МЕТОДИКА НАГРУЗОЧНОГО ТЕСТИРОВАНИЯ РЕШЕНИЙ NGFW/UTM

JET SECURITY TEAM

ОБЩЕЕ ОПИСАНИЕ

Мы разработали данную методику 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 типовой организации. В рамках нагрузочного тестирования NGFW/UTM мы рассмотрели сценарий, когда устройство используется как периметровый межсетевой экран с включением доступных модулей безопасности. В ходе работ мы использовали типичные настройки для организации с количеством сотрудников от 500 до 1000 человек, наличием 1-2 филиалов и использованием таких корпоративных приложений, как почта, CRM, аудио- и видеоконференцсвязь, корпоративный портал.

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%.

Значения, которые мы получили, не могут быть единственно верными, так как на результат влияет множество факторов:

- методика тестирования: производители NGFW используют собственные подходы к проведению нагрузочного тестирования, которые отличаются от текущего;
- профиль трафика: в каждой организации он свой;
- настройки NGFW/UTM: включенные модули, используемое количество сигнатур IPS и т. п.;
- настройки правил МСЭ: количество правил, широта правил (количество трафика, проверяемое каждым правилом, особенно актуально для SSL Inspection, AppControl и т. п.).

Мы полагаем, что полученные нами значения ближе к реальной производительности, чем полученные на синтетическом трафике. Однако для каждой организации они будут отличаться из-за различных профилей трафика и настроек устройства.

ОПИСАНИЕ СТЕНДА

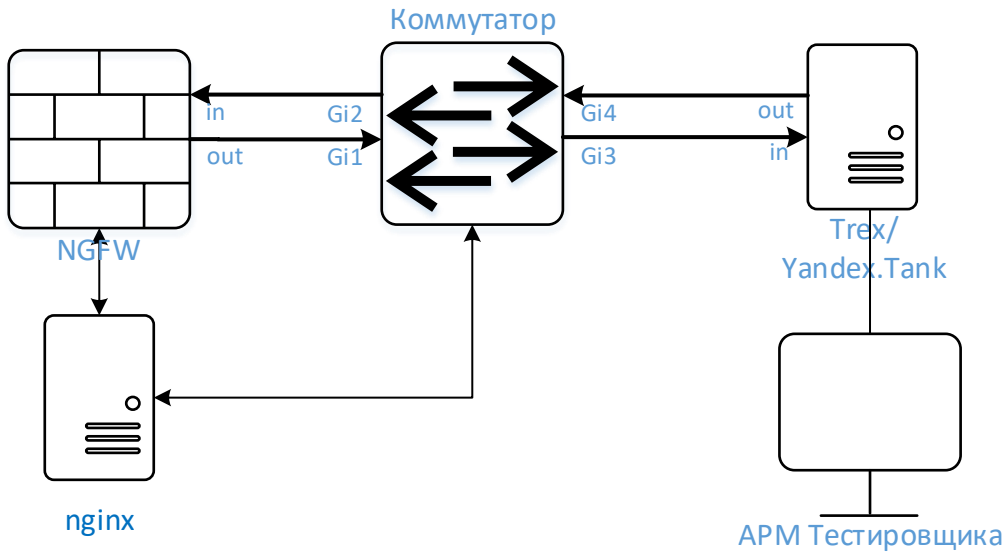

Рис. 1. Принципиальная схема тестового стенда

Схема тестового стенда приведена на рисунке 1. Характеристики сервера и используемое ПО приведены в **Таблица 1.** , IP-адресация в **Таблица 2**.

Таблица 1.

Характеристики сервера и используемое 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 v3.04 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	10.31.122.228/24	ВМ с сервисом HTTPS для определения SSL TPS

НАСТРОЙКА ТЕСТОВОГО СТЕНДА

Установка TRex основывается на конфигурациях и инструкциях, приведенных на https://github.com/OlegKashtanov/ngfw-trex-perf. Для проведения атак во время нагрузочных сценариев были подготовлены и записаны с периметровой инфраструктуры «Инфосистемы Джет» более 80 различных вариантов атак в виде pcap-файлов. Такое разнообразие необходимо, чтобы сгенерировать блокировки у всех тестируемых вендоров. 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 Приложение 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/snippets/ssl-params.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 испытаний проводится 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 лимитирующим фактором должна быть 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. Создать 200 правил. Правила создаются так, чтобы весь трафик обрабатывался последним правилом.

№	ИМЯ	ИСТОЧНИК	НАЗНАЧЕНИЕ	СЕРВИС	ДЕЙСТВИЕ	СОВ	LOGS
1	Test_1	10.10.0.1/24	10.10.1.1/24	Any	Отбросить	Выкл.	Вкл.
…							
199	Test_199	10.10.0.199/24	10.10.1.199/24	Any	Отбросить	Выкл.	Вкл.
200	EndRule	16.0.0.0/8	Any	Any	Пропустить	Вкл.	Вкл.

2. IPS:
   2.1. Настроить использование рекомендуемого вендором 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. Логирование включить для всех правил.

ПЛАН ТЕСТИРОВАНИЯ

1. Настроить объект тестирования и подключить к Zabbix JWST.
2. Определить набор детектируемых pcap-атак каждым из проверяемых модулей (не менее трех на каждый проверяемый функционал).
3. Проверка ACL. Подать нагрузку для определения граничных значений с базовым профилем в течение 10 с при подборе «потолка» и 300 с после его определения для фиксации стабильности работы, записать последние значения:
   3.1. EMIX Throughput (описание профиля в Приложение 1);
   3.2. HTTP Concurrent connection;
   3.3. HTTP Connection per second;
   3.4. UDP Packet per second.
4. Проверка IPS:
   4.1. Подать EMIX Throughput в максимуме и детектируемый набор pcap-атак для IPS в течение 300 с. Критерии успешности: NGFW/UTM демонстрирует производительность, определенную на этапе 3, все pcap-атаки заблокированы.
5. Проверка антивируса:
   5.1. Подать EMIX Throughput в максимуме и детектируемый набор pcap-атак для антивируса в течение 300 с. Критерии: NGFW/UTM демонстрирует производительность, определенную на этапе 3, все pcap-атаки заблокированы.
6. Проверка AppControl:
   6.1. Подать EMIX Throughput в максимуме и pcap-файл с RDP-трафиком в течение 300 с. Критерии: NGFW/UTM демонстрирует производительность, определенную на этапе 3, все попытки использования RDP-приложения были заблокированы.
7. Проверка контентной фильтрации:
   7.1. Подать EMIX Throughput в максимуме и pcap-файл с целевым трафиком в течение 300 с. Критерии: NGFW/UTM демонстрирует производительность, определенную на этапе 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 Приложение 2).
   8.2. После определения SSL TPS записать последние значения.

ТЕСТ-КЕЙСЫ

PERF_1. Максимальное число новых соединений в секунду

Задача: определить максимальное значение Connections Per Second (CPS) — скорости открытия новых HTTP-сессий каждую секунду, максимальные потери до 1%.

Процесс теста

- Запустить генератор трафика (TCP, HTTP, 1 KB response):
  - Профиль трафика http_max_cps.py.
- В статистике генератора зафиксировать:
  - текущее значение CPS;
  - число потерь.
- Используя бинарный поиск, найти значение CPS, при котором потери трафика близки к 1%.
- Рекомендуемое время прохождения каждой итерации теста — 5 минут.

PERF_2. Максимальное число конкурентных соединений в секунду

Задача: определить максимальное значение Concurrent Connections (CC) — количества открытых HTTP-сессий без их закрытия.

Процесс теста

- Запустить генератор трафика:
  - Профиль трафика max_cc.py.
- В статистике генератора зафиксировать:
  - текущее значение CC;
  - число потерь.
- Используя бинарный поиск, найти значение CC, при котором потери сессий близки к 1%.
- Рекомендуемое время прохождения каждой итерации теста — 5 минут.

PERF_3. Максимальное число пакетов в секунду

Задача: определить максимальное количество пакетов, передаваемых за 1 секунду.

Процесс теста

- Запустить генератор трафика (UDP PPS):
  - Профиль трафика udp_pps.py.
- В статистике генератора зафиксировать:
  - текущее значение pps;
  - число потерь.
- Используя бинарный поиск, найти значение PPS, при котором потери близки к 1%.
- Рекомендуемое время прохождения каждой итерации теста — 5 минут.

PERF_4. Определение максимальной пропускной способности (EMIX)

Задача: определить максимальную пропускную способность для трафика EMIX.

Процесс теста

- Запустить генератор трафика:
  - Профиль трафика emix.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%.
- Рекомендуемое время прохождения каждой итерации теста — 5 минут.

PERF_5. Максимальное число транзакций в секунду при инспекции TLS

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

Процесс теста

- Настроить политику инспекции (декрипта) для пользовательского трафика.
- Применить политики на устройство.
- Обратиться к тестовому HTTPS-сервису.
- Проверить издателя сертификата.
- Запустить генератор трафика:
  - Профиль трафика в Приложение 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.

PERF_6. Работоспособность функций безопасности под нагрузкой

Задача: проверить работоспособность функций безопасности (IPS, AV, AppControl, URL Filtration) при максимальной нагрузке, определенной в тесте PERF_4.

Процесс теста

- Используя генератор трафика, запустить нагрузку, равную 100% определенной в тесте PERF_4:
  - Профиль трафика EMIX.
- Оставить межсетевой экран под заданной нагрузкой на 5 минут.
- Спустя указанное время параллельно направить через NGFW/UTM трафик pcap-атак, определенных в процессе настройки NGFW/UTM.

ПРОФИЛЬ EMIX

ТРАФИК	% BY BYTES	% BY PACKETS	% BY CONN
NFS (TCP)	0,1	0,5	1
HTTP (TCP)	33,44	25,2	19,94
HTTPS (TLS1.2)	35,08	23,83	15,95
HTTP (POST)	0,77	0,88	1,99
VIDEO_CALL	20,09	19,03	1
RTP (UDP)	0,84	11,52	1
SMB (TCP)	1,45	1,59	5,38
DNS (UDP)	0,26	2,55	7,98
SMTP (TCP)	0,55	1,33	4,49
POP (TCP)	0,16	0,59	1
IMAP (TCP)	0,05	0,51	1
Generic TCP	2,02	2,54	9,97
Generic UDP	1,97	4,51	5,38
SIP (UDP)	2,84	4,42	19,94
RDP (UDP)	0,18	0,15	1,99
SYSLOG (UDP)	0,09	0,47	1
SSH (TCP)	0,08	0,38	1

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[1], 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
```

[1] Значение подбирается в процессе тестов