Тарасов Иван Вячеславович
РГУ нефти и газа (НИУ) имени И.М. Губкина
Александров Егор Андреевич
РГУ нефти и газа (НИУ) имени И.М. Губкина

СРАВНЕНИЕ ИНСТРУМЕНТОВ СОЗДАНИЯ И РАБОТЫ С СЕТЕВЫМИ ПАКЕТАМИ. ВОПРОСЫ ТЕСТИРОВАНИЯ

Аннотация. Статья посвящена сравнительному анализу инструментов для создания, отправки и анализа сетевых пакетов: Scapy, Hping3, SIPp. В теоретической части рассматриваются назначение, основные возможности и особенности каждого из этих средств, а также приводится сравнительная таблица по ключевым параметрам (поддерживаемые платформы, уровень модели OSI, интерфейс, функциональность работы с пакетами, применение для тестирования и автоматизация). Практическая часть содержит подробные пошаговые инструкции по установке каждого инструмента и примеры базового использования на ОС Альт – с приведением команд и сценариев.

Ключевые слова: генератор пакетов, Scapy, Hping3, SIPp, сетевое тестирование, IDS, firewall, автоматизация.

Введение. В современных сетях для диагностики, настройки и обеспечения безопасности часто возникает необходимость работать с сетевыми пакетами напрямую. Стандартные утилиты (например, ping, traceroute, анализаторы трафика вроде Wireshark) позволяют наблюдать или отправлять лишь ограниченные типы пакетов. Однако при тестировании сложных сценариев – проверке фильтров брандмауэра, реакции системы обнаружения вторжений, задержек (latency) в сети или поведения нестандартных протоколов – требуются специализированные инструменты, называемые генераторами/крафтерами пакетов. Такие программы позволяют конструировать произвольные сетевые пакеты, отправлять их в сеть, а также перехватывать и анализировать ответы.

При работе с сетевым оборудованием необходимо овладеть навыками использования специализированных инструментов для проведения тестирования программных продуктов в сфере информационной безопасности. С помощью таких инструментов можно, например, имитировать атаки для проверки работы IDS, настроить особые флаги TCP/UDP пакетов для выявления правил фильтрации в firewall, измерить время отклика сервера на нестандартные запросы или провести нагрузочное тестирование канала связи. Кроме того, некоторые инструменты предоставляют скриптовое управление, что позволяет автоматизировать тестовые сценарии и интегрировать их в процессы непрерывного тестирования сети.

Данная статья рассматривает три популярных инструмента для работы с сетевыми пакетами – Scapy, Hping3, SIPp. Scapy представляет собой гибкую библиотеку и консольное приложение на языке Python, Hping3 – утилита командной строки для генерации TCP/IP пакетов, SIPp – инструмент для тестирования производительности протокола SIP, использующийся для проверки качества потоков вызовов в VoIP-приложениях. В теоретической части мы сравним эти инструменты по их возможностям и области применения. В практической части приведены инструкции по установке каждого средства и примеры их базового использования на ОС Альт, включая типичные команды и сценарии тестирования.

Теоретическая часть. Приведем обзор и сравнение инструментов.

Scapy

Scapy – это интерактивная оболочка и библиотека на Python для манипулирования сетевыми пакетами. Данный инструмент работает с библиотекой захвата трафика libpcap и может одновременно выступать и как сниффер (перехватчик/анализатор трафика), и как конструктор пакетов.

Scapy позволяет создавать пакеты практически любых протоколов, декодировать и изменять их, отправлять в сеть, а также получать и сопоставлять ответы. Фактически Scapy предоставляет программисту удобный интерфейс для низкоуровневого доступа к сети через Python.

Благодаря поддержке множества протоколов и гибкости Python, Scapy способен выполнять самые разные задачи: формирование пакетов с произвольными полями, отправка их на нужном уровне (канальном или сетевом), прослушивание трафика и фильтрация ответов. С его помощью можно проводить сканирование портов, traceroute, активный и пассивный анализ сетей, тестирование на проникновение, разработку и отладку новых сетевых протоколов. Scapy отличается тем, что за счет скриптового подхода в несколько строк кода можно реализовать сценарии, которые заменяют работу сразу нескольких специализированных утилит (таких как hping, nmap, arpspoof, tcpdump и др.) [3]. Для работы Scapy требует повышенных привилегий (запуск от администратора/суперпользователя), поскольку осуществляет прямой доступ к сетевому интерфейсу.

Как можно применить Scapy для тестирования:

Этот инструмент ценят эксперты по безопасности за возможность писать свои сценарии атак и проверок. Например, Scapy позволяет сымитировать сложные сетевые атаки (DNS Spoofing, ARP Poisoning, TCP-сессию с нестандартными флагами), чтобы проверить реакцию системы защиты. Также Scapy применяется для тестирования брандмауэров – можно отправлять пакеты с разными комбинациями флагов TCP (SYN, ACK, FIN и др.) или с фрагментацией, чтобы выявить, какие проходят через фильтр. С помощью скриптов Scapy можно реализовать собственные сканеры портов или трассировщики маршрута с гибкими настройками (например, менять TTL или размер пакета для выявления MTU). Еще одна область – измерение задержек и производительности: Scapy может отправлять серию пакетов и вычислять статистику RTT (Round-Trip Time). За счет полного контроля над

пакетами Scapy подходит для тестирования нестандартных протоколов и устройств (IoT, промышленные сети), где готовых средств может не быть.

Hping3

Hping3 – консольный инструмент командной строки для генерации и анализа TCP/IP пакетов. По сути, hping задумывался как расширенная версия утилиты ping, способная отправлять не только ICMP-эхо запросы, но и любой произвольный пакет TCP, UDP или RAW-IP [4]. Hping3 позволяет вручную задавать различные поля IP-пакета (IP-адрес отправителя – в том числе с подделкой, TTL, флаги, номера портов и пр.), отправлять серии пакетов с заданной скоростью, а затем выводить полученные ответы (например, пакеты ICMP или TCP RST/ACK в ответ) подобно тому, как ping отображает ответы ICMP. Hping3 поддерживает фрагментацию IP-пакетов, режим traceroute (определение маршрута путем увеличения TTL), сканирование портов (флаг --*scan*), и может работать в различных режимах (TCP, UDP, ICMP, RAW-IP) переключаемых опциями командной строки [5].

Как можно применить Hping3 для тестирования:

Основная сфера применения – аудит безопасности и тестирование брандмауэров. Hping давно стал «де-факто» инструментом для тестирования правил firewall и проведения различных видов порт-сканирования. Например, используя hping3 можно проверить, какие порты на удаленном хосте открыты, отправляя TCP SYN-пакеты на диапазон портов (*hping3 --scan 1-1024 -S <адрес>*). Кроме того, hping используется для имитации DoS-атак и нагрузочного тестирования – режим --*flood* отправляет пакеты максимально быстро, что позволяет генерировать трафик высокой интенсивности. Также hping3 полезен для измерения производительности и задержек: им можно выполнить TCP ping – отправлять TCP SYN на определенный порт и измерять RTT, если целевой порт отвечает SYN+ACK (пример: *hping3 -S -p 80 <адрес>* показывает время ответа похожее на ping, но по TCP-порту 80). Режим traceroute (--*traceroute*) дает возможность трассировать маршрут до хоста,

посылая не стандартные UDP/ICMP, а любые пакеты (что помогает, если стандартный traceroute блокируется).

SIPp

SIPp – это инструмент для тестирования производительности протокола SIP. Он включает несколько базовых сценариев для агентов пользователя SipStone (UAC и UAS) и устанавливает и освобождает несколько вызовов с методами INVITE и BYE. SIPp позволяет тестировать многие реальные устройства SIP, такие как SIP-прокси, B2BUAs, SIP-медиасерверы, SIP/x-шлюзы и SIP-PBX. Также с его помощью можно эмулировать тысячи агентов пользователя, вызывающих SIP-систему.

SIPp генерирует SIP-трафик с настраиваемой интенсивностью, проводить тестирование производительности как серверов, так и клиентов, а также моделировать различные сценарии взаимодействия между участниками коммуникации.

Параметры тестирования можно настраивать через конфигурационные файлы, что позволяет адаптировать инструмент под конкретные задачи. SIPp может читать XML-файлы сценариев, описывающие любую конфигурацию для тестирования производительности. Среди функций программы — динамическое отображение статистики о проводимых тестах (частота вызовов, задержка в пути и статистика сообщений), периодические дампы статистики в формате CSV, TCP и UDP через несколько сокетов или мультиплексирование с управлением ретрансмиссией, регулярные выражения и переменные в файлах сценариев, динамически регулируемая частота вызовов.

Среди возможных сценариев применения можно выделить следующие:

- Тестирование пропускной способности SIP-серверов
- Проверка устойчивости системы к пиковым нагрузкам
- Анализ задержек при различных условиях сети
- Отладка новых функций SIP-приложений

- Оценка производительности при разных сценариях использования

Исходя из всего вышеперечисленного, SIPp охватывает широкий спектр задач: от тестирования пропускной способности SIP-серверов до проверки устойчивости системы к пиковым нагрузкам. Инструмент активно используется при отладке новых функций SIP-приложений и оценке производительности в различных сценариях использования.

Практическая часть. Далее приведены практические инструкции по установке и базовому применению каждого из рассмотренных инструментов. Каждый раздел содержит процесс установки на Linux и пример запуска или сценария использования с пояснениями.

Установка и базовое использование Scapy

Установка на ОС Альт: Его можно установить командой: *sudo apt-get update && sudo apt-get install python3-scapy*. Это установит Scapy для Python3. Альтернативно, можно установить последнюю версию через pip: *pip install scapy* (возможно, потребуется pip3 и права root либо создание виртуального окружения).

Запуск и использование: Scapy можно использовать в двух основных режимах:

- Интерактивная консоль. В терминале набираем *sudo scapy*. Появится приглашение >>>, где можно вводить команды на Python, используя возможности Scapy.
- Скрипт Python. Можно написать .py-скрипт, импортировать модуль Scapy (*from scapy.all import* *) и выполнять необходимые действия, затем запускать скрипт.

  Рассмотрим простой пример интерактивной работы (Python-команды вводятся после >>>):

```
                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version git-archive.dev8b63d73a17
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | Craft packets like I craft my beer.
       scccccp///pSP///p          p//Y   |            -- Jean De Clerck
      sY/////////y  caa           S//P   |
       cayCyayP//Ya                pY/Ya
        sY/PsY////YCc            aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs

>>> from scapy.all import *
>>> a=sniff(count=10)
>>> a.nsummary()
0000 Ether / IP / UDP 10.0.2.15:netbios_dgm > 10.0.2.255:netbios_dgm / NBTDatagram / Raw
0001 Ether / IP / UDP 10.0.2.15:netbios_dgm > 10.0.2.255:netbios_dgm / NBTDatagram / Raw
0002 Ether / IP / UDP / NTP v??, ??
0003 Ether / IP / UDP / NTP v??, ??
0004 Ether / ARP who has 10.0.2.2 says 10.0.2.15
0005 Ether / ARP is at 52:54:00:12:35:02 says 10.0.2.2 / Padding
0006 Ether / IP / UDP / NTP v??, ??
0007 Ether / IP / UDP / NTP v??, ??
0008 Ether / ARP who has 10.0.2.2 says 10.0.2.15
0009 Ether / ARP is at 52:54:00:12:35:02 says 10.0.2.2 / Padding
>>>
```

Рис. 1. Генерация трафика в Scapy

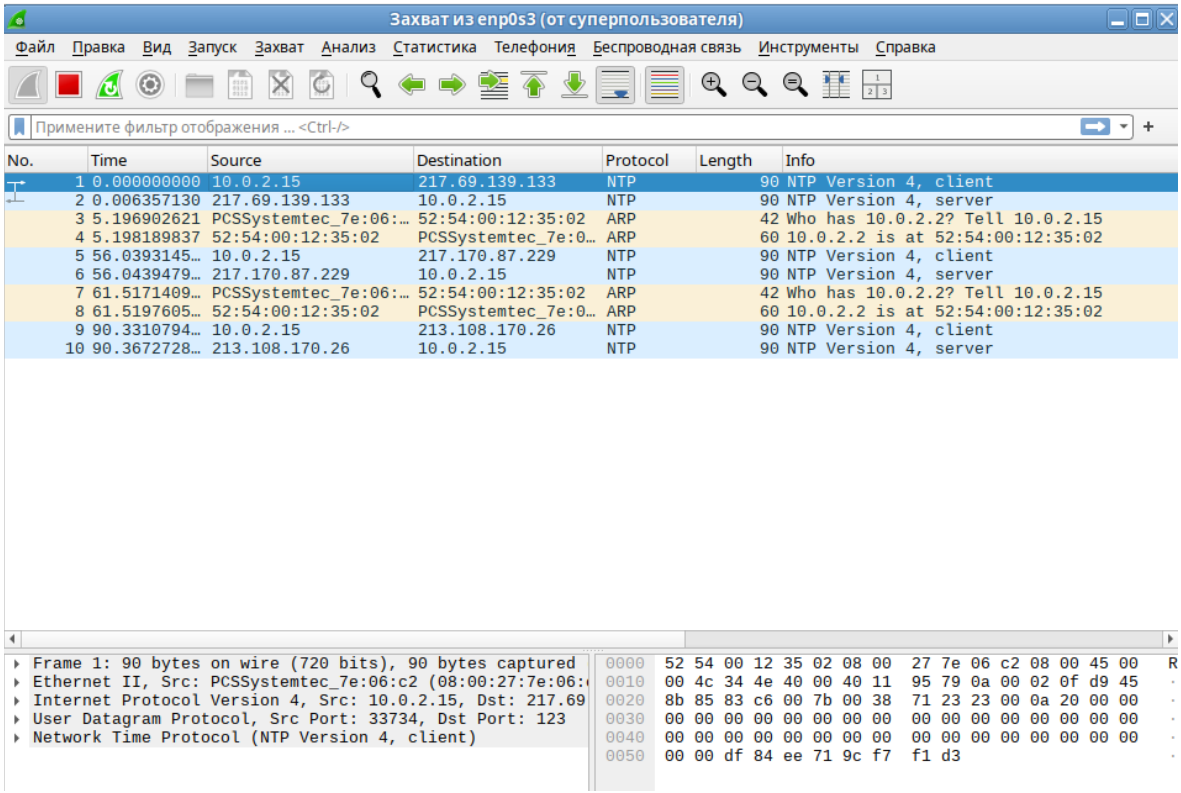

Рис. 2. Перехват трафика Scapy через Wireshark

Число *count* означает количество пакетов, которые мы будем слушать в эфире до окончания программы. В нашем случае это 10, но этот параметр не является обязательным, и если его не указать, то наш сниффер будет работать до получения комбинации ctrl + c. Метод nsummary(), в свою очередь, выводит достаточно подробную статистику о захваченном трафике.

С помощью Wireshark мы видим, какая информация перехватилась:

- Ether — это кадр канального уровня (Ethernet), указывающий, что это Ethernet-фрейм.
- NTP — сетевой протокол для синхронизации внутренних часов компьютера.
- ARP — это протокол, предназначенный для определения MAC-адреса другого компьютера по известному IP-адресу.
- Также представлены IP-адреса источника и назначения.

Установка и базовое использование Hping3

Установка на ОС Альт: В терминале достаточно выполнить *sudo apt-get install hping3*. Это установит бинарник hping3 в систему. Hping3 не требует дополнительных библиотек кроме libpcap и TCL (для скриптов, обычно они предустановлены).

Базовое использование: Hping3 запускается из командной строки: *hping3 [опции] <целевой хост>*. Рассмотрим 2 примера:

1) Traceroute через hping:

```
hping3> hping3 --traceroute -V -S -p 80 example.com
using lo, addr: 127.0.0.1, MTU: 65536
HPING example.com (lo 127.0.0.1): S set, 40 headers + 0 data bytes
len=40 ip=127.0.0.1 ttl=64 DF id=0 tos=0 iplen=40
sport=80 flags=RA seq=0 win=0 rtt=1.4 ms
seq=0 ack=65450897 sum=d7f4 urp=0

len=40 ip=127.0.0.1 ttl=64 DF id=0 tos=0 iplen=40
sport=80 flags=RA seq=1 win=0 rtt=2.0 ms
seq=0 ack=401207805 sum=67a2 urp=0

len=40 ip=127.0.0.1 ttl=64 DF id=0 tos=0 iplen=40
sport=80 flags=RA seq=2 win=0 rtt=1.7 ms
seq=0 ack=446379310 sum=845c urp=0

len=40 ip=127.0.0.1 ttl=64 DF id=0 tos=0 iplen=40
sport=80 flags=RA seq=3 win=0 rtt=0.3 ms
seq=0 ack=308073635 sum=4cc5 urp=0

^C
root@tarasov:~$
--- example.com hping statistic ---
4 packets tramitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.3/1.4/2.0 ms
```

Рис. 3. Генерация трафика в Hping3

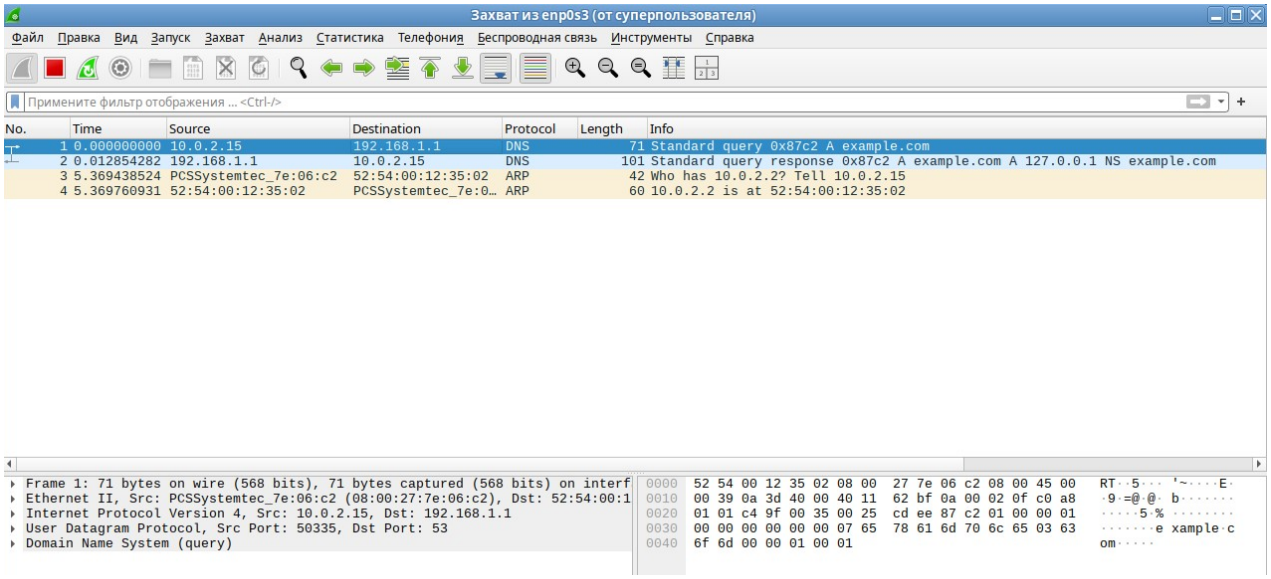

Рис. 4. Перехват трафика Hping3 через Wireshark

Здесь флаг *--traceroute* запускает режим трассировки, *-V – verbose* (подробный вывод). Hping3 начнет увеличивать TTL и покажет промежуточные узлы маршрута до *www.example.com*. В отличие от обычного traceroute, hping3 мог бы делать то же с TCP (-S) или UDP (-2) пакетами, если ICMP блокируется. С помощью Wireshark мы видим, что DNS сервер отправляет запрос на получение IP адреса и получает ответ.

Какая информация сгенерировалась с помощью Hping3:

- Общая длина пакета *len* в байтах.
- IP-адрес источника.
- Time To Live (*ttl*) — ограничение на количество маршрутизаторов, через которые пакет может пройти.
- Don't Fragment флаг (*DF*) — указывает, что фрагментация пакета запрещена.
- Идентификатор IP-пакета *id*, обычно используется для сборки фрагментированных пакетов.
- Тип сервиса (Type of Service, *tos*).
- Общая длина IP-пакета *iplen*.
- Исходный порт *sport* (здесь 80 – стандартный порт для HTTP).
- Нумерация *seq* последовательных номеров TCP-пакета.
- Время round-trip time (*rtt*) — задержка в обратной связи (может быть измерена при пинге или анализе).

2) Flood-тест (нагрузочный):

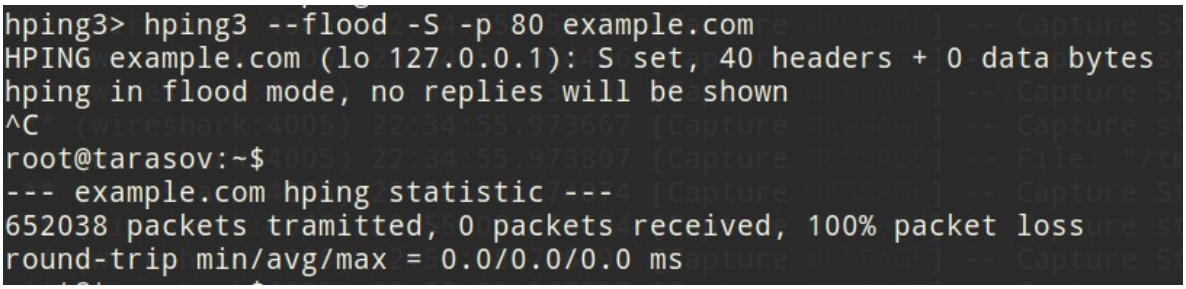

Рис. 5. Генерация трафика в Hping3

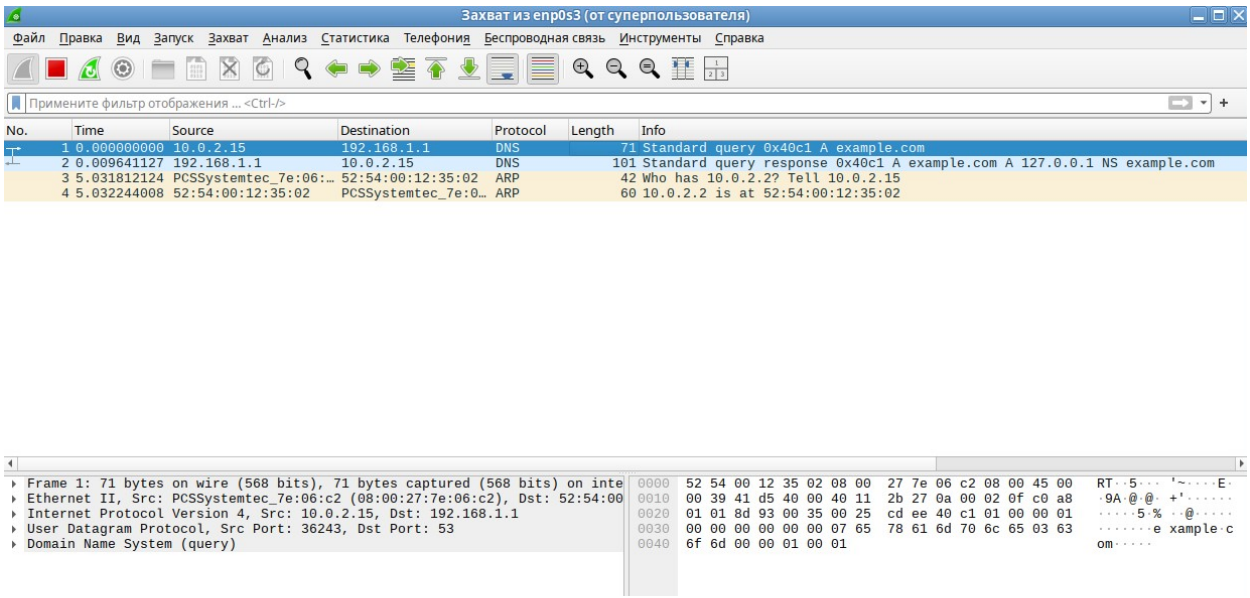

Рис. 6. Перехват трафика Hping3 через Wireshark

Ключ *--flood* заставляет отправлять пакеты максимально быстро без ожидания ответов. Данный пример начнет SYN-flood. Через несколько секунд можно прервать комбинацией Ctrl+C. Hping3 при flood не печатает ответы, т.к. их может быть слишком много, и они игнорируются. Единственная информация, которая становится доступна – это статистика переданных пакетов, а также через Wireshark мы видим, что DNS сервер отправляет запрос на получение IP адреса и получает ответ.

Установка и базовое использование SIPp

Установка на ОС Альт: SIPp устанавливается через команды *sudo apt-get update && sudo apt-get insatall sip*.

Базовое использование: Запускаем приемник с помощью команды *sipp -sn uas -i 127.0.0.1 -p 5060 -trace_msg -trace_err*. Используя встроенный сценарий UAS, привязываем процесс к loopback-адресу 127.0.0.1 и к стандартному SIP-порту 5060, чтобы удобно фильтровать трафик в Wireshark. Ключи `-trace_msg` и `-trace_err` включают логирование SIP-сообщений и возможных ошибок — это помогает быстро понять, что именно произошло на стороне сервера.

```
[root@vbox build]# sipp -sn uas -i 127.0.0.1 -p 5060 -trace_msg -trace_err
```

Рис. 7. Запуск SIPp

Происходит запуск следующего сценария:

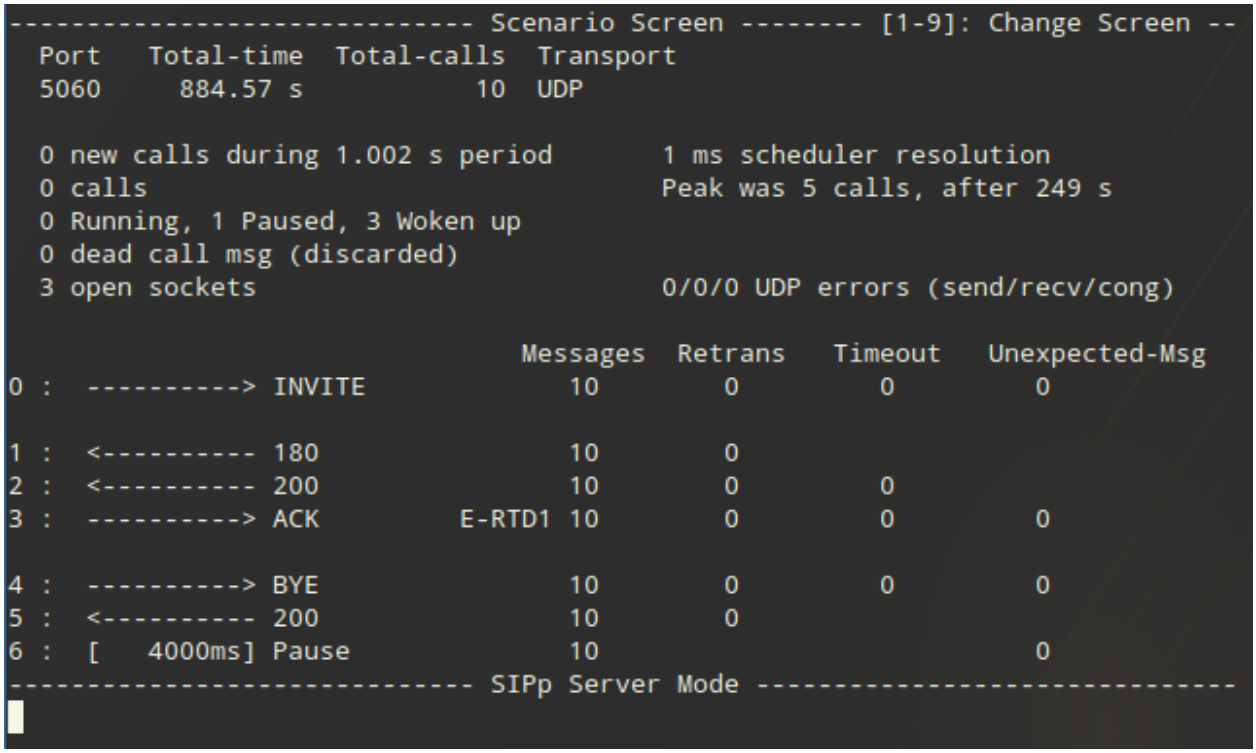

Рис. 8. Запуск SIPp UAS (127.0.0.1:5060)

Запускаем генератор вызовов: *sipp -sn uac 127.0.0.1:5060 -i 127.0.0.1 -p 5070 -s 1001 -m 5 -r 2 -trace_msg*.

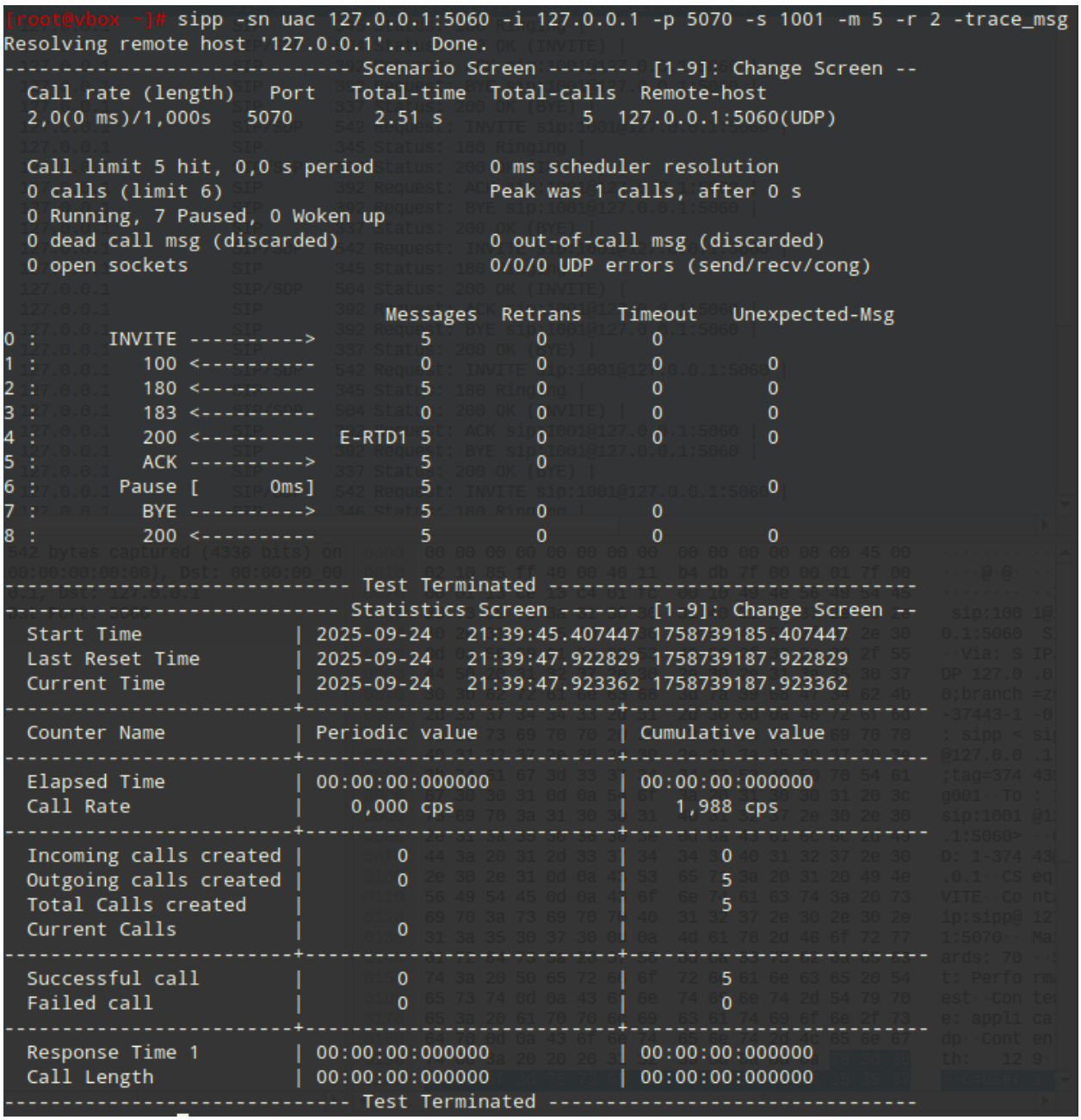

Рис. 9. Успешные вызовы SIPp UAC

UAC звонит на UAS по адресу 127.0.0.1:5060. Параметры выбраны так: отдельный локальный порт клиента `-p 5070`, чтобы не конфликтовать с сервером на 5060; идентификатор вызываемого абонента `-s 1001` — просто метка, её видно в логах и в Wireshark; количество вызовов -m 5 и скорость -r 2 (2 вызова/сек) — достаточно, чтобы показать устойчивую работу без перегруза; -trace_msg сохраняет сообщения клиента для подтверждения результата.

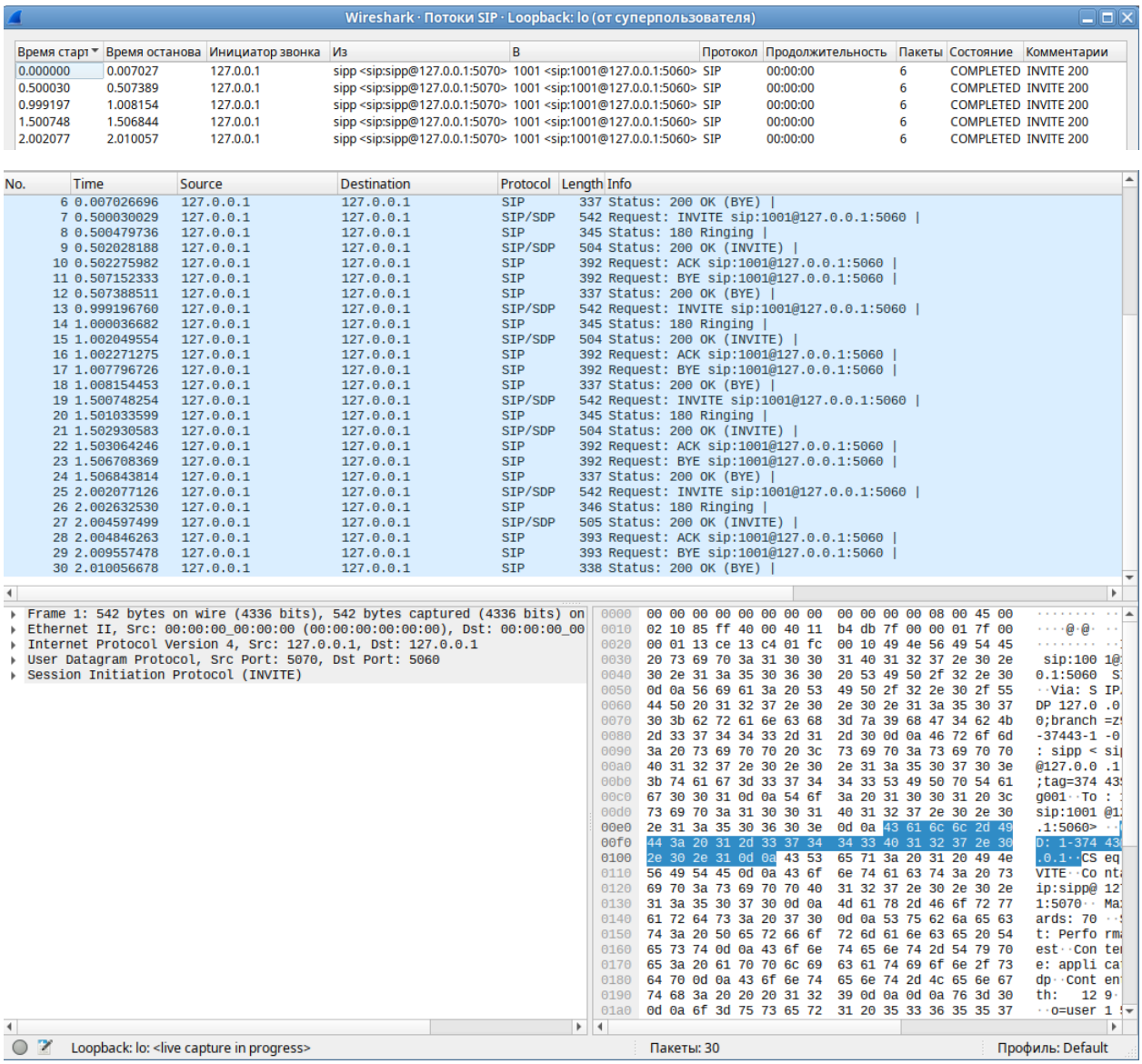

Рис. 10. Wireshark (lo, фильтр sip): последовательность SIP-сообщений для вызовов UAC - UAS

На экране UAC получаем Successful call: 5, Failed: 0.

Итак, мы рассмотрели 3 инструмента для работы с сетевыми пакетами. В таблице 1 ниже приведены ключевые характеристики данных инструментов для наглядного сравнения.

Таблица 1 – Сравнение инструментов Scapy, Hping3, SIPp по основным характеристикам

Параметр	**Scapy**	**Hping3**	**SIPp**
Лицензия	GPLv2	GPLv2	GPL
Версия на официальном сайте	2.6.1	3.a2.ds2-10 (Debian)	3.4
Версия в репозитории Альт	2.6.1-alt2	0.0.20051105-alt8	3.5.1-alt1
Генерация пакетов (типы)	**Канальный уровень**: Ethernet-кадры (Ether), VLAN-теги (Dot1Q). **Сетевой уровень**: IPv4-пакеты, IPv6-пакеты, ICMP-сообщения. **Транспортный уровень**: TCP-сегменты, UDP-датаграммы. **Прикладной уровень**: HTTP-запросы.	**Сетевой уровень:** ICMP-пакеты *(альтернатива стандартному ping, анализ ICMP-ответов)*, RAW-IP пакеты *(создание произвольных IP-пакетов, ручная настройка всех полей заголовка)*. **Транспортный уровень:** TCP-пакеты *(с различными TCP-флагами (SYN, ACK, FIN, RST), поддержкой сканирования портов, возможностью создания SYN-флуда)*, UDP-пакеты *(с*	**Типы SIP-сообщений:** INVITE (для инициирования вызова), ACK (подтверждение установления соединения), BYE (завершение вызова), CANCEL (отмена запроса), OPTIONS (проверка возможностей сервера), REGISTER (регистрация пользователя), MESSAGE (передача мгновенных сообщений), NOTIFY (отправка уведомлений), REFER (передача вызова). **Генерация SIP-трафика через:** UDP, TCP, TLS, IPv6.

		возможностью отправки пользовательских UDP-датаграмм, настройкой портов назначения).	**В дополнение к SIP-сигналам, SIPp может генерировать:** RTP-пакеты для передачи медиа-данных, PCAP-потоки записанных медиа-сессий, RFC2833 DTMF-сигналы. Также имеются встроенные шаблоны uac (клиент) и uas (сервер).
Способ генерации	Python-скрипты *(пакет создается, начиная с нижнего уровня (IP) и постепенно добавляя верхние уровни)* и отправка пакетов с помощью функций «send()» *(асинхронная отправка)*, «sendp()» *(отправка на канальном уровне)*, «sr()» *(синхронная отправка с получением ответов)*, «sr1()» *(получение первого ответа)*	bash-скрипты с параметрами: -1 *(генерация ICMP-пакетов)*, -2 *(генерация UDP-пакетов)*, -S *(установка флага SYN для TCP)*, -p (*указание порта назначения)*, -c *(количество отправляемых пакетов)*, -d *(размер данных в пакете)*, -T *(установка значения TTL)*	С помощью XML-скрипта (в UAC-режиме *(режим клиента, инициирующего вызовы)* или в UAS-режиме *(режим сервера, принимающего вызовы)*) или bash-скрипта с параметрами: -m *(количество вызовов)*, -r (*интенсивность вызовов)*, -d *(длительность вызова в миллисекундах)*, -p *(порт прослушивания)*, -i *(локальный IP-адрес)*, -sf *(путь к пользовательскому сценарию)*
Настраиваемые параметры	Любые поля заголовков: IP-заголовок *(dst, src, ttl, id, flags)*, TCP-заголовок	Протоколы *(--icmp, --udp)*, TCP-флаги *(-S, -A, -F, -R)*, параметры адресации *(-p, -s, -*	Нет, нужен внешний анализатор (Wireshark/tcpdump)

	(dport, sport, flags, seq), UDP-заголовок *(dport, sport)*. А также: настройка прокси для определенных протоколов; фильтрация пакетов через BPF-фильтры; настройка времени ожидания при отправке пакетов	*a, -l)*, параметры генерации трафика *(-c, -d, --flood, -i)*, параметры трассировки *(--traceroute, -V)*, параметры данных и подписей *(--sign)*, параметры сканирования *(-8, --scan)*	
Перехват и анализ трафика	Да, с помощью функций: «sniff()», «sr()», «show()»	Ограниченно: Hping3 показывает ответы на отправленные пакеты	Да, через сценарии XML и интеграцию в bash-скрипты
Автоматизация	С помощью Python API скриптов и Python-модуля Automaton	С помощью bash-скриптов и передачи данных через пайпы	Через XML-форматирование или с помощью интеграции с внешними системами: системы мониторинга (Zabbix, Prometheus), системы логирования (ELK Stack), CI/CD инструменты (Jenkins, GitLab CI), системы управления тестированием
Типовые сценарии применения	Анализ ARP-трафика; тестирование DHCP; исследование DNS; анализ HTTP-запросов; тестирование VPN; исследование Wi-Fi	Тестирование правил файервола; поиск открытых портов; проверка работы DNS; анализ маршрутизации; тестирование VPN-соединений; проверка работы прокси-серверов	Нагрузочное тестирование SIP-серверов, VoIP-инфраструктуры

Главный вывод. Scapy является наиболее универсальным инструментом, способным работать с различными протоколами и создавать произвольные пакеты. Hping3 имеет более узкую специализацию. SIPp же полностью сфокусирован на протоколе SIP.

Также SIPp оптимизирован для генерации большого количества SIP-сессий и обеспечивает высокую производительность при тестировании SIP-систем, в то время как Scapy и Hping3 больше ориентированы на качество и детализацию анализа, чем на массовую генерацию трафика.

Становится очевидно, что каждый из представленных инструментов имеет свою уникальную нишу применения:

- Scapy – лучший выбор для исследователей и разработчиков, которым требуется глубокий анализ сетевых протоколов и создание нестандартных пакетов.
- Hping3 – оптимальное решение для системных администраторов и специалистов по безопасности, которым нужен простой инструмент для тестирования сетевых соединений и проверки безопасности.
- SIPp – идеальный инструмент для тестирования и отладки SIP-систем, особенно в случаях, когда требуется генерация большого количества SIP-сессий и анализ производительности VoIP-систем.

Знание возможностей каждого из рассмотренных средств расширяет арсенал специалиста по сетям, позволяя решать широкий спектр задач по анализу и обеспечению работы сетей передачи информации.

Список литературы

1) Уймин, А. Г. Классификация корпоративного трафика с использованием алгоритмов машинного обучения / А. Г. Уймин // Автоматизация и информатизация ТЭК. – 2023. – № 7(600). – С. 22-29. – DOI 10.33285/2782-604X-2023-7(600)-22-29. – EDN WXXUPK. (дата обращения: 30.06.2025)
2) Biondi Ph. Scapy (официальная документация) [Электронный ресурс]. – URL: https://scapy.net (дата обращения: 30.06.2025).
3) Scapy – Википедия, свободная энциклопедия [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Scapy (дата обращения: 30.06.2025).
4) hping3 | Kali Linux Tools [Электронный ресурс]. – URL: https://www.kali.org/tools/hping3/ (дата обращения: 30.06.2025).
5) Fyodor. Hping // SecTools: Top Network Security Tools [Электронный ресурс]. – URL: https://sectools.org/tool/hping/ (дата обращения: 30.06.2025).
6) SIPp [Электронный ресурс]. – URL: https://sipp.sourceforge.net/doc/reference.html (дата обращения: 30.06.2025).