (sock_sender < 0) {
printf("socket() error = %d (%s)\n", errno, strerror(errno));
}
```

В Linux – подобных операционных системах тип SOCK_RAW является устаревшим и в настоящий момент не используется. Однако в BSD – подобных OS дело обстоит иначе: SOCK_RAW не является устаревшим, однако данный тип следует использоваться вместе с адресным пространством AF_NDRV

Привязка сырого сокета к сетевой карте

Прежде чем передавать данные через сокет, его необходимо связать с адресом в выбранном домене (эту процедуру называют именованием сокета). Иногда связывание осуществляется неявно (внутри функций connect и accept), но выполнять именование сокета необходимо во всех случаях. Вид адреса зависит от выбранного вами домена.

Функция bind() - связывает локальный адрес и порт (любой из адресов данного устройства) с сокетом.

Связывание сырого сокета с сетевой картой в программе:

```
bind(sock_sender,(struct sockaddr *)&s_ndrv, sizeof(struct sockaddr));
```

В качестве аргументов функция принимает:

1) Дескриптор сокета
2) Параметры сетевого интерфейса в виде структуры
3) Размер структуры с параметрами сетевого интерфейса

В Linux – подобных операционных системах при отправке сырых пакетов функцией sendto() (см. раздел 5.7) можно не применять функцию bind(). В BSD – подобных операционных системах наличие функции bind() до отправки сырого пакета обязательно.

Отправка пакетов через сырой сокет

Для отправки сформированного пакета требуется использовать функцию sendto().

В качестве аргументов функция принимает:

1) Дескриптор сокета
2) Пакет для отправки

3) Размер пакета
4) Флаги (в данной работе флаги не используются)
5) Параметры сетевого интерфейса в виде структуры
6) Размер структуры с параметрами сетевого интерфейса

```
if (sendto(sock_sender, &packet, sizeof(packet), 0 ,
(struct sockaddr *) &s_ndrv, sizeof(struct sockaddr_ndrv)) < 0) {
 perror("sendto() failed");
 exit(-1);
}
```

Анализ сети после реализации первоначального алгоритма.

Для анализа работы сети в данной работе использовалась программа wireshark.

Wireshark (ранее — Ethereal) — программа-анализатор трафика для компьютерных сетей Ethernet и некоторых других. Имеет графический пользовательский интерфейс.

Функциональность, которую предоставляет Wireshark, очень схожа с возможностями программы tcpdump, однако Wireshark имеет графический пользовательский интерфейс и гораздо больше возможностей по сортировке и фильтрации информации. Программа позволяет пользователю просматривать весь проходящий по сети трафик в режиме реального времени, переводя сетевую карту в неразборчивый режим (англ. promiscuous mode).

Программа распространяется под свободной лицензией GNU GPL и использует для формирования графического интерфейса кроссплатформенную библиотеку GTK+. Существуют версии для большинства типов UNIX, в том числе Linux, Solaris, FreeBSD, NetBSD, OpenBSD, Mac OS X, а также для Windows.

Если атака на сетевой коммутатор удалась успешно, согласно первоначальному алгоритму проведения атаки и имеющимся предположениям сетевой коммутатор должен начать работать в режиме хаба.

Так – как размер таблицы мак адресов атакуемого коммутатора заранее известен - было отправлено 8192 пакета с поддельными мак – адресами.

Однако через wireshark удалось увидеть только один «чужой» пакет – далее работа коммутатора нормализовалась.

Переосмысление логики работы коммутатора.

После проведения атаки по первоначальному алгоритму и анализа работы сети после данной атаки стало понятно, что коммутатор, при переполнении таблицы мак – адресов не начинает работать в режиме хаба, а перезаписывает свою таблицу mac адресов начиная с первого.

Была смоделирована работа сети, функционирование которой представлено в виде условной UML диаграммы (Рис. 4**.** Переосмысление работы сетевого коммутатора). По результатам моделирования стало ясно, что отсылку вредоносных пакетов требуется продолжать на протяжении всей атаки. На диаграмме (Рис. 4**.** Переосмысление работы сетевого коммутатора условно изображены 4 объекта рассматриваемой подсети:

1) ПК№2, имеющий условный мак-адрес P2
2) ПК№1, имеющий условный мак-адрес P1
3) switch (атакуемый коммутатор). При моделировании было условлено, что размер таблицы соответствий равен 4 записям. Условная память свитча отображена и на упомянутой диаграмме.
4) Атакующий ПК, имеющий свой мак-адрес F и бесчисленное множество вымышленных мак-адресов вида F*, F**, ….

Диаграмма (Рис. 4**.** Переосмысление работы сетевого коммутатора) иллюстрирует следующую ситуацию.

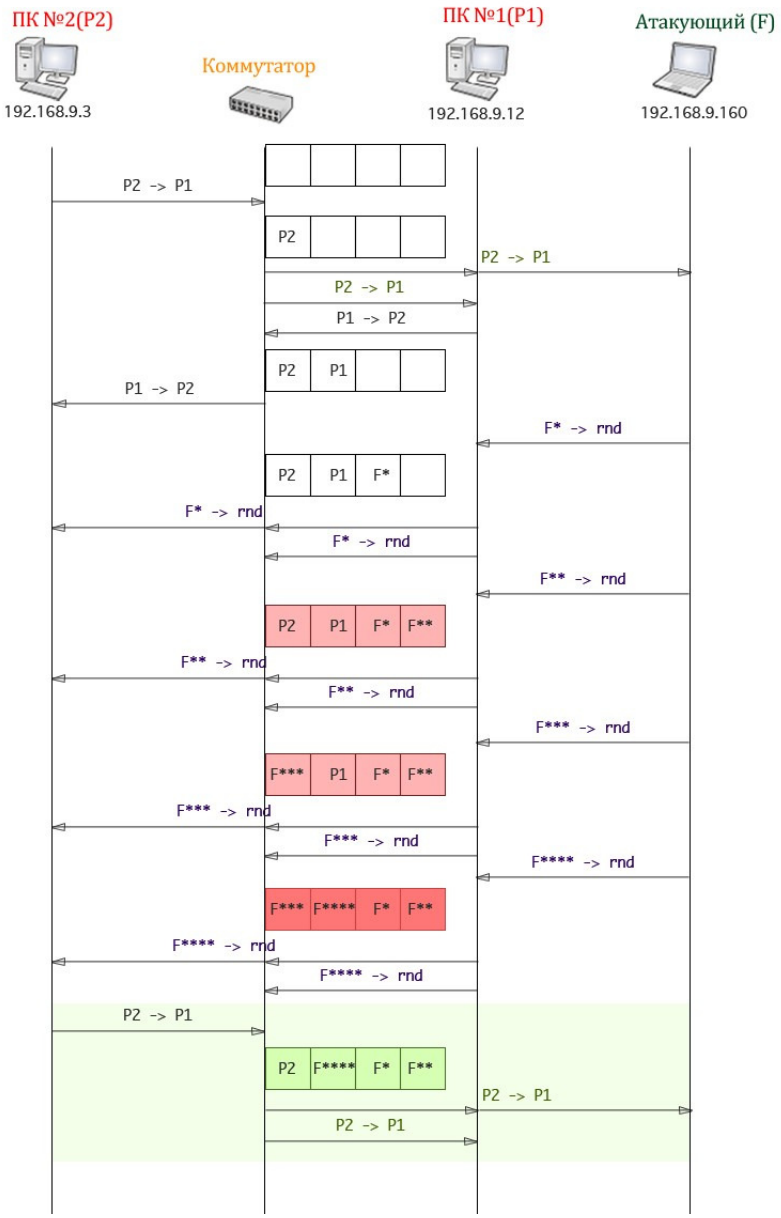

Рис. 4. Переосмысление работы сетевого коммутатора

В начале моделирования рассматриваемая подсеть только начала свою работу: память соответствий у коммутатора пуста. На следующих 2х шагах происходит общение между ПК№2 и ПК№1. После такого общения в таблице соответствий коммутатора появляются мак-адреса P1 (ПК№1) и P2 (ПК№2). Сразу после этого, в рамках данного моделирования, в работу вступает «флудер», задача которого «забить» память коммутатора. Чего он с успехом достигает через некоторое число шагов (заполненная память коммутатора имеет красный фон). После того, как память заполнена – задачей атакующего становится «вытеснить» из таблицы соответствий мак-адреса ПК№1 и ПК№2 (память коммутатора подсвечена ярко-красным цветом). Как только адреса вытеснены – можно считать, что атака удалась – новое сообщение от P2 к P1 будет разослано на все

порты коммутатора и , следовательно, дойдёт до атакующего. Мак-адрес P2 заново попадёт в память свича.

Заключение

Для подтверждения аналитических вводов в отношении возможности успешной атаки был проведён натурный эксперимент. С этой целью был разработан вариант программы, способной в сетях IPv4 нарушить работу коммутатора и получить транзитный трафик для дальнейшего анализа.

В целом, результаты эксперимента подтверждают наличие потенциальной проблемы информационной безопасности в локальных сетях.

В рамках проведенных экспериментальных исследований была известна конфигурация коммутатора, топология сети, мак-адреса машин в сети. Безусловно, априорно известные условия эксперимента в определенной степени обеспечили успешность исследования. Идеализация условий описанного эксперимента, тем не менее, не исключает возможности их достижения в реальной ситуации для атаки на сеть с неизвестными параметрами.

Еще одним из результатов, полученных в ходе эксперимента, явилась невозможность гарантированно захватить весь транзитный трафик. Причиной явилась сложность достижения стационарности процесса постоянно «зафлуженной» память коммутатора средствами разработанного варианта тестовой программы.

Анализ результатов эксперимента указывает на необходимость продолжения исследований в следующих направлениях: разработка более совершенного алгоритма атак на коммутатор; поиск эффективных способов оперативного и скрытого оценивания состава и характеристик локальной сети; совершенствование методики постановки и проведения экспериментальных работ, в том числе, на сетях «случайной» конфигурации; разработка комплекса мер и средств обеспечения информационной безопасности за счет максимального закрытия слабых мест известных типов коммуникаторов.

Список литературы

1. Википедия – свободная энциклопедия [Электронный ресурс] / Wikimedia Foundation. — электронный ресурс. — 2013. — Режим доступа: http://ru.wikipedia.org/, публичный. — Яз. рус. —
2. Gregory McGarry: News and Views [Электронный ресурс] / Gregory McGarry .

3. электронный ресурс. — 2005. — Режим доступа: http://www.gregorymcgarry.com/, публичный. — Яз. en. —
4. Программирование боевого софта под Linux. — СПб.: БХВ-Петербург, 2007. — 416 с.: ил + CD. — ISBN 5-94157-897-9
5. С. В. Запечников, Н. Г. Милославская, А. И. Толстой, Д. В. Ушаков –Информационная безопасность открытых систем. В 2 томах. Том 1. Угрозы, уязвимости, атаки и подходы к защите.
6. Отчет о НИР по теме: «Поисковые исследования методов обеспечения контроля за потоками информации в распределённых АСУ специального назначения» (шифр «Скаут-АСУ») / МГТУ. Руководитель темы С.Л. Старчак. Исполнители: Спиридонов А.А. [и др.]-М., 2012.