Система анализа сетевого трафика «TrActor»

Леонид Лабошин

Published with GitBook

Содержание

Введение

Описание программного продукта

Becoming a super hero is a fairly straight forward process:

```
$ give me super-powers
```

> Super-powers are granted randomly so please submit an issue if you're not happy with yours.

Once you're strong enough, save the world:

```
// Ain't no code for that yet, sorry
echo 'You got to trust me on this, I saved the world'
```

Захват сетевого трафика. Форматы файлов снимков данных Pcap, PcapNg

Программа захвата сетевых пакетов, часто называемая анализатором пакетов, анализатором сети или просто сниффером (от англ. to sniff — нюхать), — это программа, библиотека или устройство, которое получает (фактически подслушивает) пакеты данных, проходящие через определенный сегмент сети, к которому она подключена с помощью сетевой платы. Терминология, связанная со средствами захвата и анализа сетевого трафика может быть немного запутанной. Большинство программ, записывающих пакеты из сети и обрабатывающих их, производя, например, разбор заголовков или извлечение данных для последующей обработки, называются анализаторами, пакетными анализаторами, сетевыми анализаторами, снифферами. В данной работе используется термин **сниффер** для устройства, производящего получение копии необработанных пакетов с сетевой карты или сетевого интерфейса и термин сетевой анализатор для устройств, обрабатывающих полученные сниффером **снимки** или так называемые **дампы**. Таким образом, программное обеспечение для декодирования пакетных данных и их анализа, такое как tcpdump, wirehasrk, например, можно рассматривать и как снифферы (так как все они полагаются на библиотеку захвата трафика) и как анализаторы пакетов. С другой стороны, например, библиотеки, такие как Libpcap или Winpcap следует рассматривать просто как инструменты захвата трафика.

Libpcap является библиотекой захвата для UNIX-систем. Системы Windows используют порт Libpcap под названием Winpcap. Эта библиотека предоставляет программисту программный интерфейс для использования средств фильтра пакетов BSD или любой другой архитектуры, основанной на пакетном фильтре Беркли, для создания программ сетевого захвата на уровне пользователей. Libpcap был выпущен разработчиками tcpdump исследователями лаборатории Лоуренса Беркли. Libpcap предоставляет следующие возможности: захват пакетов с сетевой карты, запись и чтение пакетов в файлы снимков. Libpcap был извлечен из программы tcpdump и распространяется в виде библиотеки. Ответственным за развитие этой библиотеки является группа разработчиков tcpdump.

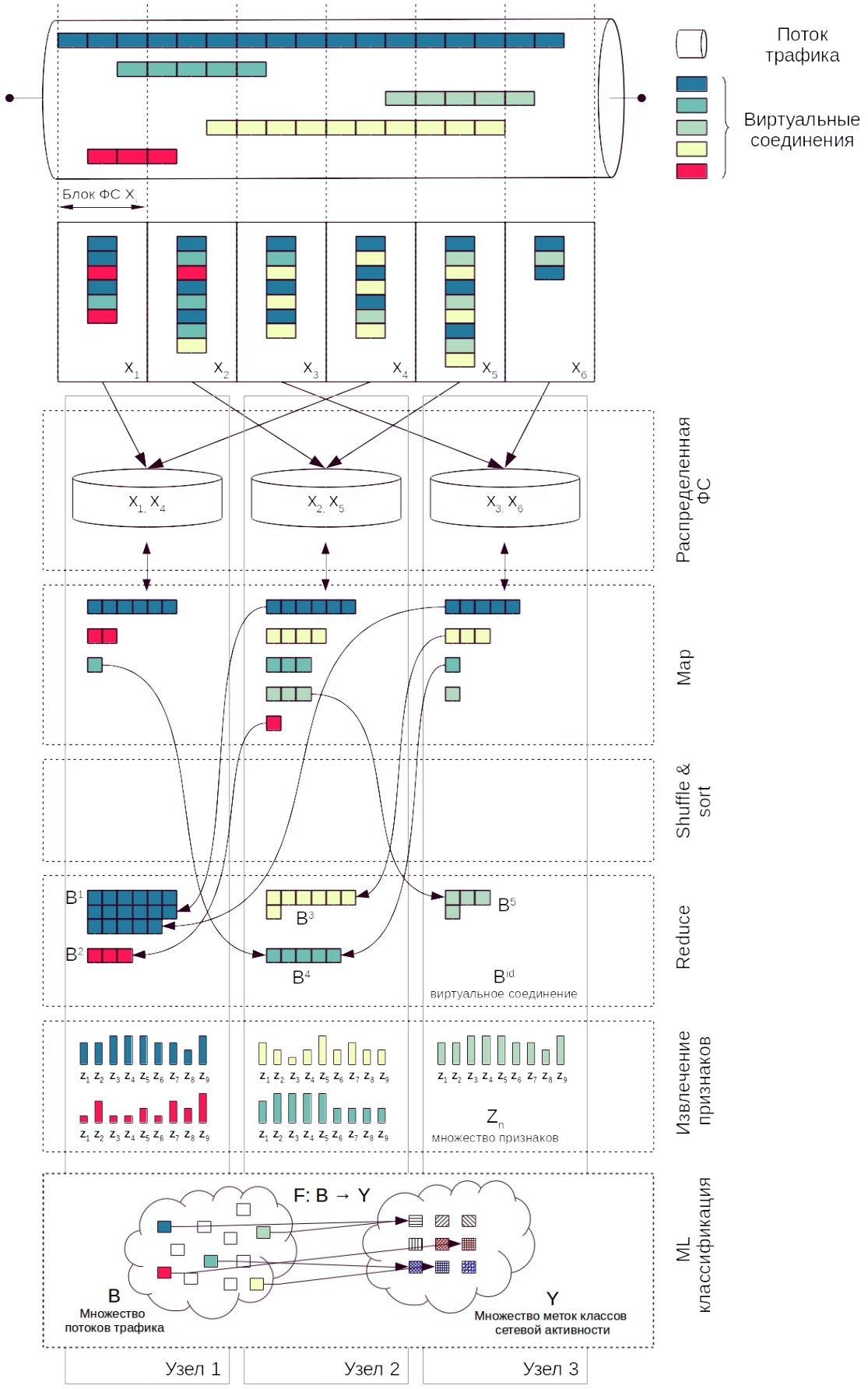
Одними из самых распространенных форматов хранения полных данных сетевого трафика являются формат Pcap (англ. Packet CAPture) и его обновленная версия PcapNg (Pcap Next generation). Оба формата представляют собой бинарные файлы, состоящие из глобального заголовка, позволяющего его идентифицировать и записей для каждого захваченного пакета (рис.).

Структура файла снимка сетевого трафика.

Далее будут рассмотрены структура и поля заголоков для обоих типов файлов, и их различия.

Формат пакетной записи формата Pcap

Формат пакетной записи формата PcapNg


Поток трафика
Виртуальные соединения
Блок ФС X
X1
X2
X3
X4
X5
X6
X1, X4
X2, X5
X3, X6
Распределенная ФС
Map
Shuffle & sort
Reduce
B1
B2
B3
B4
B5
Bid
виртуальное соединение
Извлечение признаков
z1 z2 z3 z4 z5 z6 z7 z8 z9
Zn
множество признаков
ML классификация
F: B → Y
B
Множество потоков трафика
Y
Множество меток классов сетевой активности
Узел 1
Узел 2
Узел 3

СОВ на основе сигнатур имеют важные ограничения, не позволяющие обнаруживать ранее не известные угрозы безопасности, поскольку для обнаружения новых атак требуются новые правила их обнаружения. С начала 1990 года исследователи во всем мире начали заниматься вопросами применения методов статистического анализа для обнаружения аномалий и классификации сетевого трафика.

Из предложенных методов, линейные методы, такие как модели логистического обеспечения, регрессионные модели, анализ основных компонентов или кластеризованный анализ, являются основными методами, пригодными для использования при моделировании и изучении поведения пользователей на основе данных сетевого трафике. Нелинейные методы, основанные на алгоритмах искусственного интеллекта, таких как искусственные нейронные сети, алгоритмы нечеткой логики и алгоритмы k-ближайших соседей, также были признаны эффективными для принятия решений по обнаружению вторжений в сеть.

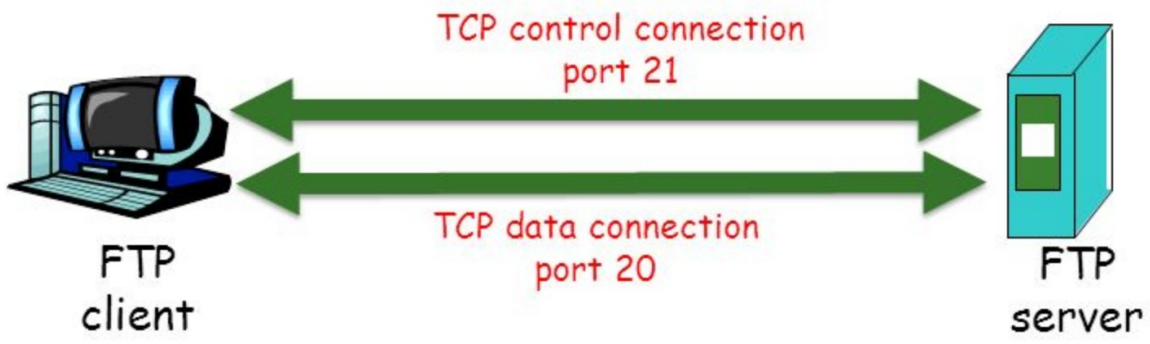
Методы интеллектуального анализа данных основанные на сочетании машинного обучения, статистического анализа и технологий баз данных для поиска шаблонов и тонких связей между полями данных сети, что позволяет прогнозировать будущие результаты.

Технология генерации признаковых описаний виртуальных соединений

Виртуальное соединение (ВС) – это логически упорядоченный обмен сообщениями между узлами сети. Трафик компьютерной сети – это совокупность виртуальных соединений.

Виртуальные соединения классифицируются на два уровня – технологические виртуальные соединения (ТВС) и информационные виртуальные соединения (ИВС).

Для реализации механизмов классификации сетевого трафика, исходные данные пакетов передаваемых данных декомпозируются в форму ИВС и ТВС. ТВС можно определить как потоки пакетов, формируемые сетевыми приложениями в рамках информационного взаимодействия. ТВС может быть представлено в виде счетного подмножества декартова произведения множества пакетов и временных меток. Такое представление характеризуется конечным набором параметров, характеризующих субъект и объект доступа, а также действие в форме потока пакетов между ними в рамках межсетевого взаимодействия. Параметрами в простейшем случае являются идентификаторы субъекта и объекта, такие как адреса, порты, и другие характеристики сетевых протоколов. Для оперативной классификации трафика, наряду с моделью ТВС, используется модель ИВС для описания взаимодействия между объектом и субъектом на уровне прикладных сервисов. Модель ИВС представляет собой совокупность ТВС, используемых одним прикладным приложением в процессе его функфционирования. Примером ИВС как совокупности ТВС является работа с FTP сервером (рис.). Протокол FTP построен на модели отдельных потоков управления и передачи данных между клиентом и сервером. В этом случае можно говорить об одном ИВС, состоящим из двух ТВС.



Технологические соединения при работе с ftp

Признаковое описание технологических виртуальных соединений на основе статистических данных потока сетевого трафика

Для

Признаковое описание информационныъ виртуальных соединений с использованием временных характеристик потока данных сетевого трафика

Подготовка данных для обучения системы классификации

Изоляция трафика приложения с ипользованием Linux Namespaces

С появлением таких утилит, как LXC и Docker стало возможным очень легко изолировать процессы в Linux системе в их собственное системное окружение. Это дало возможность запускать целый набор приложений на одной Linux машине и быть уверенным в том, что они не будут мешать друг другу, без необходимости использования виртуальных машин.

Ключевая функциональность ядра, которая позволяет добиться такой изоляции – Linux Namespaces, появилась в Linux начиная с версии 2.4.19 в 2002 году (CLONE_NEWNS), после чего с последующими обновлениями ядра добавлялись новые:

1. UTS namespaces (CLONE_NEWUTS, Linux 2.6.19)
2. IPC namespaces (CLONE_NEWIPC, Linux 2.6.19)
3. PID namespaces (CLONE_NEWPID, Linux 2.6.24)
4. Network namespaces (CLONE_NEWNET, Linux 2.6.29)
5. User namespaces (CLONE_NEWUSER, Linux 3.8)
6. cgroup namespace (CLONE_NEWCGROUP, Linux 4.0)

Аналогично тому, как chroot позволяет процессам видеть только определённую директорию как корень файловой системы – механизмы Linux namespaces позволяют выполнять операции изоляции в других механизмах операционной системы, такими как дерево процессов, сетевые интерфейсы, точки монтирования, IPC и так далее.

Для изоляции сетевого трафика приложения нас интересуют в первую очередь Network Namespaces. Network namespaces позволяют в рамках одной машины в каждом netns иметь собственные, изолированные от других:

- набор таблиц маршрутизации;
- arp-таблицу;
- правила iptables;
- устройства.

Создаем namespace:

```
ip netns add R0
```

В новом изолированном пространстве, отсутвуют сетевые интерфейсы

```
ip netns exec R0 ip a
```

> 1: lo: mtu 65536 qdisc noop state DOWN group default qlen 1000 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

Создаем пару виртуальных интерфейсов:

```
ip link add veth-a type veth peer name veth-b
```

> Тип создаваемого интерфейса - «Virtual Ethernet»

Перенесем интерфейс veth-a из initial network namespace в namespace R0:

```
ip link set veth-a netns test
```

Настроим ip-адреса на созданных виртуальных интерфейсах:

```
ip netns exec test ifconfig veth-a up 10.0.0.1 netmask 255.255.255.0
```

```
ifconfig veth-b up 10.0.0.254 netmask 255.255.255.0
```

Настроим шлюз по-умолчанию для изолированного интерфейса:

```
ip netns exec test route add default gw 10.0.0.254 dev veth-a
```

Активируем ip_forward

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

С помощью iptables настроим NAT для интерфейса.

```
iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o eth0 -j SNAT --to-source 192.168.88.80
```

Теперь в созданом netspace можно запустить процесс, трафик которого нужно записать, и программу захвата трафика, например *wireshark*

```
ip netns exec test Telegram
ip netns exec test wireshark
```

Программа захвата трафика приложения

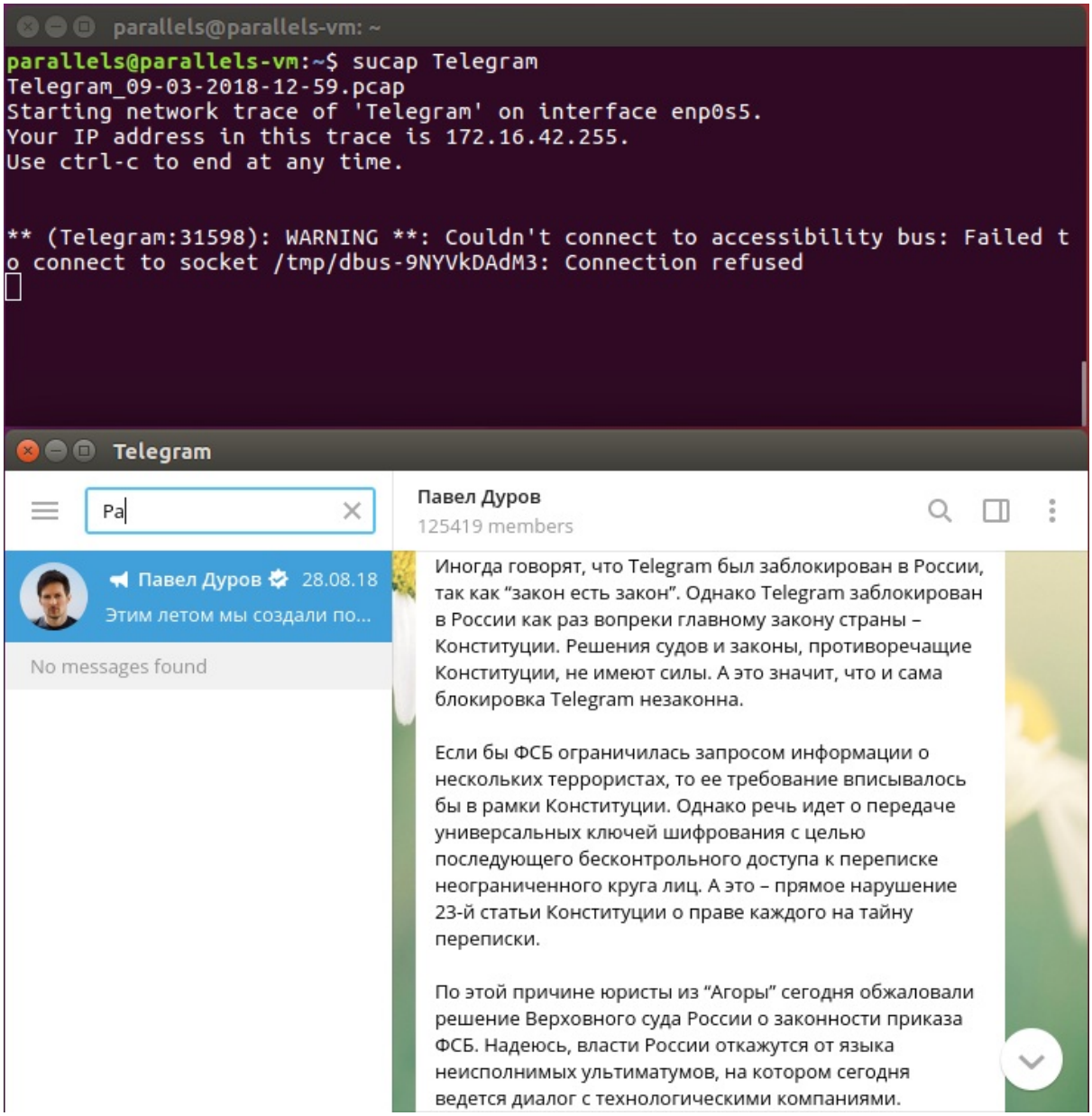


Рис.: Запуск мессенжера телеграм в утилите захвата сетевого трафика

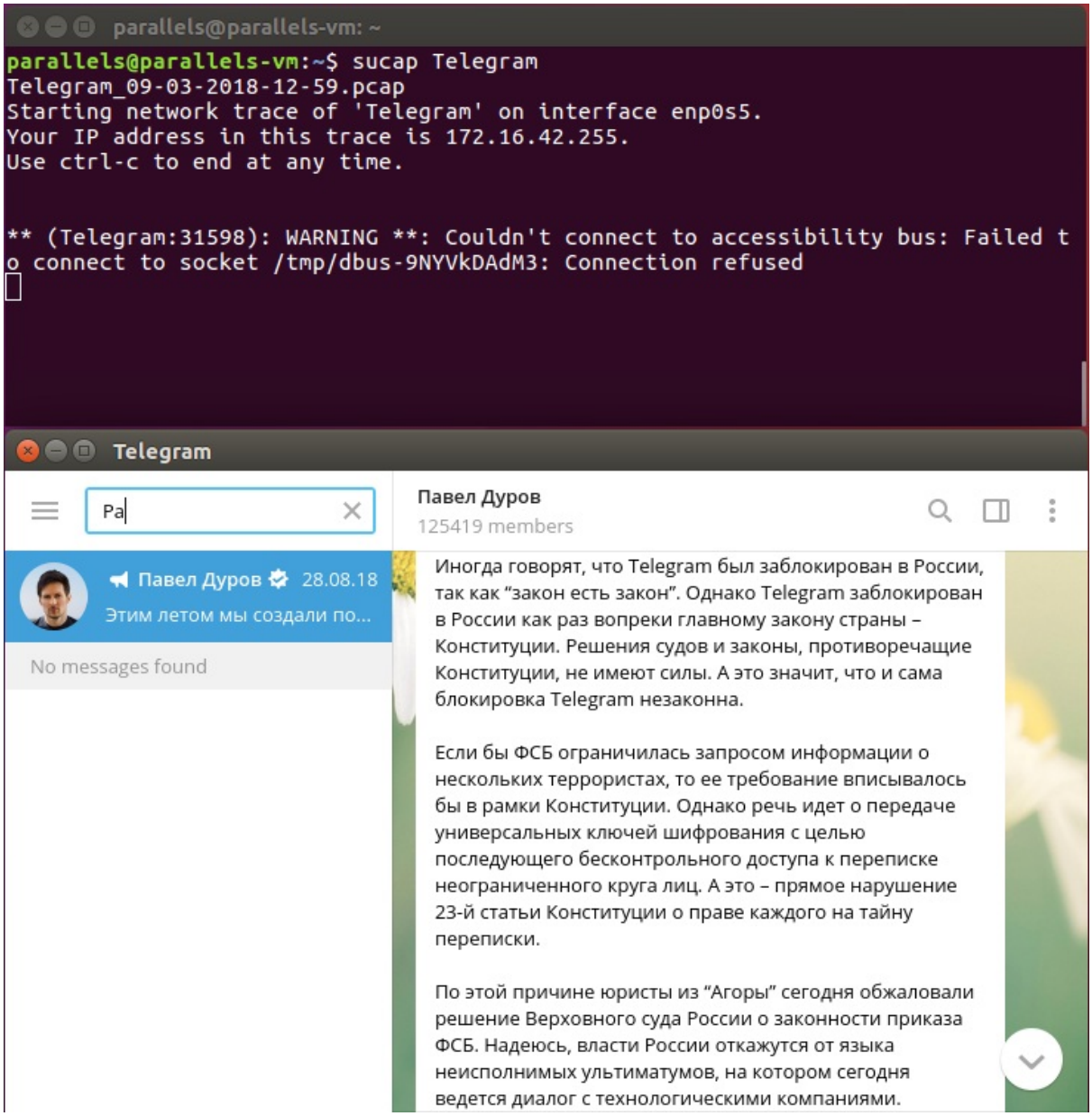


Рис.: Выделенный сетевой интерфейс для приложения Telegram

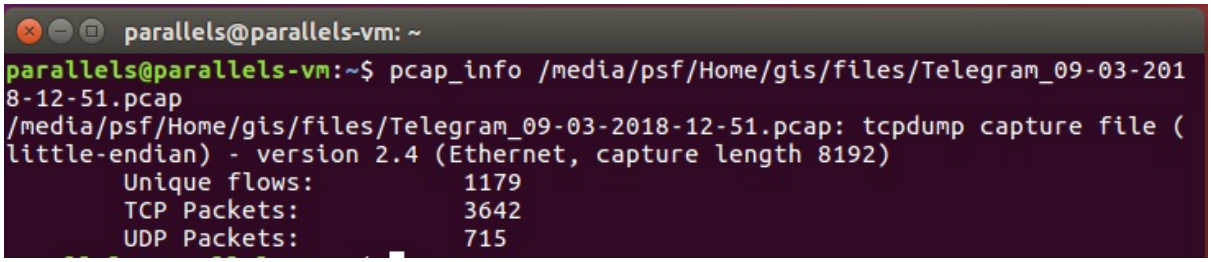


Рис.: Файл снимка сетевого трафика приложения Telegram

Экспериментальные результаты:

duration	total_fiat	total_biat	min_fiat	min_biat	max_fiat	max_biat	mean_fiat	mean_biat	flowPktsPerSecond	flowBytesPerSecond	min_flowiat	max_flowiat	mean_flowiat	std_flowiat	min_active	mean_active	max_active	std_active	min_idle	mean_idle	max_idle	std_idle	label
192013	-1	-1	-1	-1	0.	0.	0.	0.	10.416	2333.18	192013	192013	192013.	0.	-1	0	-1	0	-1	0	-1	0	P2P
237026	-1	-1	-1	-1	0.	0.	0.	0.	8.43789	1890.09	237026	237026	237026.	0.	-1	0	-1	0	-1	0	-1	0	P2P
52857	-1	-1	-1	-1	0.	0.	0.	0.	37.8379	8475.7	52857	52857	52857.	0.	-1	0	-1	0	-1	0	-1	0	P2P
64424	-1	-1	-1	-1	0.	0.	0.	0.	31.0443	6953.93	64424	64424	64424.	0.	-1	0	-1	0	-1	0	-1	0	P2P
1634113	-1	-1	-1	-1	0.	0.	0.	0.	1.22391	274.155	1634113	1634113	1634113.	0.	1634113	1634113	1634113	0	1634113	1634113	1634113	0	P2P
282237	-1	-1	-1	-1	0.	0.	0.	0.	7.08624	1587.32	282237	282237	282237.	0.	-1	0	-1	0	-1	0	-1	0	P2P
69388	-1	-1	-1	-1	0.	0.	0.	0.	28.8234	6456.45	69388	69388	69388.	0.	-1	0	-1	0	-1	0	-1	0	P2P
273443	-1	-1	-1	-1	0.	0.	0.	0.	7.31414	1638.37	273443	273443	273443.	0.	-1	0	-1	0	-1	0	-1	0	P2P
339893	-1	-1	-1	-1	0.	0.	0.	0.	5.8842	1318.06	339893	339893	339893.	0.	-1	0	-1	0	-1	0	-1	0	P2P

Введение

Описание программного продукта

Becoming a super hero is a fairly straight forward process:

```
$ give me super-powers
```

> Super-powers are granted randomly so please submit an issue if you're not happy with yours.

Once you're strong enough, save the world:

```
// Ain't no code for that yet, sorry
echo 'You got to trust me on this, I saved the world'
```

Архитектура системы

Программный интерфейс управления версии v1 системы Tractor

/files

Available endpoints

- /files/list
- /files/process
- /file/{job_id}/status
- /file/{job_id}/details
- /file/{job_id}/statistics* /file/{job_id}/statistics/labels
- /file/{job_id}/statistics/ip
- /file/{job_id}/flows/labels
- /labels/list

/files/list

Collection of available files in repository

GET:

Get the files collection

Response code: 200

application/json (application/json)


```
[
  {
    "name": "some file name",
    "extension": "pcap",
    "time_start": 1530476322,
    "time_end": 1530878322,
    "status": {
      "state": "finished",
      "started_at": 1530878325,
      "finished_at": 1530878333,
      "job_id": 12345
    },
    "size": 102
  },
  {
    "name": "some file name 2",
    "extension": "pcapng",
    "time_start": 1530476322,
    "time_end": 1530878322,
    "status": {
      "state": "new"
    },
```

```
    "size": 36
  }
]
```

List of *File*:

Name	Type	Description	Required	Pattern
name	string	file name	true	
extension	string	file extension	true	
time_start	integer	start timestamp period of file	true	
time_end	integer	end timedtamp period of file	true	
status	object	status of file	true	
size	integer	filesize in kB	true	
flows_count	integer	Count of flows	false	

status:

Name	Type	Description	Required	Pattern
state	string	Value of status	true	
started_at	integer	Start time of processing	false	
finished_at	integer	End time of processing	false	
job_id	integer	uniqie job id	false	

Response code: 404

Error (application/json)

```
{
  "status": 404,
  "code": 404,
  "message": "Not found"
}
```

***Error*:**

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

Response code: 500

Error (application/json)

```
{
  "status": 500,
  "code": 500,
```

```
  "message": "Internal server error"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

/files/process

Process of available files in repository

POST:

Start process of file

FileName (application/json)

```
{
  "file_name": "some_file_name_with_extension.pcap"
}
```

FileName:

Name	Type	Description	Required	Pattern
file_name	string	file name with extension	true	

Response code: 200

FileStatus (application/json)

```
{
  "state": "in_progress",
  "started_at": 1530878325,
  "job_id": 12345
}
```

FileStatus:

Name	Type	Description	Required	Pattern
state	string	Value of status	true	
started_at	integer	Start time of processing	false	
finished_at	integer	End time of processing	false	
job_id	integer	uniqie job id	false	

Response code: 404

Error (application/json)

```
{
  "status": 404,
  "code": 404,
  "message": "Not found"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

Response code: 500

Error (application/json)

```
{
  "status": 500,
  "code": 500,
  "message": "Internal server error"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

/file

Available endpoints

- /files/list
- /files/process
- /file/{job_id}/status
- /file/{job_id}/details
- /file/{job_id}/statistics* /file/{job_id}/statistics/labels
- /file/{job_id}/statistics/ip
- /file/{job_id}/flows/labels

- /labels/list

/file/{job_id}/status

Status of file

GET:

Get status of file

Response code: 200

FileStatus (application/json)

```
{
  "state": "in_progress",
  "started_at": 1530878325,
  "job_id": 12345
}
```

FileStatus:

Name	Type	Description	Required	Pattern
state	string	Value of status	true	
started_at	integer	Start time of processing	false	
finished_at	integer	End time of processing	false	
job_id	integer	uniqie job id	false	

Response code: 404

Error (application/json)

```
{
  "status": 404,
  "code": 404,
  "message": "Not found"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

Response code: 500

Error (application/json)

```
{
  "status": 500,
  "code": 500,
  "message": "Internal server error"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

/file/{job_id}/details

GET:

Get file details by job id

Response code: 200

File (application/json)

```
{
  "name": "some file name",
  "extension": "pcap",
  "time_start": 1530476322,
  "time_end": 1530878322,
  "status": {
    "state": "finished",
    "started_at": 1530878325,
    "finished_at": 1530878333,
    "job_id": 12345
  },
  "size": 102,
  "flows_count":30000
}
```

File:

Name	Type	Description	Required	Pattern
name	string	file name	true	
extension	string	file extension	true	
time_start	integer	start timestamp period of file	true	
time_end	integer	end timedtamp period of file	true	
status	object	status of file	true	
size	integer	filesize in kB	true	
flows_count	integer	Count of flows	false	

status:

Name	Type	Description	Required	Pattern
state	string	Value of status	true	
started_at	integer	Start time of processing	false	
finished_at	integer	End time of processing	false	
job_id	integer	uniqie job id	false	

Response code: 404

Error (application/json)

```
{
  "status": 404,
  "code": 404,
  "message": "Not found"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

Response code: 500

Error (application/json)

```
{
  "status": 500,
  "code": 500,
  "message": "Internal server error"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

/file/{job_id}/statistics

GET:

Get general statistic of processed file

Response code: 200

application/json (application/json)


```
[
  {
    "title": "protocols",
    "content": [
      {
        "label": "ftp",
        "count": 10
      },
      {
        "label": "udp",
        "count": 16
      }
    ]
  },
  {
    "title": "clients",
    "content": [
      {
        "label": "ip",
        "count": 10
      }
    ]
  }
]
```


List of *StatisticItem*:

Name	Type	Description	Required	Pattern
title	string	Title of statistic item	true	
content	items array	Iterable array of content items	true	

items:

Name	Type	Description	Required	Pattern
label	string		true	
count	integer		true	

Response code: 404

Error (application/json)


```
{
  "status": 404,
  "code": 404,
  "message": "Not found"
}
```


***Error*:**

Name	Type	Description	Required	Pattern

status	integer		true	
code	integer		true	
message	string		true	

Response code: 500

Error (application/json)

```
{
  "status": 500,
  "code": 500,
  "message": "Internal server error"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

/file/{job_id}/statistics/labels

GET:

Get statistics based by Labels

Query parameters

Name	Type	Description	Required	Examples
label	string		true	

Response code: 200

application/json (application/json)

```
[
  {
    "title": "protocols",
    "content": [
      {
        "label": "ftp",
        "count": 10
      },
      {
        "label": "udp",
        "count": 16
      }
    ]
  },
  {
```

```
    "title": "clients",
    "content": [
      {
        "label": "ip",
        "count": 10
      }
    ]
  }
]
```

List of *StatisticItem*:

Name	Type	Description	Required	Pattern
title	string	Title of statistic item	true	
content	items array	Iterable array of content items	true	

items:

Name	Type	Description	Required	Pattern
label	string		true	
count	integer		true	

Response code: 404

Error (application/json)

```
{
  "status": 404,
  "code": 404,
  "message": "Not found"
}
```

***Error*:**

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

Response code: 500

Error (application/json)

```
{
  "status": 500,
  "code": 500,
  "message": "Internal server error"
}
```

***Error*:**

Name	Type	Description	Required	Pattern

status	integer		true	
code	integer		true	
message	string		true	

/file/{job_id}/statistics/ip

GET:

Get statistics based by IP

Query parameters

Name	Type	Description	Required	Examples
ip	string		true	

Response code: 200

DetailsIp (application/json)

```
{
  "country": "USA",
  "city": "NewYork",
  "user_agent": "Macintosh/Mozilla1.0",
  "domain_name": "example.com",
  "statistics": [
    {
      "title": "protocols",
      "content": [
        {
          "label": "ftp",
          "count": 10
        }
      ]
    },
    {
      "title": "cients",
      "content": [
        {
          "label": "ip",
          "count": 10
        }
      ]
    }
  ]
}
```

DetailsIp:

Name	Type	Description	Required	Pattern
country	string	County	true	
city	string	City	true	
user_agent	string	User Agent	false	
domain_name	string	Domain name	false	

statistics	items array	Iterable array of statistics items	true

items:

Name	Type	Description	Required	Pattern
title	string	Title of statistic item	true	
content	items array	Iterable array of content items	true	

items:

Name	Type	Description	Required	Pattern
label	string		true	
count	integer		true	

Response code: 404

Error (application/json)

```
{
  "status": 404,
  "code": 404,
  "message": "Not found"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

Response code: 500

Error (application/json)

```
{
  "status": 500,
  "code": 500,
  "message": "Internal server error"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

/file/{job_id}/flows/labels

GET:

Get statistics based by Labels

Query parameters

Name	Type	Description	Required	Examples
label	string		true	

Response code: 200

application/json (application/json)

```
[
  {
    "client_ip": "192.168.88.80",
    "client_port": 34567,
    "server_ip": "8.8.8.8",
    "server_port": 80,
    "time_start": 12345678,
    "time_end": 12345679,
    "comment": "some comment"
  },
  {
    "client_ip": "192.167.88.82",
    "client_port": 34,
    "server_ip": "1.1.1.1",
    "server_port": 9090,
    "time_start": 12345678,
    "time_end": 12345679,
    "comment": "some comment"
  }
]
```

List of *DetailsLabelItem*:

Name	Type	Description	Required	Pattern
client_ip	string	Client IP	true	
client_port	integer	Client Port	true	
server_ip	string	Server IP	true	
server_port	integer	Server Port	true	
time_start	integer	Session start time	true	
time_end	integer	Session end time	true	
comment	string	Comment	true	

Response code: 404

Error (application/json)

```
{
```

```
  "status": 404,
  "code": 404,
  "message": "Not found"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

Response code: 500

Error (application/json)

```
{
  "status": 500,
  "code": 500,
  "message": "Internal server error"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	

/labels

Available endpoints

- /files/list
- /files/process
- /file/{job_id}/status
- /file/{job_id}/details
- /file/{job_id}/statistics* /file/{job_id}/statistics/labels
- /file/{job_id}/statistics/ip
- /file/{job_id}/flows/labels
- /labels/list

/labels/list

GET:

Get labels collection

Response code: 200

application/json (application/json)

```
[
  {
    "group_name": "social",
    "labels": [
      "facebook",
      "vk"
    ]
  },
  {
    "group_name": "peer2peer",
    "labels": [
      "bittorrent",
      "skype"
    ]
  },
  {
    "group_name": "peer2peer",
    "labels": [
      "vpn",
      "sokcs5"
    ]
  },
  {
    "group_name": "other",
    "labels": [
      "ssh",
      "telnet"
    ]
  }
]
```

List of *Label*:

Name	Type	Description	Required	Pattern
group_name	string	Group name of certain labels	true	
labels	array	array of labels	true	

Response code: 404

Error (application/json)

```
{
  "status": 404,
  "code": 404,
  "message": "Not found"
}
```

***Error*:**

Name	Type	Description	Required	Pattern
status	integer		true	

code	integer		true	
message	string		true	

Response code: 500

Error (application/json)

```
{
  "status": 500,
  "code": 500,
  "message": "Internal server error"
}
```

Error:

Name	Type	Description	Required	Pattern
status	integer		true	
code	integer		true	
message	string		true	