Инфосистемы Джет × JET SECURITY TEAM

МЕТОДИКА НАГРУЗОЧНОГО ТЕСТИРОВАНИЯ РЕШЕНИЙ NGFW/UTM ВЕРСИЯ 2.0

ОБЩЕЕ ОПИСАНИЕ

Настоящая методика разработана для обеспечения единообразия и унификации проводимых испытаний NGFW/UTM. При соблюдении данной методики все тестируемые решения будут находиться в равных условиях.

ЦЕЛЬ ТЕСТИРОВАНИЯ

Цель нагрузочного тестирования — определить производительность решения NGFW/UTM, которое выполняет функции защиты периметра и контроля доступа в Интернет. В рамках нагрузочного тестирования NGFW/UTM рассматривался сценарий, когда устройство используется как периметровый межсетевой экран с включением доступных модулей безопасности.

ЗАДАЧА ТЕСТИРОВАНИЯ

Задача нагрузочного тестирования — определить предельную производительность NGFW/UTM, при превышении которой его работа становится нестабильной либо не выполняются заявленные функции безопасности.

ОГРАНИЧЕНИЯ

Тестирование производилось на базе EMIX-трафика (Enterprise Mix), созданного на основе профиля реального трафика в корпоративной сети «Инфосистемы Джет». Условие достижения предела производительности — потери пакетов/сессий более 1%.

Полученные значения не могут быть единственно верными, так как на результат влияет множество факторов:

- методика тестирования: производители NGFW/UTM используют собственные методики нагрузочного тестирования, которые отличаются от данной;
- профиль трафика: в каждой организации формируется свой профиль трафика;
- настройка NGFW/UTM: включенные модули, используемое количество сигнатур IPS и т. п.;
- настройка правил МЭ: количество правил, широта правил (количество трафика, проверяемое каждым правилом, особенно актуально для SSL Inspection, AppControl и т. п.).

Предполагаем, что полученные значения на базе EMIX-трафика ближе к реальной производительности, чем полученные на синтетическом трафике. Однако для каждой организации такие значения будут отличаться из-за различных профилей трафика и настройки устройства.

ОПИСАНИЕ СТЕНДА

Схема тестового стенда приведена на **рисунке 1.**

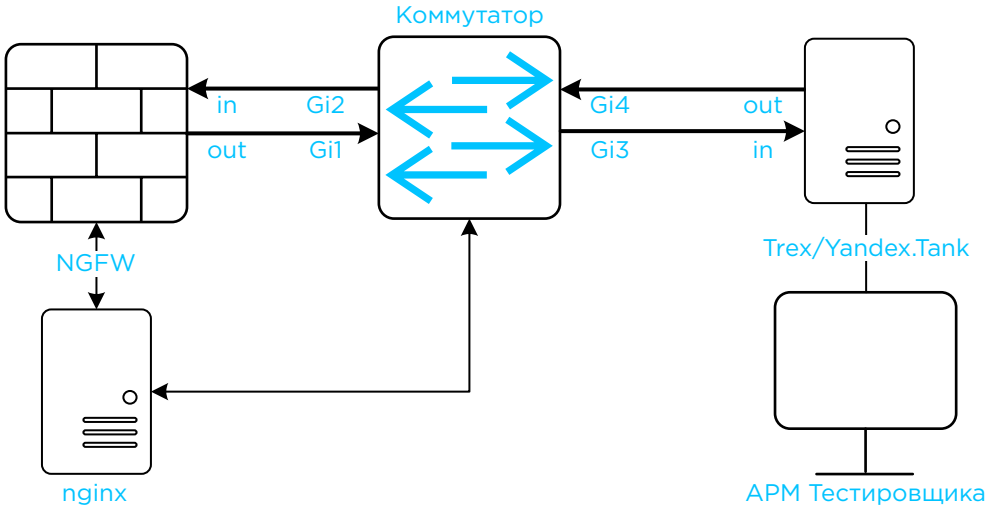

Рис. 1. Принципиальная схема тестового стенда

Характеристики серверов и используемого ПО приведены в **Таблице 1**, IP-адресация в **Таблице 2**.

Таблица 1. Характеристики серверов и используемого ПО

Тип	Описание
Характеристики сервера Trex/Yandex.Tank	98 GB RAM Intel(R) Xeon(R) CPU E5-2630 @ 2.30GHz Thread(s) per core: 2 Core(s) per socket: 6 Socket(s): 2 NUMA: NUMA node(s): 2
Версии ПО на сервере Trex/Yandex.Tank	Ubuntu 23.10 Mantic wrk 4.2.0 TRex v3.05 OpenSSL 3.0.10 nginx version: nginx/1.24.0 (Ubuntu)
Характеристики ВМ Nginx	RAM: 16 GB vCores: 32 (2 GHz)
Версии ПО на ВМ Nginx	Ubuntu 22.04 jammy OpenSSL 3.0.10 nginx version: nginx/1.18.0 (Ubuntu)

Таблица 2. IP-адресация

№	Адресация	Назначение
1	16.0.0.1 -16.0.255.191	Адреса, используемые генератором трафика в качестве клиентов
2	48.0.0.1 - 48.1.255.191	Адреса, используемые генератором трафика в качестве серверов
3	192.168.253.109/24	Адреса, используемые генератором трафика в качестве клиента при определении SSL TPS
4	16.0.255.192/26	Адреса, используемые генератором трафика в качестве клиента для pcap с вредоносной нагрузкой
5	48.1.255.192/26	Адреса, используемые генератором трафика в качестве сервера для pcap с вредоносной нагрузкой
6	192.168.1.228/24	ВМ с сервисом HTTPS для определения SSL TPS

НАСТРОЙКА ТЕСТОВОГО СТЕНДА

Установка TRex основывается на конфигурациях и инструкциях, приведенных по адресу **https://github.com/OlegKashtanov/ngfw-trex-perf**. Для эмуляции вредоносной нагрузки в ходе выполнения сценариев тестирования были подготовлены более 80 различных вариантов атак в виде pcap-файлов. Такое разнообразие необходимо, чтобы сгенерировать блокировки для всех тестируемых решений.
Конфигурация файла /etc/trex_cfg.yaml:

```
### Config file generated by dpdk_setup_ports.py ###

- version: 2
  interfaces: ['11:00.0', '11:00.1']
  port_bandwidth_gb: 10
  port_info:
      - ip: 192.168.253.109
        default_gw: 192.168.253.114
      - ip: 192.168.254.109
        default_gw: 192.168.254.114

  platform:
      master_thread_id: 0
      latency_thread_id: 6
      dual_if:
        - socket: 0
          threads: [1,2,3,4,5,12,13,14,15,16,17]
        - socket: 1
          threads: [7,8,9,10,11,18,19,20,21,22,23]
```

Yandex.Tank используется в Docker-исполнении. Настройка выполняется в соответствии с документацией производителя **(https://yandextank.readthedocs.io/en/latest/install.html)**. Базовый профиль нагрузки приведен в Приложении 1.

Настройки Nginx для измерения SSL TPS:

1. **Конфигурация файла /etc/nginx/sites-available/default:**

```
server {
        listen 443 ssl reuseport backlog=300072 deferred
so_keepalive=off;
        include snippets/self-signed.conf;
        include snippets/ssl-params.conf;
        keepalive_requests 0;
        keepalive_timeout 0s;
        tcp_nopush on;
        tcp_nodelay on;
        lingering_close off;
        lingering_time 1;
        ssl_session_cache   off;
        ssl_session_timeout  5m;
        ssl_session_tickets off;
        ssl_prefer_server_ciphers   on;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name _;
        location / {
                try_files $uri $uri/ =404;
        }
}
```

2. Конфигурация файла /etc/nginx/nginx.conf:

```
user www-data;
worker_priority -10;
worker_processes 28;
worker_cpu_affinity 11111111111111111111111111110000;
worker_rlimit_nofile 400000;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        use epoll;
        worker_connections 8192;
        multi_accept off;
        accept_mutex off;
}

http {

        ssl_buffer_size 65536;
        sendfile on;
        types_hash_max_size 2048;
        include /etc/nginx/mime.types;
        default_type application/octet-stream;
        access_log off;
        error_log /var/log/nginx/error.log crit;
        gzip off;
        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}
```

3. Конфигурация файла /etc/nginx/nginx.conf:

```
ssl_protocols       TLSv1.3;
ssl_ciphers         LOW:MED:HIGH;
```

4. Конфигурация файла /etc/nginx/snippets/self-signed.conf:

```
ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
```

Перед проведением нагрузочного тестирования выполняется проверка корректности работы нагрузочного стенда следующим образом (для SSL TPS иначе):

- в конфигурации TRex в качестве nexthop указываются L3-интерфейсы коммутатора;
- на коммутаторе настраиваются маршруты для сетей 16.0.0.0/8 и 48.0.0.0/8;
- проводятся нагрузочные тесты, где лимитирующими факторами являются аппаратная платформа TRex (например, предельная загрузка CPU), сетевые интерфейсы (например, утилизация полосы на границе пропускной способности интерфейса).

Проверка корректности работы нагрузочного стенда для SSL TPS проводится также путем изменения статической маршрутизации на L3-коммутатор, но по цепочке Yandex.Tank <–> SW(L3) <–> Nginx.

Для фиксации результатов нагрузочного тестирования использовались следующие инструменты:

1. Метрики производительности оборудования отправлялись на Zabbix-сервер с дальнейшей визуализацией в Grafana.
2. Дополнительно — ручной мониторинг процессов нагрузки (для решений, где это возможно, т. к. деградация производительности часто возникает в значениях от 90% на ядрах обработки data/control plane, что Zabbix не всегда может отобразить, не усредняя).
3. Результаты TRex транслировались и анализировались с помощью influxdb + Grafana.
4. Результаты тестов Yandex.Tank (SSL TPS) анализировались с помощью ресурса overload.yandex.net.

НАСТРОЙКА ОБЪЕКТА ТЕСТИРОВАНИЯ

1. **ACL:**
   1.1. Для тестов создаются правила следующего типа. Количество правил зависит от теста. Правила создаются так, чтобы весь трафик обрабатывался последним правилом.

№	Имя	Источник	Назначение	Сервис	Действие	COB	Logs
1	Test_1	10.10.0.1/24	10.10.1.1/24	Any	Отбросить	Выкл.	Вкл.
…							
N-1	Test_N-1	10.10.0.199/24	10.10.1.199/24	Any	Отбросить	Выкл.	Вкл.
N	EndRule	16.0.0.0/8	Any	Any	Пропустить	Вкл.	Вкл.

2. **IPS:**
   2.1. Настроить использование рекомендуемого компанией-производителем набора сигнатур (баланс между защитой и производительностью).

2.2. Подготовить три pcap-атаки, которые выявляются тестируемым IPS. Для это произвести тестовый прогон pcap-атак.
2.3. Включить IPS.

3. **Антивирус:**
   3.1. Подготовить три pcap-атаки, которые детектируются тестируемым антивирусом. Для это произвести тестовый прогон pcap-атак.
   3.2. Включить потоковый антивирус.

4. **AppControl:**
   4.1. Настроить блокировку по протоколу RDP.
   4.2. Включить AppControl.

5. **Контент-фильтрация:**
   5.1. Включить блокировку следующих категорий: TOR, VPN, torrent, развлекательный контент или аналоги, не более пяти категорий.
   5.2. URL-фильтрация: создать правило на блокировку URL из pcap-файла (192.168.31.110, 192.168.31.110:10000, cybercamp.jet.su, cybercamp.jet.su:443).
   5.3. Тип контента: создать правило на блокировку типа контента из pcap-файла.

6. **SSL TPS:**
   6.1. При тестировании использовать TLS 1.2.
   6.2. Включить инспектирование SSL-трафика с источника 192.168.253.109.
   6.3. С помощью Yandex.Tank подать нагрузку на Nginx-сервер в течение 300 с.

7. **Журналирование включить для всех правил.**

ПЛАН ТЕСТИРОВАНИЯ

1. Настроить объект тестирования и подключить к Zabbix.
2. Определить набор детектируемых pcap-атак каждым из проверяемых модулей (не менее трех на каждый проверяемый функционал). Для этого выполнить команду с последующем анализом результатов ее выполнения:

```
/opt/trex/trex-perf/trex_run_v1.2.py -f /home/toor/emix_jet.py -m 5 -a
/opt/attacks/ --ngfw-hostname "NGFW name_AttacksOnce" --attacks-once
```

3. Проверка ACL:
   3.1. Подать EMIX Throughput для определения граничных значений в течение 30 с. при подборе предельного значения и 300 с. после его определения для фиксации стабильности работы, записать последние значения:
- EMIX Throughput (описание профиля в приложении 1);
- Concurrent connection;
- Connection per second;
- Packet per second.

4. Проверка IPS[1]:
   4.1. Подать максимальную нагрузку EMIX Throughput и детектируемый набор pcap-атак для IPS в течение 300 с. Критерии успешности: NGFW/UTM демонстрирует производительность, определенную на шаге 3, все pcap-атаки заблокированы.

[1] Проверки 4-7 могут быть совмещены

5. Проверка антивируса:
   5.1. Подать максимальную нагрузку EMIX Throughput и детектируемый набор pcap-атак для антивируса в течение 300 с. Критерии: NGFW/UTM демонстрирует производительность, определенную на шаге 3, все pcap-атаки заблокированы.

6. Проверка AppControl:
   6.1. Подать максимальную нагрузку EMIX Throughput и pcap-файл с RDP-трафиком в течение 300 с. Критерии: NGFW/UTM демонстрирует производительность, определенную на шаге 3, все попытки использования RDP-приложения были заблокированы.

7. Проверка контентной фильтрации:
   7.1. Подать максимальную нагрузку EMIX Throughput и pcap-файл с целевым трафиком в течение 300 с. Критерии: NGFW/UTM демонстрирует производительность, определенную на шаге 3, все попытки открытия web-страниц по URL, по ключевым словам, по типу контента были заблокированы.

8. Проверка SSL TPS:
   8.1. С помощью Yandex.Tank подать нагрузку на NGINX-сервер в течение 300 с по профилю (описание профиля в приложении 2).
   8.2. После определения SSL TPS записать последние значения.

ТЕСТ-КЕЙСЫ

Результаты тестов внести в таблицу

Таблица 3. Результаты тестов

Определение максимальной пропускной способности (EMIX)	EMIX Throughput, Gb/s	Concurrent connection, k	Connection per second, k c/s	Packet per second, kp/s
PERF_2. 200 правил МЭ				
PERF_4. 200 правил МЭ и 20 правил NAT				
PERF_6. 2 000 правил МЭ				
PERF_8. 2 000 правил МЭ и 200 правил NAT				
PERF_10. 5 000 правил МЭ				
PERF_13. 5 000 правил МЭ и 500 правил NAT				
PERF_14. 10 000 правил МЭ				
PERF_16. 10 000 правил МЭ и 1 000 правил NAT				

PERF_1. Определение максимальной пропускной способности (EMIX). 200 правил МЭ

Задача: определить максимальную пропускную способность для трафика EMIX.

Условия теста: тест проводится со всеми включенными модулями безопасности и 200 правилами МЭ.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_1" -m 140
```

где

/opt/trex/trex-perf/trex_run_v1.2.py – путь до скрипта запуска;
-d 300 – длительность теста в секундах;
-f /home/toor/emix_jet.py – путь до EMIX-профиля;
--ngfw-hostname "NGFW name_PERF_1" – в кавычках указать название тестируемого NGFW/UTM и номер теста, например, "Continent 4.1.9_PERF_1"
-m 140 – мультипликатор задающий объем нагрузки (10 единиц примерно равны 500 Мбит/с).

Процесс теста:

- Запустить генератор трафика:
  - Профиль трафика emix_jet.py.
- В статистике генератора зафиксировать:
  - Текущее значение пропускной способности (в Гбит/с);
  - Число потерь.
- Используя бинарный поиск, найти значение пропускной способности, при котором потери сессий близки к 1%.
- Время прохождения каждой итерации теста — 5 минут.

PERF_2. Работоспособность функций безопасности под нагрузкой. 200 правил МЭ

Задача: проверить работоспособность функций безопасности (IPS, AV, AppControl, URL Filtration) при максимальной нагрузке, определенной в тесте PERF_1.

Условия теста: тест проводится со всеми включенными модулями безопасности и 200 правилами МЭ.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_2" -a /home/toor/attacks_NGFW_name/ -m 140
```

где:

```
-a /home/toor/attacks_NGFW_name/ – путь до каталога с отобранными атаками,
attacks_NGFW_name – уникальное имя каталога с отобранными pcap-файлами атак.
```

Процесс теста:

- Используя генератор трафика, запустить нагрузку, равную 100% определенной в тесте PERF_1:
  - Профиль трафика emix_jet.py.
- Параллельно направить через NGFW/UTM трафик pcap-атак, определенных в процессе настройки NGFW/UTM.

PERF_3. Определение максимальной пропускной способности (EMIX). 200 правил МЭ и 20 правил NAT

Задача: определить максимальную пропускную способность для трафика EMIX.

Условия теста: тест проводится со всеми включенными модулями безопасности, 200 правилами МЭ и 20 правилами NAT.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_3" -m 140
```

Процесс теста:

- Запустить генератор трафика:
  - Профиль трафика emix_jet.py.
- В статистике генератора зафиксировать:
  - Текущее значение пропускной способности (в Гбит/с);
  - Число потерь.
- Используя бинарный поиск, найти значение пропускной способности, при котором потери сессий близки к 1%.
- Время прохождения каждой итерации теста — 5 минут.

PERF_4. Работоспособность функций безопасности под нагрузкой. 200 правил МЭ и 20 правил NAT

Задача: проверить работоспособность функций безопасности (IPS, AV, AppControl, URL Filtration) при максимальной нагрузке, определенной в тесте PERF_3.

Условия теста: тест проводится со всеми включенными модулями безопасности, 200 правилами МЭ и 20 правилами NAT.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_4" -a /home/toor/attacks_NGFW_name/ -m 140
```

Процесс теста:

- Используя генератор трафика, запустить нагрузку, равную 100% определенной в тесте PERF_3:
  - Профиль трафика emix_jet.py.
- Параллельно направить через NGFW/UTM трафик pcap-атак, определенных в процессе настройки NGFW/UTM.

PERF_5. Определение максимальной пропускной способности (EMIX). 2 000 правил МЭ

Задача: определить максимальную пропускную способность для трафика EMIX.

Условия теста: тест проводится со всеми включенными модулями безопасности и 2 000 правилами МЭ.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_5" -m 140
```

Процесс теста:

- Запустить генератор трафика:
  - Профиль трафика emix_jet.py.
- В статистике генератора зафиксировать:
  - Текущее значение пропускной способности (в Гбит/с);
  - Число потерь.
- Используя бинарный поиск, найти значение пропускной способности, при котором потери сессий близки к 1%.
- Время прохождения каждой итерации теста — 5 минут.

PERF_6. Работоспособность функций безопасности под нагрузкой. 2 000 правил МЭ

Задача: проверить работоспособность функций безопасности (IPS, AV, AppControl, URL Filtration) при максимальной нагрузке, определенной в тесте PERF_5.

Условия теста: тест проводится со всеми включенными модулями безопасности и 2000 правилами МЭ.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_6" -a /home/toor/attacks_NGFW_name/ -m 140
```

Процесс теста:

- Используя генератор трафика, запустить нагрузку, равную 100% определенной в тесте PERF_5:
  - Профиль трафика emix_jet.py.
- Параллельно направить через NGFW/UTM трафик pcap-атак, определенных в процессе настройки NGFW/UTM.

PERF_7. Определение максимальной пропускной способности (EMIX). 2 000 правил МЭ и 200 правил NAT

Задача: определить максимальную пропускную способность для трафика EMIX.

Условия теста: тест проводится со всеми включенными модулями безопасности, 2 000 правилами МЭ и 200 правилами NAT.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_7" -m 140
```

Процесс теста:

- Запустить генератор трафика:
  - Профиль трафика emix_jet.py.
- В статистике генератора зафиксировать:
  - Текущее значение пропускной способности (в Гбит/с);
  - Число потерь.
- Используя бинарный поиск, найти значение пропускной способности, при котором потери сессий близки к 1%.
- Время прохождения каждой итерации теста — 5 минут.

PERF_8. Работоспособность функций безопасности под нагрузкой. 2 000 правил МЭ и 200 правил NAT

Задача: проверить работоспособность функций безопасности (IPS, AV, AppControl, URL Filtration) при максимальной нагрузке, определенной в тесте PERF_7.

Условия теста: тест проводится со всеми включенными модулями безопасности, 2 000 правилами МЭ и 200 правилами NAT.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_8" -a /home/toor/attacks_NGFW_name/ -m 140
```

Процесс теста:

- Используя генератор трафика, запустить нагрузку, равную 100% определенной в тесте PERF_7:
  - Профиль трафика emix_jet.py.
- Параллельно направить через NGFW/UTM трафик pcap-атак, определенных в процессе настройки NGFW/UTM.

PERF_9. Определение максимальной пропускной способности (EMIX). 5 000 правил МЭ

Задача: определить максимальную пропускную способность для трафика EMIX.

Условия теста: тест проводится со всеми включенными модулями безопасности и 5 000 правилами МЭ.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_9" -m 140
```

Процесс теста:

- Запустить генератор трафика:
  - Профиль трафика emix_jet.py.
- В статистике генератора зафиксировать:
  - Текущее значение пропускной способности (в Гбит/с);
  - Число потерь.
- Используя бинарный поиск, найти значение пропускной способности, при котором потери сессий близки к 1%.
- Время прохождения каждой итерации теста — 5 минут.

PERF_10. Определение максимальной пропускной способности (EMIX). 5 000 правил МЭ

Задача: проверить работоспособность функций безопасности (IPS, AV, AppControl, URL Filtration) при максимальной нагрузке, определенной в тесте PERF_9.

Условия теста: тест проводится со всеми включенными модулями безопасности и 5 000 правилами МЭ.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_10" -a /home/toor/attacks_NGFW_name/ -m 140
```

Процесс теста:

- Используя генератор трафика, запустить нагрузку, равную 100% определенной в тесте PERF_9:
  - Профиль трафика emix_jet.py.
- Параллельно направить через NGFW/UTM трафик pcap-атак, определенных в процессе настройки NGFW/UTM.

PERF_11. Определение максимальной пропускной способности (EMIX). 5 000 правил МЭ и 500 правил NAT

Задача: определить максимальную пропускную способность для трафика EMIX.

Условия теста: тест проводится со всеми включенными модулями безопасности, 5 000 правилами МЭ и 500 правилами NAT.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_11" -m 140
```

Процесс теста:

- Запустить генератор трафика:
  - Профиль трафика emix_jet.py.
- В статистике генератора зафиксировать:
  - Текущее значение пропускной способности (в Гбит/с);
  - Число потерь.
- Используя бинарный поиск, найти значение пропускной способности, при котором потери сессий близки к 1%.
- Время прохождения каждой итерации теста — 5 минут.

PERF_12. Работоспособность функций безопасности под нагрузкой. 5 000 правил МЭ и 500 правил NAT

Задача: проверить работоспособность функций безопасности (IPS, AV, AppControl, URL Filtration) при максимальной нагрузке, определенной в тесте PERF_11.

Условия теста: тест проводится со всеми включенными модулями безопасности, 5 000 правилами МЭ и 500 правилами NAT.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_12" -a /home/toor/attacks_NGFW_name/ -m 140
```

Процесс теста:

- Используя генератор трафика, запустить нагрузку, равную 100% определенной в тесте PERF_11:
  - Профиль трафика emix_jet.py.
- Параллельно направить через NGFW/UTM трафик pcap-атак, определенных в процессе настройки NGFW/UTM.

PERF_13. Определение максимальной пропускной способности (EMIX). 10 000 правил МЭ

Задача: определить максимальную пропускную способность для трафика EMIX.

Условия теста: тест проводится со всеми включенными модулями безопасности и 10 000 правилами МЭ.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_13" -m 140
```

Процесс теста:

- Запустить генератор трафика:
  - Профиль трафика emix_jet.py.
- В статистике генератора зафиксировать:
  - Текущее значение пропускной способности (в Гбит/с);
  - Число потерь.
- Используя бинарный поиск, найти значение пропускной способности, при котором потери сессий близки к 1%.
- Время прохождения каждой итерации теста — 5 минут.

PERF_14. Работоспособность функций безопасности под нагрузкой. 10 000 правил МЭ

Задача: проверить работоспособность функций безопасности (IPS, AV, AppControl, URL Filtration) при максимальной нагрузке, определенной в тесте PERF_13.

Условия теста: тест проводится со всеми включенными модулями безопасности и 10 000 правилами МЭ.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_14" -a /home/toor/attacks_NGFW_name/ -m 140
```

Процесс теста:

- Используя генератор трафика, запустить нагрузку, равную 100% определенной в тесте PERF_13:
  - Профиль трафика emix_jet.py.
- Параллельно направить через NGFW/UTM трафик pcap-атак, определенных в процессе настройки NGFW/UTM.

PERF_15. Определение максимальной пропускной способности (EMIX). 10 000 правил МЭ и 1 000 правил NAT

Задача: определить максимальную пропускную способность для трафика EMIX.

Условия теста: тест проводится со всеми включенными модулями безопасности, 10 000 правилами МЭ и 1 000 правилами NAT.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_15" -m 140
```

Процесс теста:

- Запустить генератор трафика:
  - Профиль трафика emix_jet.py.
- В статистике генератора зафиксировать:
  - Текущее значение пропускной способности (в Гбит/с);
  - Число потерь.

- Используя бинарный поиск, найти значение пропускной способности, при котором потери сессий близки к 1%.
- Время прохождения каждой итерации теста — 5 минут.

PERF_16. Работоспособность функций безопасности под нагрузкой. 10 000 правил МЭ и 1 000 правил NAT

Задача: проверить работоспособность функций безопасности (IPS, AV, AppControl, URL Filtration) при максимальной нагрузке, определенной в тесте PERF_15.

Условия теста: тест проводится со всеми включенными модулями безопасности, 10 000 правилами МЭ и 1 000 правилами NAT.

Команда запуска:

```
/opt/trex/trex-perf/trex_run_v1.2.py -d 300 -f /home/toor/emix_jet.py
--ngfw-hostname "NGFW name_PERF_16" -a /home/toor/attacks_NGFW_name/ -m 140
```

Процесс теста:

- Используя генератор трафика, запустить нагрузку, равную 100% определенной в тесте PERF_15:
  - Профиль трафика emix_jet.py.
- Параллельно направить через NGFW/UTM трафик pcap-атак, определенных в процессе настройки NGFW/UTM.

PERF_17. Максимальное число транзакций в секунду при инспекции TLS

Задача: определить максимальное число транзакций в секунду при инспекции TLS (SSL TPS).

Процесс теста:

- Настроить политику SSL Inspection для пользовательского трафика.
- Применить политики на устройство.
- Обратиться к тестовому HTTPS-сервису.
- Проверить издателя сертификата.
- Запустить генератор трафика:
  - Профиль трафика в приложении 2.
- В статистике генератора зафиксировать:
  - Текущее значение TPS;
  - Число потерь.
- Используя бинарный поиск, выполнить найти значение TPS, при котором потери близки к 1%.
- Рекомендуемое время прохождения каждой итерации теста — 5 минут.

Профиль EMIX

Трафик	% by bytes	% by packets	% by conn	Avg pkt size, B
NFS (TCP)	0,1	0,5	1	199
HTTP (TCP)	33,44	25,2	19,94	1370
HTTPS (TLS1.2)	35,08	23,83	15,95	1520
HTTP (POST)	0,77	0,88	1,99	902
VIDEO_CALL	20,09	19,03	1	1090
RTP (UDP)	0,84	11,52	1	75,6
SMB (TCP)	1,45	1,59	5,38	941
DNS (UDP)	0,26	2,55	7,98	107
SMTP (TCP)	0,55	1,33	4,49	429
POP (TCP)	0,16	0,59	1	289
IMAP (TCP)	0,05	0,51	1	108
Generic TCP	2,02	2,54	9,97	823
Generic UDP	1,97	4,51	5,38	452
SIP (UDP)	2,84	4,42	19,94	664
RDP (UDP)	0,18	0,15	1,99	1270
SYSLOG (UDP)	0,09	0,47	1	203
SSH (TCP)	0,08	0,38	1	220

Профиль Yandex.Tank

```
overload:
    enabled: true
    package: yandextank.plugins.DataUploader
    token_file: "token.txt"
phantom:
    address: 10.31.47.201:443
    header_http: "1.1"
    headers:
        - "Host: nginx"
        - "[Connection: close]"
    uris:
        - /0kb.bin
    load_profile:
        load_type: rps
        schedule: line(1, 200000 , 300s)
    ssl: true
    client_cipher_suites: ECDHE-RSA-AES256-GCM-SHA384
    instances: 20000
autostop:
    autostop:
        - http(5xx,10%,5s)
console:
    enabled: true
telegraf:
    enabled: false
```