SkyDNS on-prem

SkyDNS on-prem решение

1. Общее описание продукта

SkyDNS on-prem решение для контентной фильтрации DNS-трафика представляет собой DNS Proxy, который получая DNS-запрос идентифицирует пользователя, сравнивает целевой домен с политикой фильтрации для этого клиента и принимает решение о блокировке (или не блокировке) трафика.

Блокировка, с технической точки зрения, осуществляется путем подмены IP целевого ресурса на IP страницы блокировки. Это может быть как собственная корпоративная страница, расположенная за пределами нашего решения, так и встроенная в наше решение типовая страница блокировки (при этом её можно кастомизировать).

Важным ограничением является необходимость добавление в список доверенных нашего корневого сертификата на каждое конечное устройство пользователя для отображения страницы блокировки по протоколу https. Для отображения страницы блокировки по протоколу http это не требуется. Без установленного сертификата на конечном устройстве при блокировании домена по https, пользователь не увидит страницу блокировки, но доступ к ресурсу всё равно будет заблокирован.

В случае, если блокировка не требуется, работая как DNS Proxy, запрос на резолв целевого домена перенаправляется на следующий в цепочке кэширующий DNS-сервер. Это может быть как локальный корпоративный DNS-сервер или DNS провайдера, так и любой публичный DNS.

Кроме этого, пропуская через себя весь DNS-трафик это решение позволяет анализировать весь трафик в разрезе пользователей. Аккумулируются логи всех запросов и предоставляется доступ к статистике.

2. Варианты схемы размещения продукта в контуре компании

В зависимости от топологии сети клиента, схем размещения может быть множество. Ниже представлены примеры типовых решений.

Для провайдеров

image-1724682719596.png

Image not found or type unknown

Такая схема размещения используется в крупных сетях, где конечные пользователи расположены за NAT относительно основного оборудования клиента и нет возможности идентифицировать их по индивидуальному IP.

Для корпоративных клиентов

image-1724683252591.png

Image not found or type unknown

или

image-1724682921607.png

Image not found or type unknown

Эта схема используется в сетях, где в точке расположения нашего решения возможна идентификация конечного пользователя по IP. Здесь в зависимости от наличия в компании собственного кэширующего DNS сервера - запросы проксируются либо на него, либо на внешнюю инфраструктуру (DNS провайдера или любой публичный DNS, типа 1.1.1.1 или 8.8.8.8).

Для корпоративного клиента с AD (или другим локальным DNS)

image-1724683044371.png

Image not found or type unknown

В этом случае запросы получает сначала сервер AD, обслуживает локальные адреса, затем, в качестве внешнего источника домен передается в наш сервис, где происходит фильтрация определенных доменов и только в случае необходимости обеспечить доступ к целевому ресурсу - запрос передается на внешний рекурсивный DNS. В такой схеме - невозможно идентифицировать конечного пользователя, т.к. источником всех запросов будет служить домен-контроллер или ддругой локальный DNS-сервер.

Однако, прямо сейчас SkyDNS разрабатывает решение, которое позволит устранить эту проблему и возможно будет сохранить связь конечных пользователей с домен-контроллером, при этом идентифицировать их на нашем оборудовании. Кроме того, станет возможным управлять фильтрацией через групповые политики AD. Это решение должно выйти до конца 2024 года.

3. Идентификация пользователей

Для возможности применения различных политик фильтрации, а также для разделения статистики по запросам необходимо идентифицировать конечного пользователя. Для идентификации в on-prem решении используется один из следующих вариантов:

- IP
- IP/подсеть
- IP:порт
- IP:[порт]-[порт]

В случае наличия у каждого пользователя своего IP (с которого будут поступать DNS-запросы) мы клиента идентифицируем по этому IP. Либо, если нет необходимости детального разделения - сразу по подсети.

Если же пользователь находится за NAT (CGNAT, NAT44 и др.) и запросы от разных конечных пользователей будут приходить с одного и того же IP, то в этом случае мы можем идентифицировать конечного пользователя по связке IP + порт, либо по диапазону портов.

4. Из чего состоит

image-1725514270806.png

Image not found or type unknown

Внутри наше решение состоит из:

- Модуля DNS Proxy, который получат DNS-запрос и возвращает на него ответ в виде IP
- Внутренней БД, содержащей информацию о всех пользователях и политиках фильтрации
- RestAPI для обновления данных в БД:
  - Создание/изменение профилей фильтрации (какие категории должны быть заблокированы)
  - Создание/изменение страницы блокировки
  - Создание/обновление пользователя или группы пользователей с указанием:
    - идентификатора (подсеть/IP/порт)
    - профиля фильтрации
    - страницы блокировки
- Страницы блокировки
- Модуля парсинга бинарных логов DNS Proxy (StatsLoader)
- Модуля экспорта статистики в СУБД Clickhouse

Расшифровка других элементов схемы:

Клиент
Представляет конечного пользователя, который будет использовать сервис DNS-фильтрации. Клиент отправляет DNS-запросы в SkyDNS on prem.

DB Files
Это база данных, которая хранит информацию о клиентах и правила фильтрации. RestAPI извлекает информацию из этой базы данных.

DNSPROXY
Этот компонент является центральным элементом процесса DNS-фильтрации. Он получает DNS-запросы от клиента, применяет правила фильтрации на основе данных, полученных из DB Files, а затем перенаправляет запросы на кэширующий DNS.

Кэширующий DNS сервер
Это кэш-сервер, развернутый на стороне клиента. Он выполняет фактическое разрешение DNS-запросов после их фильтрации через DNSPROXY.

HostPATH
После обработки DNS-запросов DNSPROXY создает binlog-файлы (журнальные файлы, содержащие информацию о DNS-запросах). HostPATH отвечает за хранение этих binlog-файлов.

StatsLoader
После сохранения binlog-файлов StatsLoader получает и анализирует двоичные данные от HostPATH, затем обрабатывает статистику и отправляет их в кластер ClickHouse для хранения и анализа.

Балансировщик нагрузки (LB)
StatsLoader отправляет обработанную статистику на балансировщик нагрузки, который распределяет её по узлам ClickHouse.

Кластер ClickHouse
Это распределенная база данных, используемая для хранения и анализа статистики DNS-запросов. Кластер разделен на шарды, и каждый шард содержит несколько узлов ClickHouse для отказоустойчивости (mirroring). Шард 1 и Шард 2 предназначены для высокой производительности, т.к. чтение и запись производится параллельно.

Zookeeper
Эти экземпляры управляют конфигурацией и координацией кластера ClickHouse, обеспечивая консистентность данных и надежность распределенной системы.

5. Настройка и взаимодействие с продуктом

Вся настройка, обслуживание и поддержка осуществляется исключительно специалистами SkyDNS, что позволяет однозначно определить зоны ответственности между SkyDNS и клиентом. Клиент предоставляет оборудование (сервер на архитектуре x64) согласно требованиям и удаленный доступ к этому серверу, после чего мы настраиваем всё "под ключ".

Нашими специалистами, совместно с заказчиком производится первичная настройка правил фильтрации и производится обучение специалистов клиента для дальнейшей настройки.

При необходимости - выделяется отдельная линия поддержки для клиентов с On-Prem решениями.

6. Работа со статистикой

На самом сервере SkyDNS Security хранятся бинарные логи. В составе нашего решения присутствует модуль, который парсит эти логи и выгружают во внешнюю СУБД для последующего анализа и формирования отчётов. В данный момент существует коннектор для выгрузки логов в СУБД ClickHouse, которая показывает наилучшую производительность при работе с таким типом данных, но при необходимости, логи можно выгружать и в другие СУБД. Модуль статистики не является обязательным для обеспечения основного функционала SkyDNS Security, однако, без неё невозможно оценить эффективность работы и проводить расследование инцидентов.

7. Системные требования к продукту

Минимальные системные требования для сервера под модуль SkyDNS Security составляют:

- CPU 4 ядра (архитектура x64)
- RAB 8 Gb
- 200 Gb NVMe
- ОС Debian 11

Таких характеристик достаточно для функционирования всех модулей системы и обработки до 1000 запросов в секунду.

На характеристиках 64 ядра CPU и 128 Gb RAM система способна обрабатывать до 2 млн. запросов в секунду при достаточной пропускной способности сетевых интерфейсов.

Для кластера СУБД Clickhouse минимальные системные требования на один сервер:

- CPU 8 ядер (архитектура x64)
- RAB 16 Gb
- 500 Gb NVMe (до 2 Tb в зависимости от объёма трафика клиента)

При этом мы рекомендуем минимальный кластер из 4 серверов с разбивкой 2x2. Т.е. два шарда для параллельной записи/чтения и 2 сервера в каждом шарде для резервирования.

Опционально, при небольшом объёме трафика можно назвернуть Stand-Alone сервер Clickhouse и отказаться от модуля Zookeeper.

8. Rest API

Всё взаимодействие со SkyDNS Security производится с помощью Rest API, который обычно отвечает на порту 8080 с того же IP, с которого обрабатываются запросы от пользователей. По умолчанию, API отвечает только на запросы с IP из белого списка, указанных при разворачивании сервера.

Пример работы

Для понимания того, как с ним работать, рассмотрим конкретный сценарий первичной настройки. Для этого нужно отправить следующий запрос:

```
{
    "profiles": [
        {
            "profile": {
                "id": 1,
                "page_id": 1
            },
            "cat_ids": [3, 4, 12]
        }
    ],
    "blockpages": [
```

```
        {
            "id": 1,
            "type": 0
        },
        {
            "id": 2,
            "type": 1
        }
    ],
    "bw_lists": [
       {
            "profile_id": 1,
            "type": "deny",
            "domains": [
                "example1.com",
                "example2.com",
                "example3.com"
            ]
        }
    ],
    "nets": [
        {
            "ip": "100.100.100.100",
            "profile_id": 1,
            "prefix_len": 32
        },
        {
            "ip": "100.110.110.0",
            "profile_id": 1,
            "prefix_len": 24
        },
        {
            "ip": "100.120.0.0",
            "profile_id": 1,
            "prefix_len": 16
        }
    ],
    "nets6": [],
    "napts": []
}
```

- тип: POST
- URL: <domain>/init/
- Данные:

В этом init-файле (по порядку) мы создаем в настройках SkyDNS On-prem:

- Единственный профиль фильтрации с id=1 и со страницей блокировки id=1 **[profiles → profile]**
- В настройках этого профиля указываем три блокируемые категории (в данном примере - malware, фишинг и ботнеты) **[profiles → cats_ids]**
- Создаем две страницы блокировки двух разных типов (type 0 - с указанием заблокированного домена, type 1 - без указания) [**blockpages**]
- В черный список для профиля фильтрации с id=1 добавляем три домена [**bw_lists**]
- К профилю фильтрации с id=1 добавляем IP-адреса, запросы с которых будут фильтроваться согласно политике этого профиля (в этом примере 1 конкретный IP и 2 подсети с 16-ой и 24-ой маской) [**nets**]
- [**nets6**] (то же самое, что и nets, но в IPv6) и [**napts**] (идентификация пользователя за NAT по порту) в данном примере мы не заполняем, но их обязательно нужно отправить (пустыми), чтобы инициировать создание первичной конфигурации.

После отправки такого запроса, в случае положительного ответа с кодом 204, первичная настройка завершена и любой DNS-трафик на этот сервер будет обрабатываться согласно указанным правилам.

Изменение категорий фильтрации

Теперь рассмотрим типовые кейсы изменения или обновления настроек фильтрации. Например, нам захотелось изменить категории, которые мы фильтруем. Для этого нужно обновить целиком список категорий для нужного профиля. Исходя из примера выше у нас был всего один профиль создан с двумя категориями. Допустим, нам нужно добавить ещё три категории (криптоджекинг, DGA и Ransomware). Для этого нужно отправить следующий запрос:

- тип: PATCH
- URL: <domain>/profiles/**1**/
- Данные:

```
{
    "cat_ids": [3, 4, 12, 66, 70, 71],
    "page_id": 1
}
```

Т.е. здесь мы указали id профиля прямо в url, а в теле запроса передали обновленный список категорий (прошлые 3 и новые 3), указав ту же страницу блокировки, что и была ранее.

Добавление адреса в белый список

Теперь представим ситуацию, что нам какой-то домен среди блокируемых нужно добавить в белый список, чтобы он не блокировался по правилу категории. Для этого отправляем следующий запрос:

- тип: POST
- URL: <domain>/profile/**1**/bw_list
- Данные:

```
{
  "type": "allow",
  "domain": "example4.com"
}
```

Таким образом, для профиля фильтрации с id=1 мы добавили домен в белый список и он будет иметь приоритет над блокировкой по категории

Добавление доменов в черный список

Допусти, теперь нам нужно часть доменов, которых нет в блокируемых категориях принудительно заблокировать. Для этого добавим их в черный список групповым методом:

- тип: POST
- URL: <domain>/profile/**1**/bw_list/batch
- Данные:

```
{
  "type": "deny",
  "domains": ["example5.com", "example6.com", "example7.com", "example8.com", "example9.com"]
}
```

Добавление новой сети (или IP) для фильтрации:

- тип: POST
- URL: <domain>/net/

- Данные:

```
{
    "ip": "100.100.100.101",
    "profile_id": 1,
    "prefix_len": 32
}
```

Таким образом, к нашему профилю фильтрации мы добавили один новый IP. В этом случае мы указали префикс сети 32. Если нам нужно указать было целиком подсеть, то мы могли отправить, например:

```
{
    "ip": "100.100.101.0",
    "profile_id": 1,
    "prefix_len": 24
}
```

Удаление IP из списка

Добавление новой сети (или IP) для фильтрации:

- тип: DELETE
- URL: <domain>/net/**1684300901**
- Данные: отсутствуют

В этом запросе отсутствует тело. Единственное, что нужно здесь передать - это IP в десятеричном формате в URL. Для этого нужно ввести именно тот IP, который мы ранее передавали (без маски). Например, 100.100.100.101 или 100.100.101.0 для предыдущих примеров. Чтобы получить IP в десятеричном формате можно воспользоваться, например, вот этим конвертером.

Изменение профиля фильтрации для конкретного IP

- тип: PATCH
- URL: <domain>/net/**1684301056**
- Данные:

```
{
    "ip": "100.100.101.0",
    "profile_id": 2,
    "prefix_len": 24
}
```

В данном случае мы обратились к записи 100.100.101.0 (в десятеричном формате в URL) и изменили профиль с id=1 на id=2. Для изменения префикса, например, с 24 на 16, нам нужно было бы удалить эту запись и добавить новую с соответствующим префиксом, чтобы выставить 100.100.0.0 вместо 100.100.101.0.

Создание нового профиля фильтрации

В примере выше мы поменяли профиль фильтрации для конкретного IP. Но у нас еще не был создан такой профиль. Создать его можно вот так:

- тип: POST
- URL: <domain>/profiles/
- Данные:

```
{
    "profile": {
        "id": 2,
        "page_id": 1
    },
    "cat_ids": [3, 4, 12, 66, 70, 71, 13]
}
```

Таким образом мы создали профиль фильтрации с id=2, указали для него тот же тип страницы блокировки и указали для него те же самые категории, что в первом профиле, добавив новую категорию - "для взрослых".

Полное описание методов API

- ***Инициализация базы юзердаты (создание пачки данных):***
  - тип: POST
  - URL: <domain>/init/
  - данные:

```
{
    'profiles': [
        {
            'cat_ids': [<cat_id>,],
            'profile': {
                'page_id': <page_id>,
                'id': <id>
            }
        }, ...
    ],
    'nets': [{'ip': <ipv4>, 'profile_id': <profile_id>, 'prefix_len':
<prefix_len4>}, ...],
    'nets6': [{'ip': <ipv6>, 'profile_id': <profile_id>, 'prefix_len':
<prefix_len6>}, ...],
    'blockpages': [{'id': <page_id>, 'type': <blockpage_type>}, ...],
    'napts': [
        {
            'ip': <ipv4>,
            "profile_id": <profile_id>,
            "lower_port_bound": <lower_port_bound>,
            "upper_port_bound": <upper_port_bound>
        },
    ]
}
```

- результат: 204, без тела

- ***Получение всех профилей:***
  - тип: GET
  - URL: <domain>/profiles/
  - результат: 200
- ***Добавление профиля:***
  - тип: POST
  - URL: <domain>/profiles/
  - данные:

```
{
    "profile": {
        "id": int,
        "page_id": int
    },
    "cat_ids": [int, ]
}
```

    - результат: 201
- ***Получение профиля:***
    - тип: GET
    - URL: <domain>/profiles/<id>/
    - результат: 200
- ***Добавление айпи адреса ipv4:***
    - тип: POST
    - URL: <domain>/net/
    - данные: ipv4, profile_id
    - результат: 201
- ***Получение ipv4 адреса:***
    - тип: GET
    - URL: <domain>/net/<int_ip>/
    - результат: 200
- ***Обновление ipv4 адреса:***
    - тип: PATCH
    - URL: <domain>/net/<int_ip>/
    - данные (необязательные): ipv4, profile_id
    - результат: 200
- ***Удаление ipv4 адреса:***
    - тип: DELETE
    - URL: <domain>/net/<int_ip>/
    - результат: 204 код, без тела
- ***Добавление айпи адреса ipv6:***
    - тип: POST
    - URL: <domain>/net6/
    - данные: ipv6, profile_id
- ***Получение ipv6 адреса:***
    - тип: GET
    - урл (предварительно): <domain>/net6/<int_ip>/
    - результат: словарь с данными ipv6, profile_id
- ***Обновление ipv6 адреса:***
    - тип: PATCH
    - URL: <domain>/net6/<int_ip>/
    - данные (необязательные): ipv6, profile_id
    - результат: 200
- ***Удаление ipv6 адреса:***
    - тип: DELETE
    - URL: <domain>/net6/<int_ip>/
    - результат: 204 код, без тела