MITIGATOR
DDOS PROTECTION SYSTEM

Руководство по установке программного комплекса «BIFIT MITIGATOR» v23.06.7 на ОС Альт Сервер 10.1

105203, г. Москва, ул. Нижняя Первомайская, 46
Техническая поддержка: support@mitigator.ru
Отдел продаж: salesteam@mitigator.ru
MITIGATOR.RU

СОДЕРЖАНИЕ

Системные требования

- Процессор с поддержкой hugepages и SSE 4.2 (не менее четырех ядер).
- Не менее 8 ГБ оперативной памяти.
- Сетевой адаптер, поддерживаемый DPDK.
- Отдельный сетевой интерфейс для управления.
- От 32 ГБ свободного места на диске. Утилизация диска зависит от количества политик защиты, времени хранения метрик, активности использования и т.д.

Поддерживаются как физические, так и виртуальные машины (KVM, VMWare).

Адаптеры Mellanox имеют особые требования к дистрибутивам.

Для установки необходимо обратиться за доступом к Docker-образам продукта.

Использование сети

Интерфейсы управления (mgmt) находятся под управлением Linux и используются системой MITIGATOR для следующего:

- доступ к web-интерфейсу и API;
- взаимодействие по BGP;
- отправка уведомлений;
- взаимодействие с внешними серверами.

Интерфейсы данных, через которые проходит очищаемый трафик, находятся под управлением MITIGATOR и недоступны ОС. Таким образом, трафик управления и очищаемый трафик физически разделены и один не влияет на другой.

Способы установки

- **Ansible**

  Задайте параметры, и популярная система оркестрации установит MITIGATOR. Для пользователей Ansible и тех, кто не хочет администрировать Linux.

- **Вручную**

  Рекомендуется для лучшего понимания системы. Требуется знание Linux и навыки работы с командной строкой.

Установка с помощью Ansible

Предполагается, что все команды выполняются от суперпользователя:

```
$ su -
```

Установить Ansible

```
# apt-get install ansible tar wget
```

Установить Docker и Docker Compose

Установить Docker из репозиториев дистрибутива:

```
# apt-get install docker-engine
```

Установить Docker Compose из официального репозитория и сделать файл исполняемым:

```
# curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname
-m)" -o /usr/bin/docker-compose
# chmod +x /usr/bin/docker-compose
```

3. Скачать и распаковать необходимые файлы:

```
# wget https://docs.mitigator.ru/v23.06/ansible/mitigator.tar -O- | tar -x
# wget https://docs.mitigator.ru/v23.06/ansible/config.yml -O mitigator/config.yml
```

Отредактировать mitigator/config.yml

- mitigator_arch: микроархитектура процессора, для которой будет загружена оптимизированная сборка обработчика пакетов: nehalem или haswell (указания по выбору).
- mitigator_nic_driver: драйвер сетевой карты для DPDK (подробнее о выборе).
- Настройки обработчика пакетов:
  - mitigator_nics: сетевые порты с указанием PCI-адресов и ядер процессора. Подразумевается, что они перечислены в порядке ext0, int0, ext1, int1 и т.д. Портов может быть нечетное количество.
- Настройки больших страниц:
  - mitigator_hugepage_size: размер страницы (2M или 1G);
  - mitigator_hugepage_nr: количество страниц.
- mitigator_nr_policies: максимальное количество политик защиты.
- **Обязательно** указать версию MITIGATOR (v23.06.7).

- При первом запуске и при mitigator_pull_images: y будет выполнена загрузка образов MITIGATOR, для чего нужно задать логин и пароль: mitigator_registry_user и mitigator_registry_pass.
- Можно настроить прокси для Docker´а и компонент MITIGATOR.

Развернуть MITIGATOR

Базовая команда:

```
# ansible-playbook -i mitigator.local, mitigator/mitigator.yml
```

Запятая после имени сервера — не опечатка.

- Если playbook выполняется на той же машине, куда устанавливается MITIGATOR, добавьте --connection=local (буквально).
- Иначе, если MITIGATOR устанавливается на удаленном сервере:
  - Если имя пользователя для подключения отличается от текущего, добавьте --user=login (где login — имя пользователя).
  - Если для подключения нужен пароль, установите sshpass и добавьте --ask-pass (будет запрошен пароль).
  - Если подключение происходит не от root, добавьте --become и --ask-become-pass (будет запрошен пароль для sudo или su).

Playbook безопасно исполнять повторно в случае проблем.

В конце установки машина перезагрузится. При выполнении playbook на той же машине, куда устанавливается MITIGATOR, необходимо самостоятельно проконтролировать успешность запуска systemd-сервиса mitigator.

После установки и запуска настроить систему для её стабильной и безопасной работы.

Ручная установка

Предполагается, что все команды выполняются от суперпользователя:

```
$ su -
```

Подготовка системы

Оборудование

Платформа

Поддерживается работа на платформах AMD и Intel с архитектурой x86-64. Для оптимальной работы платформ AMD требуется настройка BIOS.

Hyper-threading

Рекомендуется включить hyper-threading (HT) в BIOS.
При включенном HT следующая команда показывает 2:

```
# lscpu | grep 'Thread(s) per core:'
```

NUMA

Многопроцессорные платформы рекомендуется использовать в NUMA-режиме с одним процессором на NUMA-ноду. Поддерживаются платформы с одной и двумя NUMA-нодами.

Для оптимальной производительности рекомендуется разносить сетевые карты по разным NUMA-нодам, чтобы каждый процессор работал только с портами на своей ноде.

Узнать NUMA-ноду устройства по его PCI-адресу:

```
# cat /sys/bus/pci/devices/0000:04:00.0/numa_node
```

Драйвера (модули ядра)

Для сетевых карт Intel и некоторых других в системе должен быть загружен драйвер (модуль ядра), позволяющий DPDK работать с ними. Для карт Mellanox загрузка модулей ядра не требуется.

Необходимо:

1. Подобрать драйвер для установленных сетевых устройств (документация DPDK):
    - Карты Mellanox работают только с собственным драйвером.

- vfio-pci: стандартный, рекомендуется по умолчанию (см. ниже);
- uio_pci_generic: стандартный, используется вместо vfio-pci, если он не работает (см. ниже);
- virtio-pci: нужен для виртуальных адаптеров QEMU/KVM.

2. Настроить загрузку драйвера при запуске системы.

Управление драйверами

Привязка устройств к нужному драйверу производится через скрипт dpdk-devbind (документация DPDK).

Скачать и установить dpdk-devbind:

```
# wget https://docs.mitigator.ru/v23.06/dist/dpdk-devbind -O /usr/local/bin/dpdk-devbind
# chmod +x /usr/local/bin/dpdk-devbind
```

Посмотреть состояние сетевых устройств и доступных драйверов:

```
# dpdk-devbind --status-dev=net
```

В списке Network devices using DPDK-compatible driver перечислены устройства, привязанные к совместимому с DPDK драйверу. В списке Network devices using kernel driver — устройства со стандартными драйверами ядра. Текущий драйвер указан в поле drv=. Доступные драйвера перечислены в поле unused=. Отображаются только драйвера, загруженные в системе.

Если нужный драйвер не загружен, загрузить через modprobe. Например, для загрузки драйвера vfio-pci:

```
# modprobe vfio-pci
```

Сопоставить имя устройства (например, eth0 или enp3s0f0) и его PCI-адрес можно через поле if= в выводе dpdk-devbind.

Рекомендуется привязывать все устройства одной командной dpdk-devbind. Для работы с vfio-pci все порты одной карты должны быть привязаны к одному драйверу. Например, для привязки устройств 06:00.0 и 06:00.1 к драйверу vfio-pci:

```
# dpdk-devbind -b vfio-pci 06:00.0 06:00.1
```

Если команда привязки выполнилась без ошибок, драйвер можно использовать. Устройство, привязанное к специальному драйверу, пропадает из системы (вывод ip link и т.п.). Привязка работает до перезагрузки, автоматическую привязку можно настроить при установке MITIGATOR.

Устройства, отмеченные в списке **Active**, имеют IP-адрес. Обычно это порт, через который идет доступ к машине по SSH, поэтому скрипт не позволяет менять драйвер таким устройствам.

Автоматическая загрузка модуля ядра

Если нужный модуль ядра не загружается по умолчанию при старте системы, его загрузку можно включить через /etc/modules-load.d. Например, для загрузки vfio-pci:

```
# echo vfio-pci >> /etc/modules-load.d/mitigator.conf
```

Модуль vfio-pci

Модуль vfio-pci входит в состав ядра Linux. Необходим для работы DPDK с сетевыми картами. Требует поддержки процессором виртуализации ввода-вывода (Intel VT-d или AMD-V). Включается в BIOS соответствующими настройками (например, “VT-d”, “Intel Virtualization Technology”, “SVM”, “AMD-V”, “AMD Virtualization”, “IOMMU”).

В параметрах ядра необходимо добавить опции (например, через GRUB):

- Для платформ Intel: intel_iommu=on iommu=pt,
- Для платформ AMD: iommu=pt.

Проверить наличие виртуализации:

```
# grep -q 'vmx\|svm' /proc/cpuinfo && echo enabled || echo disabled
```

Проверить наличие IOMMU:

```
# compgen -G '/sys/kernel/iommu_groups/*/devices/*' > /dev/null && echo enabled || echo disabled
```

Загрузка модуля:

```
# modprobe vfio-pci
```

Модуль uio_pci_generic

Модуль uio_pci_generic входит в состав ядра Linux. Используется вместо vfio-pci, если тот не поддерживается системой или по какой-то причине не работает.

Загрузка модуля:

```
# modprobe uio_pci_generic
```

Hugepages

Для работы MITIGATOR необходимы настроенные hugepages (большие страницы памяти). Платформой могут поддерживаться hugepages разных размеров (2 МБ, 1 ГБ), рекомендуется настраивать страницы большего размера.

Необходимое количество hugepages зависит желаемого количества политик защиты. Рекомендуется отводить на hugepages 50-75% от общего объема памяти.

1 GB hugepages

Рекомендуется использовать hugepages размером 1 ГБ, если поддерживаются платформой. Их можно выделить только при загрузке системы.

Проверить поддержку:

```
# grep -q pdpe1gb /proc/cpuinfo && echo "supported" || echo "not supported"
```

Настраивается опциями в параметрах ядра. Пример для выделения 64-х 1 ГБ страниц:

```
default_hugepagesz=1G hugepagesz=1G hugepages=64
```

2 MB hugepages

Hugepages размером 2 МБ можно настраивать без перезагрузки системы.

Пример выделения 2048 страниц по 2 МБ:

```
# sysctl -w vm.nr_hugepages=2048
```

Пример настройки выделения при загрузке системы:

```
# echo 'vm.nr_hugepages = 2048' > /etc/sysctl.d/hugepages.conf
```

Docker

Установить Docker из репозиториев дистрибутива:

```
# apt-get install docker-engine
```

Установить Docker Compose из официального репозитория и сделать файл исполняемым:

```
# curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname
-m)" -o /usr/bin/docker-compose
# chmod +x /usr/bin/docker-compose
```

Установка MITIGATOR

Предполагается размещать все файлы в рабочем каталоге /srv/mitigator:

```
# mkdir -p /srv/mitigator
# cd /srv/mitigator
```

Docker Compose

1. Поместить базовую конфигурацию Docker Compose в рабочий каталог:

```
# wget https://docs.mitigator.ru/v23.06/dist/docker-compose.yml
```

Для сетевых карт Mellanox нужно использовать другую базовую конфигурацию:

```
# wget https://docs.mitigator.ru/v23.06/dist/docker-compose.hostmode.yml -O docker-compose.yml
```

2. Скачать базовый файл переменных и сохранить его под именем .env:

```
# wget https://docs.mitigator.ru/v23.06/dist/env -O /srv/mitigator/.env
```

3. В файле .env задать:
   - Версию системы (VERSION, **v23.06.7**).
   - Микроархитектуру процессора из списка указанных в файле-примере (ARCH).
   - Максимальное количество политик защиты IPv4 (DATA_PLANE_NR_POLICIES).
   - Максимальное количество политик защиты IPv6 (DATA_PLANE_NR_POLICIES_IPV6).
   - Имя экземпляра в кластере (MITIGATOR_OWN_NAME, **обязательно**).
   - Внешний адрес MITIGATOR (MITIGATOR_HOST_ADDRESS, **обязательно**).
   - Прокси для сервера лицензий (ls.mitigator.ru), почтовых уведомлений и службы «Весточка».
   - Часовой пояс (TZ).
   - Токен взаимодействия бэкенда и подсистемы детектирования (TOKEN). В файле .env задано значение TOKEN по умолчанию. **Обязательно** смените его.

Подробно эти настройки описаны внутри файла-примера.

Обработчик пакетов

Архитектура процессора

Для максимальной производительности MITIGATOR нужно использовать сборку, оптимизированную под архитектуру и набор инструкций процессора целевой машины.

В файле .env должна быть строка вида:

```
ARCH=haswell
```

Доступные варианты:

- nehalem - CPU с поддержкой SSE4.2,
- haswell - современные CPU с поддержкой AVX2.

Свой процессор можно найти в каталоге Intel, микроархитектура указана в строке *Code Name*.

Параметры обработки

Необходимо создать файл data-plane.conf, описывающий параметры запуска обработчика пакетов:

```
# touch data-plane.conf
```

Файл настроек по умолчанию пустой. Править его требуется только если нужно указать настройки, отличные от выбранных автоматически. Описание настроек.

Выбор портов

Порты в приложении называются ext0, int0, ext1, int1 и т.д. ext — порты внешней сети, int — порты внутренней (защищаемой) сети. Объединяются в ext-int пары по индексу в названии. ext-int пары портов используются для маршрутизации трафика в схеме включения «inline». В схеме включения «on-a-stick» ext-int пары не используются и могут быть любыми.

Если порты не заданы в настройках, используются все порты в системе, доступные DPDK. В таком случае, порты перечисляются в порядке возрастания их PCI-адресов. ext-int пары портов формируются только для портов с общей NUMA-ноды.

Если порядок перечисления портов по умолчанию не совпадает с физическим подключением линков, либо требуется ограничить список используемых портов, порты можно настроить явно:

```
ext0: 04:00.1
int0: 04:00.0
ext1: 84:00.1
int1: 84:00.0
```

Привязка драйверов к сетевым портам

Перед запуском MITIGATOR отведенные ему сетевые порты должны быть под управлением драйвера, выбранного при подготовке системы.

Для систем под управлением systemd нужно выполнить привязку перед запуском службы MITIGATOR (см. следующий пункт).

- Загрузить скрипт привязки и сделать его исполняемым:

```
# wget https://docs.mitigator.ru/v23.06/dist/dpdk-devbind -O /usr/local/bin/dpdk-devbind
# chmod +x /usr/local/bin/dpdk-devbind
```

- Создать каталог /etc/systemd/system/mitigator.service.d:

```
# mkdir -p /etc/systemd/system/mitigator.service.d
```

- В нем разместить файл nics.conf такого вида:

```
[Service]
ExecStartPre=/usr/local/bin/dpdk-devbind -b vfio-pci 04:00.0 04:00.1 84:00.0 84:00.1
```

- **Драйвер и PCI-адреса заменить на необходимые.**

Загрузка образов и запуск

MITIGATOR запускается командой docker-compose up -d.

Для систем под управлением systemd нужно настроить готовую службу:

- Разместить файл службы MITIGATOR:

```
# wget https://docs.mitigator.ru/v23.06/dist/mitigator.service  -O /etc/systemd/system/mitigator.service
```

- Настроить автозапуск MITIGATOR:

```
# systemctl enable mitigator
```

- При первом запуске или обновлении нужно совершить вход в хранилище образов со своими учетными данными:

```
# docker login docker.mitigator.ru
```

- Запустить MITIGATOR:

```
# systemctl start mitigator
```

- При первом запуске понадобится некоторое время для загрузки образов. Процесс можно наблюдать в выводе docker-compose logs -f или, для systemd:

```
# journalctl -u mitigator -f
```

- Спустя некоторое время, web-интерфейс MITIGATOR будет доступен по адресу интерфейса управления.

После установки и запуска настроить систему для её стабильной и безопасной работы