

NICE.Сенсор Сетевых Атак

Раздел 1. Введение

1.1. Назначение продукта

NICE.Сенсор Сетевых Атак (далее — **NICE.SSA**) — это промышленная платформа сетевой видимости и обнаружения угроз, которая в Яндекс Cloud разворачивается «в один клик» и объединяет сбор трафика, анализ событий, расследование инцидентов и долговременное хранение данных в единую управляемую систему.

Что такое NICE.SSA

NICE.SSA — это открытое и объяснимое решение класса Network Detection & Response (NDR), созданное для облачных и гибридных инфраструктур. Платформа принимает зеркалируемый трафик, выполняет многослойную детекцию (сигнатурную, поведенческую, эвристическую), обогащает события техническим контекстом, индексирует их для поиска и визуализации, а также предоставляет инструменты оперативного реагирования и глубокого сетевого расследования.

Архитектурно NICE.SSA использует проверенные открытые компоненты: высокопроизводительный анализатор трафика **Suricata**, коллектор и конвейер доставки **Vector**, поисково-аналитический стек **OpenSearch** с готовыми шаблонами индексов и ILM-политиками, интерфейс расследования **EveBox**, полнотекстовый анализ PCAP-сессий с **Arkime**, управление правилами через **Scirius**. Все пользовательские интерфейсы скрыты за единой точкой входа **Nginx** с TLS и контролем доступа.

Результат — прозрачная сетево-ориентированная аналитика безопасности, адаптированная под операционные реалии российского рынка и экосистемы Яндекс Cloud: минимум ручных действий, понятная методология, воспроизводимые процедуры реагирования, детальная телеметрия для аудита и комплаенса.

Ключевые свойства

- Единая точка входа и русскоязычный UX
- «Один клик» в Яндекс Cloud Marketplace
- Открытые компоненты без «чёрных ящиков»
- Высокодостоверные (high-fidelity) алерты
- Guided threat hunting и готовые дашборды
- Глубокое расследование на уровне PCAP
- ILM/ретеншн, аудит, экспорт артефактов

Концепция Network Detection & Response (NDR)

NDR объединяет три ключевые доменные задачи: **непрерывный контроль трафика**, **обнаружение атак** и **оперативное реагирование**. В отличие от классического подхода «только IDS», NDR делает акцент на интерпретации и верификации сигналов, снижении шума и ускорении триажа. NICE.SSA формирует события высокого доверия, снабжая их контекстом (сессии, метаданные протоколов, временные корреляции, ссылки на PCAP), чтобы действия аналитика были быстрыми и обоснованными.

Непрерывный контроль

Зеркалирование VPC-трафика в Яндекс Cloud, захват метаданных и полезной нагрузки по протоколам L2–L7, стабильная телеметрия без вмешательства в приложения и хосты.

Обнаружение атак

Сигнатуры, эвристика и поведенческие индикаторы в Suricata; нормализация и доставка через Vector; индексирование и агрегации в OpenSearch; снижение шумности за счёт свёртки однотипных сработок в «декларации» высокой важности.

Реагирование

EveBox для триажа, Arkime для PCAP-расследования, готовые плейбуки, экспорт артефактов и отчётность для аудита. Все интерфейсы доступны через Nginx с TLS и разграничением прав.

Задача NICE.SSA — превратить «сырые» сетевые сигналы в проверяемые доказательства с понятным приоритетом, чтобы команда безопасности быстрее принимала решения и могла воспроизводимо доказывать свои выводы.

Зачем нужен NICE.SSA

NICE.SSA закрывает разрыв между сетевой телеметрией и практикой реагирования в облаке: даёт видимость, снижает шум, ускоряет расследования и поддерживает требования регуляторов. Благодаря открытым компонентам и русскоязычной документации решение технологически и юридически прозрачно, легко сопровождается и масштабируется.

Мониторинг безопасности

Непрерывная сетевое-центристская видимость сервисов, детекция команд-и-контроля, эксплуатаций уязвимостей и аномалий приложений, контроль политики доступа и сегментации.

Инцидент-респонс (IR)

Быстрый триаж с EveBox, реконструкция таймлайнов, анализ артефактов на уровне PCAP в Arkime, экспорт доказательств и автоматизация типовых шагов реагирования.

Аудит сетевой активности

Систематизированные индексы и дашборды в OpenSearch, повторяемые запросы и фильтры, отчёты по сегментам, сервисам, пользователям и типам коммуникаций.

Комплаенс

Сохраняемость и доказуемость: ILM-политики хранения, журналирование действий, экспорт артефактов, прозрачные цепочки принятия решений. Упор на локализацию и эксплуатационные практики в рамках российского правового поля.

Уникальность для российского рынка

Сочетание: нативное развёртывание в Яндекс Cloud, полностью русскоязычный опыт, единая точка входа, открытые компоненты без проприетарных «чёрных ящиков», готовые методики расследований и отчётности — формирует решение, которое не имеет прямых аналогов по глубине сетевой видимости и удобству эксплуатации в экосистеме Яндекс Cloud.

Пример: первый запуск и проверка сигналов

Проверяем готовность сервисов после развёртывания:

Проверка сервисов стека NICE.SSA

```
sudo systemctl status suricata.service
sudo systemctl status suricata-vector.service
sudo systemctl status suricata-opensearch-stack.service

# Быстрая диагностика Vector healthcheck
sudo systemctl status suricata-vector-healthcheck.service
journalctl -u suricata-vector-healthcheck.service --since "10 min ago"

# Проверяем, что Suricata пишет EVE и события попадают в OpenSearch
sudo tail -n 50 /var/log/suricata/eve.json
curl -s http://localhost:9200/_cat/indices | sort
```

Фрагмент типового источника/приёмника в конвейере Vector:

```
# /etc/vector/suricata.vector.yaml (фрагмент)
sources:
  eve_file:
    type: file
    include:
      - /var/log/suricata/eve.json
    read_from: end

transforms:
  normalize_eve:
    type: remap
    inputs: [eve_file]
    source: |
      .@timestamp = to_timestamp!(.timestamp)
      .sensor = "yc-nice-ssa"
      .stream = "suricata-eve"

sinks:
  opensearch_eve:
    type: elasticsearch
    inputs: [normalize_eve]
    endpoints: ["http://127.0.0.1:9200"]
    index: "suricata-events-%Y.%m.%d"
    mode: bulk
```

В базовую поставку входят шаблоны индексов и ILM-политики для OpenSearch, позволяющие управлять сроком хранения, горячими/тёплыми сегментами и бюджетом диска без потери важных артефактов расследования.

Мини-схема потока данных

VPC Mirroring (Yandex Cloud)

```

    □
    □
  [ Suricata ] — парсинг L7, сигнатуры, эвристика □ EVE (JSON)
    □
    □
  [ Vector ] — нормализация, буферизация, доставка
    □
    □
  [ OpenSearch ] — индексы, агрегации, ILM, поиск и дашборды
    □ □
    □ □□□ [ EveBox ] — триаж, заметки, подтверждение инцидента
    □ □□□ [ Arkime ] — PCAP-сессии, глубокое расследование
    □
    □
  [ Nginx ] — единая точка входа, TLS, доступ по ролям

```

Такая схема обеспечивает воспроизводимость расследований: от высокоуровневого алерта до побайтного анализа пакетов, с сохранением артефактов и связей для отчётности и комплаенса.

1.2. Целевая аудитория

NICE.Сенсор Сетевых Атак (NICE.SSA) создан для специалистов, отвечающих за безопасность, эксплуатацию и мониторинг облачных инфраструктур. Решение ориентировано на практические сценарии, где важны прозрачность, воспроизводимость действий и возможность немедленного реагирования без привлечения сторонних инструментов.

Администраторы инфраструктуры

Для системных администраторов NICE.SSA становится визуальной панелью наблюдения за сетевой активностью облачных сервисов. Решение помогает видеть, какие узлы, приложения и контейнеры взаимодействуют друг с другом, выявлять подозрительные сессии, нарушения политик маршрутизации и доступа, а также получать отчёты по нагрузке и аномалиям трафика.

DevSecOps-инженеры

Для DevSecOps NICE.SSA — это инструмент интеграции сетевой безопасности в жизненный цикл приложений. Он позволяет получать сетевые метрики и инциденты через API или OpenSearch, встраивать данные об угрозах в CI/CD-конвейеры, а также проверять безопасность микросервисов и контейнерных сред в Яндекс Cloud без установки агентов.

SOC-аналитики и IR-группы

Для аналитиков Центров мониторинга (SOC) и специалистов по реагированию NICE.SSA предоставляет обогащённые события высокой достоверности, контекстные дашборды, встроенные сценарии Guided Threat Hunting и интеграцию с Arkime для PCAP-расследований. Всё это ускоряет триаж, снижает ложные срабатывания и обеспечивает воспроизводимость расследований.

Компании и организации

NICE.SSA идеально подходит компаниям, размещающим свои сервисы и инфраструктуру в Яндекс Cloud: от малого бизнеса до государственных и корпоративных заказчиков. Решение позволяет внедрить сетевой контроль и аудит без развёртывания сложных SIEM-платформ, сохранив при этом независимость, прозрачность и контроль над собственными данными.

NICE.SSA — это инструмент, который объединяет специалистов разных ролей вокруг единого источника сетевой правды: системные администраторы видят производственные потоки, DevSecOps получают телеметрию и API, а SOC-аналитики — доказательную базу для расследований.

Интеграции

Рекомендуемая связка: NICE.SSA + OpenVAS (Greenbone)

Для получения полной и доказуемой картины рисков в Яндекс Cloud мы рекомендуем использовать NICE.Сенсор Сетевых Атак (NICE.SSA) совместно с образами уязвимостного сканера Greenbone Vulnerability Management (OpenVAS) из Yandex Cloud Marketplace, подготовленными специалистами NiceSoft.

Виртуальная машина [OpenVAS \(Greenbone Vulnerability Management\) для NiceOS](#) представляет собой специализированную сборку GVM для российского рынка: с полностью русифицированным веб-интерфейсом, преднастроенными сервисами и возможностью быстрого развёртывания из Yandex Cloud Marketplace.

:contentReference[oaicite:0]{index=0}

Логика совместного использования проста: **NICE.SSA отвечает за сетевую**

телеметрию и обнаружение активных атак, а OpenVAS — за поиск уязвимостей в хостах, сервисах и приложениях. Вместе они формируют связку *Network Detection & Response + Vulnerability Management*, которая позволяет видеть как факты эксплуатации, так и первопричины в виде конкретных технических дыр.

Как использовать в связке

- **1. NICE.SSA как «радар»:** сенсор получает зеркалируемый VPC-трафик в Яндекс Cloud, детектирует сканирования, эксплуатации, C2-каналы и аномалии протоколов, а затем сохраняет события в OpenSearch с контекстом (IP, порты, протоколы, сессии).
- **2. OpenVAS как «рентген»:** VM OpenVAS, развёрнутая из Marketplace, подключается к тем же сетям или выделенному сканирующему сегменту и по расписанию выполняет проверки хостов и сервисов на CVE, используя регулярно обновляемые базы NVT/SCAP/CERT/Notus. :contentReference[oaicite:1]{index=1}
- **3. Корреляция результатов:** IP-адреса, домены и хостнеймы из отчётов OpenVAS сопоставляются с событиями NICE.SSA. В результате становится видно, где существует не только уязвимость, но уже зафиксирована подозрительная активность или прямые попытки эксплуатации.
- **4. Приоритизация реагирования:** инциденты, в которых атакуется реально уязвимый сервис, попадают в верхнюю часть очереди IR-команды, а обнаруженные, но пока не эксплуатируемые уязвимости переходят в план планомерного устранения.
- **5. Поддержка аудита и комплаенса:** NICE.SSA даёт сетевые журналы, PCAP-артефакты и хронологию событий, тогда как OpenVAS формирует структурированные отчёты об уязвимостях и статусе их исправления. Этот набор артефактов удобно использовать для внутренних и внешних проверок, а также для соответствия стандартам и отраслевым требованиям. :contentReference[oaicite:2]{index=2}

Почему именно такая комбинация

- Разделение ролей: NICE.SSA показывает, что происходит в сети сейчас, фиксируя реальные попытки атак, а OpenVAS отвечает за что может случиться, выявляя потенциальные точки входа. Такое разделение снижает слепые зоны в модели угроз и помогает выстраивать реалистичную приоритизацию.
- Открытый стек и локализация: оба решения основываются на открытом ПО;

образ OpenVAS от NiceSoft адаптирован для российского рынка, локализован и оптимизирован под NiceOS, а NICE.SSA строится на открытых компонентах (Suricata, Vector, OpenSearch, EveBox, Arkime, Scirius) без «чёрных ящиков».

:contentReference[oaicite:3]{index=3}

- Яндекс Cloud-нативность: NICE.SSA использует зеркалирование VPC и сервисы хранения/аналитики в Яндекс Cloud, а OpenVAS разворачивается как отдельная VM из Marketplace и управляется через Yandex Cloud Console — без сложных туннелей, VPN и переноса трафика за пределы облака.

:contentReference[oaicite:4]{index=4}

- Единый operational-подход: оба продукта поставляются в виде готовых образов, поддерживают systemd-управление и хорошо вписываются в стандартные процессы мониторинга, резервирования и IAM в Яндекс Cloud. Это снижает операционные расходы и упрощает внедрение комплексного контура безопасности.

:contentReference[oaicite:5]{index=5}

- Удобство для SOC и DevSecOps: SOC получает связанный набор артефактов — сетевые срабатывания, PCAP-сессии, отчёты об уязвимостях и их статус; DevSecOps может использовать OpenVAS API и OpenSearch NICE.SSA для интеграции с CI/CD и автоматической проверки микросервисов и окружений.

:contentReference[oaicite:6]{index=6}

Рекомендуемая модель эксплуатации:

NICE.SSA работает постоянно как фоновый сетевой мониторинг и источник инцидентов, OpenVAS запускается по расписанию (например, еженедельно/ежемесячно) и дополнительно по триггерам, когда сенсор фиксирует аномальную активность в конкретном сегменте или на конкретном хосте.

Пример связанной аналитики

Ниже приведён пример простого запроса к OpenSearch для поиска событий Suricata по хостам, для которых OpenVAS уже обнаружил критические уязвимости (условно, список IP идёт из отчёта GVM):

```
# Пример: фильтруем события Suricata по списку уязвимых хостов
VULN_IPS="10.0.1.10 10.0.2.15 10.0.3.5"

for ip in $VULN_IPS; do
  echo "=== События для уязвимого хоста $ip ==="
  curl -s -X POST "http://localhost:9200/suricata-events-*/_search" \
    -H 'Content-Type: application/json' \
    -d '{
      \"size\": 10,
```

```

    \"sort\": [{ \"@timestamp\": { \"order\": \"desc\" } }],
    \"query\": {
      \"bool\": {
        \"should\": [
          { \"term\": { \"src_ip\": \"$ip\" } },
          { \"term\": { \"dest_ip\": \"$ip\" } }
        ]
      }
    }
  } | jq '.hits.hits[]._source | {timestamp, src_ip, dest_ip, alert}'
done

```

Такой подход позволяет быстро найти те инциденты, где уязвимые по данным OpenVAS хосты уже участвуют в подозрительных сетевых взаимодействиях, и сфокусировать усилия IR-команды именно там.

В связке NICE.SSA и OpenVAS выполняют разные, но дополняющие друг друга роли: сенсор сети фиксирует, что действительно происходит в трафике, а сканер уязвимостей объясняет, почему это стало возможным. Вместе они дают целостное и доказуемое представление о рисках для сервисов в Яндекс Cloud.

1.3. Особенности NICE.SSA

NICE.Сенсор Сетевых Атак (NICE.SSA) спроектирован как законченный продукт для Яндекс Cloud: он разворачивается в один шаг, сразу включает полный стек анализа трафика и предоставляет удобный, единый вход для всех интерфейсов расследования и администрирования. Ниже — ключевые особенности, которые определяют практическую ценность решения.

...

Развертывание в один клик в Yandex Cloud Marketplace

NICE.SSA поставляется как готовый образ для Yandex Cloud Marketplace. Для запуска сенсора достаточно выбрать продукт, указать базовые параметры виртуальной машины (vCPU, память, диск) и привязку к нужным сетям/подсетям. Все основные сервисы (Suricata, Vector, OpenSearch, вспомогательные компоненты) автоматически конфигурируются и запускаются при первичном старте.

Такой подход особенно важен для команд, которые не готовы тратить недели на сборку NDR-стека вручную: сенсор появляется в инфраструктуре за минуты и сразу готов к приёму зеркалируемого трафика из VPC, что сокращает время от идеи до первых реальных инцидентов в дашбордах.

Полностью готовый стек из проверенных компонентов

В состав NICE.SSA входит тщательно подобранный набор открытых компонентов, каждый из которых отвечает за свой слой аналитики:

- **Suricata** — высокопроизводительный анализатор трафика и движок детекций (IDS/NDR).
- **Vector** — коллектор и конвейер доставки событий EVE в хранилище.
- **OpenSearch** — поисково-аналитический движок и база для дашбордов.
- **EveBox** — интерфейс для триажа, маркировки и расследования алертов.
- **Arkime** — полнотекстовый анализ PCAP-сессий и глубокое сетевое расследование.
- **Scirius** — управление правилами и источниками детекций Suricata.
- **NGINX** — фронтальной reverse прокси и единая точка входа для всех UI.

Администратору не нужно самостоятельно собирать и согласовывать эти составляющие: в NICE.SSA они уже интегрированы, используют согласованные схемы индексов, подготовленные настройки безопасности и единый стиль аутентификации и доступа.

Автоматическая настройка сбора, хранения и визуализации событий

После развёртывания NICE.SSA автоматически конфигурирует цепочку: Suricata → Vector → OpenSearch. Включаются необходимые журналы EVE, создаются индексы и шаблоны в OpenSearch, применяется политика жизненного цикла данных (ILM), а также импортируются базовые наборы дашбордов и saved search'ей.

Это означает, что сразу после подключения зеркалируемого трафика оператор может открыть дашборды и увидеть:

- общую картину сетевой активности (топ IP, портов, протоколов);
- сводку алертов по приоритетам и категориям;
- подозрительные сессии и аномальные взаимодействия;
- расширенный контекст по выбранным инцидентам.

За кулисами работает оптимизированная схема хранения: горячие/тёплые индексы, управляемые сроки хранения, экономное использование диска и однозначные правила архивации, что снижает эксплуатационные риски и позволяет предсказуемо планировать ресурсы.

Интерфейс через единую точку входа (Nginx reverse proxy)

Все пользовательские интерфейсы — OpenSearch Dashboards, EveBox, Arkime, Scirius — доступны через единую точку входа на базе **NGINX**. С точки зрения пользователя и оператора это выглядит как один защищённый веб-портал, в котором разные разделы отвечают за различные задачи: обзоры, инциденты, PCAP, управление правилами и т. д.

NGINX отвечает за:

- терминацию TLS и единые сертификаты;
- маршрутизацию по URI (например, `/dashboards`, `/evebox`, `/arkime`, `/scirius`);
- базовую аутентификацию или интеграцию с внешними IAM/SSO;
- ограничение доступа по IP/сетям и базовые механизмы защиты.

Такой подход упрощает эксплуатацию: не нужно помнить множество портов и адресов, открывать дополнительные сервисы наружу или отдельно настраивать TLS для каждого компонента. Для интеграции с существующей инфраструктурой достаточно один раз включить проксирование и задать нужные правила доступа.

В совокупности эти особенности превращают NICE.SSA из набора отдельных открытых компонентов в цельный продукт: он быстро внедряется, сразу приносит понятимую пользу и остаётся прозрачным и управляемым на всём жизненном цикле — от пилота до промышленной эксплуатации.

...

::contentReference[oaicite:0]{index=0}

Яндекс Cloud (VPC Mirroring, TAP, SPAN-порты и др.). Этот трафик направляется на сетевые интерфейсы виртуальной машины с NICE.SSA. Сенсор не вмешивается в прохождение боевого трафика, а лишь наблюдает за ним, обеспечивая пассивный контроль и анализ.

Модуль **Suricata** выполняет:

- декодирование пакетов и восстановление потоков (reassembly);
- анализ протоколов L2–L7 (HTTP, TLS, DNS, SMTP, SSH и т. д.);
- применение сигнатур, эвристик и правил детектирования;
- генерацию событий в формате **EVE JSON**, включающем алерты, метаданные потоков, файлы и др.

На этом уровне формируется «сырое» множество событий, максимально приближенное к реальному сетевому поведению сервисов и пользователей.

Suricata → Vector → OpenSearch

Сгенерированные Suricata события в формате EVE JSON поступают в конвейер **Vector**. Vector читает файлы журналов, нормализует и обогащает записи, а затем доставляет их в кластер **OpenSearch**.

Типичный конвейер включает:

- источник *file*, который отслеживает `/var/log/suricata/eve.json`;
- преобразования (remap/transform), где нормализуются временные метки, IP-адреса, типы событий, поля категоризации;
- приёмник *elasticsearch/opensearch*, который пишет данные в индексы вида `suricata-events-<дата>` с учётом подготовленных шаблонов и ILM-политик.

На выходе мы получаем структурированное хранилище событий, доступное для полнотекстового поиска, агрегаций, корреляции и визуализации. OpenSearch в этой архитектуре — центральный источник правды о сетевой активности и инцидентах.

OpenSearch → Dashboards

Над индексами OpenSearch работает интерфейс визуализации и аналитики — OpenSearch Dashboards. Он предоставляет:

- готовые дашборды по активности сети, типам атак, топам IP/портов/протоколов;
- механизмы ad-hoc поиска и фильтрации событий (по IP, сигнатуре, имени

правила, географии и другим полям);

- инструменты построения собственных визуализаций и сохранённых запросов для внутренних регламентов и отчётности;
- возможность корреляции сетевых данных с другими источниками (при необходимости расширения стека).

Dashboards — основная точка входа для обзора состояния сети и построения сводных отчётов, которые читают менеджеры, аудиторы и руководители SOC.

EveBox — аналитика и инцидент-менеджмент

EveBox использует те же данные OpenSearch, но фокусируется на жизненном цикле инцидентов. Это рабочий инструмент SOC-аналитиков, где:

- заведены представления по алертам и событиям Suricata;
- доступна фильтрация и поиск по ключевым полям;
- есть механизмы пометок (acknowledge), комментариев, группировки и эскалации;
- формируются контекстные представления инцидентов (таймлайн, связанная активность).

EveBox закрывает практическую сторону обработки событий: помогает аналитикам быстро переходить от «шума» к значимым инцидентам и фиксировать ход расследования для последующего аудита.

Arkime — детальный анализ PCAP

Arkime отвечает за долговременное хранение и анализ копий сетевого трафика (PCAP). Он записывает сессии, индексирует метаданные и предоставляет веб-интерфейс, в котором можно подняться от события Suricata до конкретных пакетов.

Типичный сценарий:

- аналитик видит алерт в Dashboards или EveBox;
- по IP/порту/времени открывает соответствующую сессию в Arkime;
- анализирует содержимое диалога, заголовки протоколов, полезную нагрузку, повторяет сессию;
- экспортирует PCAP-файл для дополнительного офлайн-анализа (Wireshark, специализированные утилиты).

Arkime превращает абстрактные алерты в конкретные «сетевые доказательства», которые можно показать разработчикам, внешним аудиторам или использовать в рамках формализованных процедур IR.

Scirius — управление правилами Suricata

Scirius — веб-интерфейс, через который управляющий персонал настраивает правило-сет Suricata: источники правил, их приоритеты, отключения, локальные дополнения и модификации.

Через Scirius можно:

- подключать и обновлять внешние наборы правил (ET, собственные репозитории и др.);
- управлять категориями и включать/выключать отдельные правила под специфику инфраструктуры;
- создавать локальные детекции для внутренних сервисов и протоколов;
- отслеживать историю изменений и версии конфигураций.

Таким образом, Scirius связывает мир аналитиков (которым важно качество детекций) и администраторов, которые отвечают за аккуратное внедрение изменений в продакшен.

Nginx — единый шлюз для всех сервисов

Все перечисленные веб-интерфейсы (OpenSearch Dashboards, EveBox, Arkime, Scirius) доступны через единый шлюз на базе **Nginx**. Это:

- единый URL для доступа к функционалу NICE.SSA;
- терминация TLS, управление сертификатами и шифросьютами;
- централизованная аутентификация (Basic Auth, интеграция с внешними IAM/SSO);
- ограничение доступа по IP/сетям, rate limiting и другие меры защиты;
- прозрачный роутинг по путям, например: `/dashboards` — OpenSearch Dashboards
`/evebox` — EveBox
`/arkime` — Arkime
`/scirius` — Scirius

Вместо набора разрозненных сервисов администратор и аналитики получают один портал, через который можно как смотреть дашборды, так и проводить расследования и управлять правилами.

В совокупности такая архитектура превращает NICE.SSA в «конвейер сетевых доказательств»: трафик попадает в Suricata, события нормализуются Vector, сохраняются и анализируются в OpenSearch, триажируются в EveBox, детализируются в Arkime, а качество детекций управляется через Scirius — всё это за единым Nginx-шлюзом и с возможностью развёртывания в один клик в Яндекс Cloud.

...

::contentReference[oaicite:0]{index=0}

2.2. Варианты развёртывания

NICE.Сенсор Сетевых Атак (NICE.SSA) спроектирован так, чтобы одинаково уверенно работать как в облачной инфраструктуре Яндекс Cloud, так и в локальных (on-premise) средах. Базовый сценарий — полностью готовый образ/стек для Яндекс Cloud, при этом on-premise-вариант находится в статусе *coming soon* и разрабатывается с учётом тех же принципов: быстрый старт, предсказуемое обновление и прозрачная эксплуатация.

...

Развёртывание в Яндекс Cloud

Облачный сценарий — основной и рекомендованный способ работы с NICE.SSA. Продукт поставляется как готовый образ/шаблон в **Yandex Cloud Marketplace**, а также как преднастроенный *docker-stack* на базе NiceOS, упакованный в виртуальную машину.

При развёртывании в Яндекс Cloud выполняются следующие шаги:

- пользователь выбирает продукт NICE.SSA в каталоге Marketplace;
- определяет характеристики ВМ (vCPU, RAM, диск) и сети/подсети для подключения;
- при необходимости указывает параметры доступа (админские пароли, ключи, доменное имя);
- после создания инстанса автоматически поднимается docker-стек OpenSearch/Vector/EveBox/Arkime/Scirius/NGINX;
- настраиваются системные сервисы Suricata и вспомогательные unit'ы (healthcheck, timers, logrotate).

Ключевой момент — **автоматическая настройка сети и mirroring-endpoint**. В

составе документации и мастеров развертывания описывается, как:

- создать источники зеркалирования (VPC Flow Mirror / TAP) для нужных подсетей и групп ВМ;
- направить зеркалируемый трафик на интерфейс сенсора;
- проверить, что Suricata видит трафик и генерирует события EVE;
- убедиться, что события индексируются в OpenSearch и отображаются в дашбордах.

В результате развёртывание в Яндекс Cloud выглядит для пользователя как «один клик + минимум параметров», после чего он сразу получает работающий NDR-контур, интегрированный с облачной сетью и готовый к приёму трафика из продакшен-сегментов.

Развёртывание On-Premise coming soon

On-premise-вариант NICE.SSA ориентирован на организации, которые по требованиям безопасности или регуляtorики размещают критичную инфраструктуру в собственных дата-центрах и не могут (или не хотят) использовать публичное облако. В этом сценарии сенсор разворачивается на базе NiceOS на выделенном сервере или виртуальной машине внутри периметра компании.

Планируемые особенности on-premise-поставки:

- установка на NiceOS из репозитория RPM-пакетов (suricata, suricata-vector, suricata-opensearch-stack и др.);
- преднастроенный docker-stack или systemd-юниты для OpenSearch, EveBox, Arkime, Scirius и Nginx;
- сценарии подключения к портам зеркалирования (SPAN), аппаратным TAP и агрегационным устройствам;
- возможность использовать как локальные диски, так и внешние СХД для хранения PCAP и индексов;
- поддержка интеграции с существующими SIEM/лог-хранилищами по API или через экспорт индексов/PCAP.

При этом on-premise-архитектура повторяет облачную модель:

- Suricata остаётся основным анализатором трафика;
- Vector — конвейером доставки событий;
- OpenSearch — хранилищем и аналитическим ядром;

- EveBox, Arkime и Scirius — интерфейсами для IR, PCAP-аналитики и управления правилами;
- Nginx — единым шлюзом ко всем веб-инструментам.

Таким образом, миграция между облачным и on-premise-сценариями (или гибридное использование) будет максимально простой: меняется только среда и способ доставки пакетов/образов, а логика работы, дашборды и процессы расследования остаются одинаковыми. Это снижает порог вхождения для команд SOC и DevSecOps, которые работают сразу с несколькими площадками.

Независимо от выбранного варианта развертывания, NICE.SSA остаётся единым продуктом с общей моделью данных, одинаковыми интерфейсами и предсказуемыми процессами эксплуатации. Облако даёт скорость и гибкость, on-premise — максимальный контроль и изоляцию, а архитектура решения позволяет сочетать оба подхода.

...

::contentReference[oaicite:0]{index=0}

2.3. Потоки данных и взаимодействие сервисов

В NICE.SSA данные движутся по чёткой цепочке: от сырых сетевых пакетов до структурированных событий, дашбордов, инцидентов и PCAP-артефактов. Ниже описано, где именно что хранится, какие порты и протоколы используются, как Vector буферизует данные, как OpenSearch индексирует события, как EveBox работает с индексами `suricata-events-*` и как Arkime получает доступ к PCAP.

...

Уровни данных и места хранения

В архитектуре NICE.SSA можно условно выделить четыре уровня данных:

- **Сырые пакеты (L2–L7):**
попадают на интерфейс сенсора (VPC mirroring/TAP/SPAN). Часть пакетов может дополнительно записываться Arkime в PCAP-хранилище для последующего анализа.
- **События Suricata (EVE JSON):**
сохраняются в файловой системе (обычно `/var/log/suricata/eve.json` или ротация в

несколько файлов). Это основной источник событий для Vector.

- **Индексы OpenSearch:**

структурированные документы в индексах «suricata-events-<дата>» и дополнительных индексах (например, для статистики, метаданных Arkime и служебных объектов).

- **Локальные БД и служебные файлы:**

конфигурационная БД EveBox (например, `/var/lib/evebox/config.sqlite`), внутренние БД Arkime, конфиги Scirius, файлы настроек Nginx и др.

Такой дизайн позволяет чётко разделить: где хранится «дорогой» PCAP, где живут индексируемые события, а где лежат только служебные или конфигурационные данные.

Порты и протоколы взаимодействия

Компоненты стека общаются друг с другом преимущественно по HTTP/HTTPS внутри сенсора, а наружу выходит только Nginx.

```
# Типичные порты внутри NICE.SSA (можно адаптировать под политику компании)
```

```
...
```

```
# Хранилище и аналитика
```

```
OpenSearch REST API    : 9200/tcp (HTTP)
```

```
OpenSearch node/metrics : 9600/tcp, 9650/tcp (HTTP)
```

```
# Визуализация и UI
```

```
OpenSearch Dashboards  : 5601/tcp (HTTP, только внутри)
```

```
EveBox                  : 5636/tcp (HTTP, только внутри; порт может отличаться)
```

```
Arkime (viewer)         : 8005/tcp или 8443/tcp (HTTP/HTTPS, только внутри)
```

```
Scirius                  : 8008/tcp (HTTP, только внутри)
```

```
# Внешний шлюз
```

```
Nginx (NICE.SSA портал) : 80/tcp, 443/tcp (HTTP/HTTPS, наружу)
```

Suricata и Vector, как правило, не поднимают отдельные внешние HTTP-порты: Suricata пишет EVE в файлы, а Vector читает их локально и уже сам общается с OpenSearch по HTTP. Все пользовательские запросы снаружи приходят только на Nginx, который проксирует их к нужным внутренним сервисам.

```
...
```

Vector: буферизация и надёжная доставка

Vector выполняет роль конвейера, который превращает поток EVE-событий в устойчивый поток документов для OpenSearch, с возможностью буферизации при проблемах со связью.

- **Чтение:** источник типа `file` следит за `/var/log/suricata/eve.json` с учётом ротации (inode-трекинг), что исключает потерю событий при смене файла.
- **Нормализация:** трансформы (например, `remap`) приводят поля к единому виду: нормализуют временные метки, добавляют имя сенсора, конвертируют типы полей (IP, числа), убирают лишний мусор.
- **Буферизация:** sink OpenSearch использует внутренние очереди. При недоступности OpenSearch события временно накапливаются в памяти и/или на диске (в зависимости от настроек), после чего отправляются повторно, когда соединение восстановится.
- **Пакетная отправка:** Vector отправляет события батчами, уменьшая нагрузку на сеть и OpenSearch, но при этом стараясь минимизировать задержку для новых алертов.

Пример фрагмента конфига, иллюстрирующий буферизацию и отправку:

```
sinks:  
...  
  
opensearch_eve:  
  type: elasticsearch  
  inputs: [normalize_eve]  
  endpoints: ["[http://127.0.0.1:9200](http://127.0.0.1:9200)"]  
  index: "suricata-events-%Y.%m.%d"  
  bulk:  
    index: "suricata-events-%Y.%m.%d"  
  request:  
    retry_attempts: 10  
    retry_backoff_secs: 5  
  buffering:  
    type: disk # disk / memory  
    max_size: 1024mb
```

...

OpenSearch: индексация и жизненный цикл данных

OpenSearch превращает поток JSON-документов в индексируемые записи, по которым можно делать поиск, агрегации и корреляцию.

- **Индексация:** события попадают в ежедневные индексы вида `suricata-events-YYYY.MM.DD`. Шаблоны индексов задаются заранее (mapping, настройки анализаторов, alias'ы).
- **Типы полей:** IP-адреса мапятся на тип `ip`, временные метки — на `date`, численные поля (байты, пакеты, порты) — на `long/integer`. Это важно для корректных агрегаций и сортировок.
- **ILM-политики:** для индексов настраиваются политики жизненного цикла (hot/warm/delete), которые автоматически переводят старые данные в тёплое хранилище или удаляют их по истечению срока, заданного в политике.
- **Alias'ы и Data View:** для удобства работы используются alias'ы (например, `suricata-events`), на которых строятся Data View в OpenSearch Dashboards. Это позволяет работать с историей, не думая о реальных именах индексов.

Таким образом, OpenSearch становится центральной точкой, где сходятся все сетевые события: их можно анализировать как через дашборды, так и программно (через REST API) или интегрировать с внешними системами.

EveBox: использование индексов Suricata-Events

EveBox не хранит копию событий у себя; он использует те же индексы OpenSearch, что и дашборды. Обычно это alias наподобие `suricata-events`, указывающий на все актуальные индексы `suricata-events-*`.

- При открытии списка алертов EveBox делает запрос к OpenSearch с фильтром по документам типа `"event_type": "alert"` и сортирует по полю `@timestamp`.
- При просмотре подробностей алерта EveBox подтягивает полный документ, включая поля сигнатуры, IP/портов, протокола, направления, и при необходимости строит «окно времени», показывающее соседние события вокруг инцидента.
- Пометки (ack, dismiss, комментарии) могут записываться либо в отдельный индекс OpenSearch, либо в локальную БД EveBox (например, SQLite), что даёт возможность хранить «служебные» отметки отдельно от исходных событий.

За счёт использования индексов `suricata-events-*` EveBox всегда работает с теми же

данными, что и дашборды: аналитик может перейти от графика в OpenSearch Dashboards к конкретной записи в EveBox и обратно без потери контекста.

Arkime: доступ к PCAP и связь с событиями

Arkime параллельно с Suricata получает копию тех же сетевых пакетов (через отдельный capture-процесс на интерфейсе сенсора), записывает их в PCAP-файлы и индексирует метаданные сессий. Метаданные (набор полей о сессии) также могут храниться в OpenSearch в отдельных индексах (например, `arkime-sessions-*`).

- **Хранение PCAP:** PCAP-файлы размещаются на выделенном диске/томе (например, `/var/lib/arkime/raw`), где для каждого временного окна создаются файлы фиксированного размера. Это позволяет Arkime быстро находить нужные сегменты.
- **Индексация:** capture-процесс Arkime извлекает из трафика метаданные (IP, порты, протокол, временные диапазоны, теги) и пишет их в поисковый backend (OpenSearch), тем самым обеспечивая быстрый поиск нужной сессии без сканирования всего PCAP.
- **Связь с Suricata:** по IP/портам/времени можно перейти от события Suricata к соответствующей сессии Arkime. Многие дашборды и интеграции предусматривают кнопку/ссылку «открыть в Arkime» для выбранного алерта.

В результате Arkime становится «слоем доказательств»: всё, что Suricata пометила как подозрительное, можно проверить в сыром трафике вплоть до отдельных пакетов, а PCAP-хранилище может использоваться как источник для дополнительного анализа сторонними инструментами.

Потоки данных в NICE.SSA выстроены так, чтобы никакая информация не пропадала зря: Suricata даёт детализацию на уровне событий, Vector обеспечивает надёжную доставку, OpenSearch превращает поток событий в аналитический слой, EveBox и дашборды — в рабочие инструменты SOC, а Arkime — в хранилище сетевых доказательств. Всё это связано едиными индексами и доступно через один шлюз Nginx.

...

::contentReference[oaicite:0]{index=0}

3. Компоненты решения

...

3.1. Suricata

Suricata — это ядро NICE.SSA: высокопроизводительный анализатор трафика, который сочетает функции IDS (Intrusion Detection System), IPS (Intrusion Prevention System) и NSM (Network Security Monitoring). В контуре NICE.SSA Suricata работает в режиме пассивного наблюдения за зеркалируемым трафиком, генерирует события в формате EVE JSON и передаёт их в конвейер Vector → OpenSearch.

Назначение и основные возможности

Suricata в составе NICE.SSA выполняет несколько ключевых ролей в контуре сетевой безопасности:

- **IDS (Intrusion Detection System):**
анализирует трафик по сигнатурам и правилам, выявляет попытки эксплуатации уязвимостей, сканирования, вредоносные протоколы, C2-каналы и другие признаки атак. Результат — события и алерты, попадающие в OpenSearch.
- **IPS (Intrusion Prevention System):**
может работать в режиме inline (NFQUEUE/AF_PACKET) и блокировать трафик, соответствующий правилам. В контексте NICE.SSA в Яндекс Cloud акцент делается на пассивный мониторинг, но архитектура не исключает использование IPS-режимов в on-premise.
- **NSM (Network Security Monitoring):**
собирает метаданные о потоках, протоколах, HTTP/SSL/DNS/SMTP/SSH и других уровнях L7, формирует полную картину сетевой активности, даже если нет явных сигнатур атак.

В совокупности это позволяет использовать NICE.SSA и как систему обнаружения атак, и как инструмент высокоуровневого мониторинга сетевой безопасности и телеметрии.

Конфигурация и логи Suricata

Основной конфигурационный файл и логи Suricata в NICE.SSA размещаются в стандартных для NiceOS путях:

- `/etc/suricata/suricata.yaml` — главный конфиг Suricata;
- `/var/log/suricata/` — каталог логов (включая `eve.json`);
- `/etc/suricata/` — дополнительные конфиги (`threshold`, `classification`, `reference`);
- `/usr/share/suricata/` — эталонные файлы и правила в поставке;
- `/var/lib/suricata/update` — рабочий каталог `suricata-update`.

Фрагмент базового конфига, отвечающего за включение EVE-логирования (ключевой источник для Vector):

```
# /etc/suricata/suricata.yaml (фрагмент)
```

```
'''
```

```
outputs:
```

```
* eve-log:
```

```
  enabled: yes
```

```
  filetype: regular
```

```
  filename: /var/log/suricata/eve.json
```

```
  community-id: yes
```

```
  community-id-seed: 0
```

```
  types:
```

```
  - alert
```

```
  - dns
```

```
  - http
```

```
  - tls
```

```
  - ssh
```

```
  - smtp
```

```
  - flow
```

```
  - stats
```

Именно на этот файл `eve.json` настроен Vector в конфигурации

`/etc/vector/suricata.vector.yaml`, обеспечивая непрерывный экспорт событий в OpenSearch.

Правила и механизм обновления (suricata-update)

Сила Suricata во многом определяется качеством и актуальностью правил. В NICE.SSA управление правилами реализовано через стандартный механизм `suricata-update` и интеграцию с веб-интерфейсом Scirius.

Базовая структура правил в системе:

```
/usr/share/suricata/rules/      # эталонные правила из поставки
```

```
app-layer-events.rules
```

```
decoder-events.rules
```

```
dns-events.rules
http-events.rules
tls-events.rules
...
/etc/suricata/rules/      # активные/локальные правила
local.rules
override.rules
custom-*.rules
```

Основные сценарии работы с `suricata-update`:

```
# Обновить индекс источников и сами правила
sudo suricata-update

# Посмотреть список подключённых источников

sudo suricata-update list-sources

# Подключить новый источник правил

sudo suricata-update enable-source my-custom-source

# Применить изменения (обычно через systemd)

sudo systemctl restart suricata.service
```

В NICE.SSA политики подключения источников, включения/отключения категорий и тонкой правки правил, как правило, выполняются через Scirius, который управляет конфигурацией `suricata-update` и итоговым набором файлов в `/etc/suricata/rules/`.

...

Интеграция с OpenSearch через EVE JSON

Интеграция Suricata с аналитической частью NICE.SSA построена на стандартизированном формате **EVE JSON**. Все события (алерты, потоки, HTTP/TLS/DNS и др.) записываются в `/var/log/suricata/eve.json`, откуда их забирает Vector.

Типичный фрагмент события алерта в EVE:

```
{
...

"timestamp": "2025-11-23T12:34:56.789012+0000",
"event_type": "alert",
"alert": {
```

```
"signature_id": 2100498,  
"signature": "ET MALWARE Possible Malicious Traffic",  
"category": "A Potential Malware Activity",  
"severity": 2  
},  
"src_ip": "10.1.2.3",  
"src_port": 54321,  
"dest_ip": "198.51.100.10",  
"dest_port": 80,  
"proto": "TCP",  
"flow_id": 1234567890123456  
}
```

Далее конвейер выглядит так:

- Vector читает EVE-файл и применяет трансформации (нормализация временных меток, типов полей);
- события отправляются в OpenSearch в индекс `suricata-events-YYYY.MM.DD`;
- OpenSearch Dashboards используют Data View по `suricata-events-*` для построения графиков и отчётов;
- EveBox обращается к тем же индексам, фильтруя по `"event_type": "alert"` и параметрам инцидента;
- Arkime может быть связан с событиями Suricata по `flow_id`, IP/портам и времени.

Важный эффект такого подхода — вся аналитика и расследования в NICE.SSA строятся на одном и том же потоке данных, исходящей от Suricata. Это обеспечивает консистентность между дашбордами, инцидент-менеджментом, PCAP-аналитикой и экспортом событий во внешние системы.

...

Suricata в NICE.SSA — это не просто движок сигнатур: это центральный сенсор, который превращает пассивный зеркалируемый трафик в структурированные события, пригодные для поиска, корреляции и расследований. От качества её конфигурации и правил зависит точность и глубина всей NDR-платформы.

...

```
::contentReference[oaicite:0]{index=0}
```

3.2. Vector

Vector в NICE.SSA — это транспортный и нормализующий слой для событий Suricata. Он забирает EVE JSON из файлов, приводит записи к единому формату, буферизует их

при сбоях, отправляет в OpenSearch и одновременно контролируется через отдельный healthcheck-сервис. Благодаря Vector поток данных от Suricata до OpenSearch остаётся устойчивым и предсказуемым.

...

Роль Vector в стеке NICE.SSA

В архитектуре NICE.SSA Vector выполняет сразу несколько функций на пути от Suricata к OpenSearch:

- **Транспорт событий:** читает файлы логов Suricata (EVE JSON) и отправляет их в OpenSearch по HTTP, используя оптимизированную пакетную отправку.
- **Нормализация и обогащение:** приводит поля к нужным типам (IP, дата, числа), добавляет служебные атрибуты (имя сенсора, источник, теги среды — prod/test и т. д.).
- **Буферизация и ретраи:** при недоступности OpenSearch временно хранит события в памяти или на диске и повторяет попытки отправки, чтобы не терять данные при кратковременных сбоях.
- **Разделение потоков:** при необходимости может раскидывать разные типы событий (alerts, flows, http, tls и др.) по отдельным индексам или контурам обработки.

Vector делает работу Suricata и OpenSearch независимыми по времени: Suricata продолжает писать EVE, даже если OpenSearch перезапускается или испытывает нагрузку, а Vector берёт на себя выравнивание потока.

Конфигурация Vector: /etc/vector/suricata.vector.yaml

В NICE.SSA конфигурация Vector, отвечающая за работу с Suricata, хранится в файле: `/etc/vector/suricata.vector.yaml`. В ней описываются:

- **sources** — откуда читать события (файлы, journald и др.);
- **transforms** — как преобразовывать и нормализовать записи;
- **sinks** — куда отправлять данные (OpenSearch, файлы, stdout, др.).

Пример упрощённого конфига Vector, иллюстрирующий схему источников, трансформов и sink:

```
# /etc/vector/suricata.vector.yaml (пример)
```

```
...
```

```

sources:
  suricata_eve:
    type: file
    include:
      - /var/log/suricata/eve.json
    read_from: end      # начинать с конца файла при старте
    ignore_older: 7d    # игнорировать слишком старые файлы
    multiline:
    mode: none

  transforms:
    normalize_eve:
      type: remap
      inputs: [suricata_eve]
      source: |
        .@timestamp = to_timestamp!(.timestamp)
        .sensor = "nice-ssa"
        .stream = "suricata-eve"
        # Приведение типов
        .src_port = to_int!(.src_port)
        .dest_port = to_int!(.dest_port)

  sinks:
    opensearch_eve:
      type: elasticsearch
      inputs: [normalize_eve]
      endpoints: ["[http://127.0.0.1:9200](http://127.0.0.1:9200)"]
      index: "suricata-events-%Y.%m.%d"
      mode: "bulk"
      request:
        retry_attempts: 10
        retry_backoff_secs: 5
      buffering:
        type: "disk"
        max_size: 1024mb

```

В реальной поставке NICE.SSA в этот файл дополнительно добавляются настройки логирования самого Vector, ограничение объёма буфера, тюнинг размеров batch и прочие параметры, влияющие на производительность.

...

Контроль работоспособности: `suricata-vector-healthcheck`

Чтобы гарантировать стабильную работу конвейера событий, в NICE.SSA предусмотрен отдельный healthcheck-сервис для Vector:

- `suricata-vector-healthcheck.service` — systemd-юнит, выполняющий проверки;
- `suricata-vector-healthcheck.timer` — таймер, запускающий проверки по расписанию.

Сценарий healthcheck обычно включает:

- проверку статуса основного сервиса Vector (`systemctl is-active vector`);
- опционально — проверку возможности записать тестовый документ в OpenSearch;
- логирование результатов в `journalctl` для последующего мониторинга.

Пример упрощённого systemd-юнита healthcheck:

```
# /usr/lib/systemd/system/suricata-vector-healthcheck.service (пример)
...

[Unit]
Description=Healthcheck Vector для NICE.SSA
After=network-online.target

[Service]
Type=oneshot
ExecStart=/usr/sbin/suricata-vector-healthcheck

# можно добавить Environment= для параметров

[Install]
WantedBy=multi-user.target
```

Пример таймера, запускающего healthcheck, скажем, раз в минуту:

```
# /usr/lib/systemd/system/suricata-vector-healthcheck.timer (пример)
[Unit]
Description=Периодический healthcheck Vector для NICE.SSA

[Timer]
OnBootSec=2min
OnUnitActiveSec=60s
Unit=suricata-vector-healthcheck.service

[Install]
WantedBy=timers.target
```

...

Эксплуатация и диагностика Vector

Vector — критическая часть конвейера, поэтому в NICE.SSA предусмотрен набор типовых команд и практик для эксплуатации:

```
# Проверить статус основного сервиса
```

```
...
```

```
sudo systemctl status suricata-vector.service
```

```
# Посмотреть логи за последние 10 минут
```

```
journalctl -u suricata-vector.service --since "10 min ago"
```

```
# Проверить работу healthcheck
```

```
sudo systemctl status suricata-vector-healthcheck.timer
```

```
journalctl -u suricata-vector-healthcheck.service --since "10 min ago"
```

```
# Тест: отправить один документ напрямую в OpenSearch
```

```
curl -s -XPOST
```

```
"[http://127.0.0.1:9200/suricata-events-test/_doc](http://127.0.0.1:9200/suricata-events-test/_doc)"
```

```
-H 'Content-Type: application/json'
```

```
-d '{"message": "vector-test", "@timestamp": "2025-11-23T12:00:00Z"}'
```

В документации NICE.SSA для эксплуатационных команд обычно описывается:

- как корректно изменять `/etc/vector/suricata.vector.yaml` и перезапускать сервис;
- какие типичные ошибки могут появляться в логах (проблемы с индексами, правами, размером batch);
- как интерпретировать сообщения healthcheck и какие действия предпринимать при нарушении доставки.

В итоге Vector в NICE.SSA воспринимается как «транспортный двигатель» платформы: пока он исправен, все события Suricata надёжно доходят до OpenSearch, становятся видимыми в дашбордах и доступными для EveBox и других компонентов.

```
...
```

Если Suricata — глаза и уши NICE.SSA, то Vector — это кровеносная система платформы: он перекачивает события от сенсора к аналитическому ядру, следит за их целостностью и помогает пережить временные сбои в хранилище, не теряя информации.

```
...
```

```
::contentReference[oaicite:0]{index=0}
```

3.3. OpenSearch

OpenSearch в NICE.SSA — это основное хранилище и аналитический движок для событий Suricata. Все данные, которые проходит через Vector, попадают в индексы OpenSearch, где становятся доступными для полнотекстового поиска, агрегаций, построения дашбордов и работы EveBox/Arkime. От качества настройки индексов и ILM-политик зависит производительность и предсказуемость платформы в долгосрочной перспективе.

...

Хранилище и аналитический движок

В контуре NICE.SSA OpenSearch выполняет сразу несколько ключевых функций:

- **Документное хранилище событий:**
каждое событие EVE (алерт, поток, HTTP-запрос, TLS-сессия, DNS-запрос и т. д.) превращается в документ, который попадает в индекс с правильными типами полей.
- **Поисковый движок:**
позволяет выполнять фильтрацию и поиск по любым полям (IP, порты, имена правил, категории, гео, юзер-агенты, домены и др.), в том числе через REST API.
- **Агрегации и аналитика:**
поддерживает построение метрик и графиков (top-N, гистограммы по времени, распределения по сегментам, пользователям, типам атак), которые используются в OpenSearch Dashboards.
- **Основа для внешней интеграции:**
любые внешние системы (SIEM, отчётные сервисы, внутренние аналитические утилиты) могут забирать данные из OpenSearch по API, не вмешиваясь в работу Suricata и Vector.

Таким образом, OpenSearch — это «центр тяжести» NICE.SSA: все компоненты либо пишут в него данные, либо считывают из него информацию для визуализации и исследований.

Шаблоны индексов и ILM-политики

из `/usr/share/suricata/opensearch/`

Пакет `suricata-opensearch` в NICE.SSA содержит набор готовых артефактов для OpenSearch в каталоге `/usr/share/suricata/opensearch/`, в том числе:

- `template-suricata-events.json` — шаблон индексов событий Suricata;
- `ilm-suricata-events.json` — ILM-политика жизненного цикла данных;
- `pipeline-suricata-eve.json` — ingest-пайплайн для EVE-событий (при использовании ingest node).

При первоначальной настройке NICE.SSA специальный скрипт (например, `/usr/sbin/suricata-opensearch-setup`) применяет эти шаблоны к кластеру OpenSearch:

```
# Пример применения шаблона и ILM-политики (упрощённо)
'''

curl -X PUT
"[http://127.0.0.1:9200/_ilm/policy/suricata-events](http://127.0.0.1:9200/_ilm/policy/suricata-events)"
-H 'Content-Type: application/json'
--data-binary @/usr/share/suricata/opensearch/ilm-suricata-events.json

curl -X PUT
"[http://127.0.0.1:9200/_index_template/suricata-events-template](http://127.0.0.1:9200/_index_template/suricata-events-template)"
-H 'Content-Type: application/json'
--data-binary @/usr/share/suricata/opensearch/template-suricata-events.json
```

Упрощённый пример ILM-политики (логика; реальные значения могут быть иными):

```
{
  "policy": {
    "phases": {
      "hot": {
        "min_age": "0d",
        "actions": {
          "rollover": { "max_age": "7d", "max_size": "50gb" }
        }
      },
      "warm": {
        "min_age": "30d",
        "actions": {
          "shrink": { "number_of_shards": 1 }
        }
      },
      "delete": {
        "min_age": "90d",
        "actions": {
          "delete": {}
        }
      }
    }
  }
}
```

```
}
```

Такая схема позволяет автоматически управлять временем хранения и размером индексов: свежие данные живут в «горячих» индексах с максимальной производительностью, старые — переводятся в более компактный формат или удаляются по истечении согласованного срока.

...

Основные индексы: `suricata-events-*` и `suricata-alerts-*`

В базовой конфигурации NICE.SSA используются несколько ключевых индексов (и/или их `alias`'ы), связанных с Suricata:

- **`suricata-events-*`**

Главный индекс семейства, в который Vector пишет все события Suricata (alerts, flow, http, dns, tls и т. д.). Имя, как правило, включает дату: `suricata-events-YYYY.MM.DD`.

- **`suricata-alerts-*`**

Дополнительный индекс или `alias` для алертов; может использоваться как физический отдельный индекс (например, через ingest-пайплайн или отдельный sink Vector), либо как `alias` с фильтром по `event_type=alert`. Удобен для быстрой работы SOC и внешних систем.

- **`Alias`'ы:**

`suricata-events` □ все текущие индексы `suricata-events-*`;

`suricata-alerts` □ все актуальные алертные индексы.

Проверить состояние индексов можно стандартной командой:

```
curl -s "http://127.0.0.1:9200/_cat/indices/suricata-*?v" | sort
```

Именно на эти индексы завязаны Data View в OpenSearch Dashboards и поисковые запросы EveBox. Для пользователя это выглядит как единое логическое хранилище сетевых событий, хотя физически оно может состоять из множества ежедневных индексов с разными фазами ILM.

OpenSearch Dashboards как точка визуализации

Поверх индексов OpenSearch работает **OpenSearch Dashboards** — основной интерфейс для визуализации и интерактивного анализа событий. В NICE.SSA он:

- использует Data View, основанные на шаблонах `suricata-events-*`;
- включает предустановленные дашборды по ключевым сценариям (обзор сети, алерты, DNS/HTTP/TLS и др.);
- предоставляет конструктор визуализаций и возможность сохранять собственные запросы и панели;
- поддерживает фильтрацию по любым полям, time-picker и drill-down до уровней отдельных событий;
- служит основой для экспортируемых отчётов (PDF, CSV, скриншоты) в рамках внутренних регламентов.

Пример создания Data View (логика действий в UI):

1. Открыть OpenSearch Dashboards ▢ "Stack Management" ▢ "Data Views".
2. Создать новый Data View с именем, например, "Suricata Events".
3. В поле Index pattern указать: `suricata-events-*`.
4. В качестве поля временной метки выбрать: `@timestamp`.
5. Сохранить и использовать Data View в Discover, Visualize, Dashboards.

OpenSearch Dashboards — главный инструмент обзора состояния сети и начальной аналитики. Для глубокой работы с инцидентами используется EveBox, а для PCAP — Arkime, но все они опираются на те же индексы OpenSearch, обеспечивая единую картину данных.

Если представить NICE.SSA как «лабораторию сетевой безопасности», то OpenSearch — это её центральное хранилище образцов и журналов: сюда попадает всё, что увидела Suricata; отсюда читают Dashboards, EveBox и Arkime; сюда же подключаются внешние системы. Правильно настроенные шаблоны и ILM-политики превращают этот массив данных в управляемый и прогнозируемый ресурс.

```
::contentReference[oaicite:0]{index=0}
```

3.4. EveBox

EveBox в NICE.SSA — это специализированный интерфейс для расследования событий Suricata, управления алертами и ведения истории реагирования. Если OpenSearch Dashboards — это «радар» и панель обзора, то EveBox — рабочий стол

SOC-аналитика, где инциденты принимают форму кейсов, помечаются, комментируются и закрываются по результатам расследования.

...

Функции EveBox: расследование, корреляция, алерт-менеджмент

EveBox собирает в одном интерфейсе всё, что нужно для повседневной работы SOC с алертами Suricata:

- **Список алертов и событий:**

представление в виде ленты событий с основными полями — время, источник/назначение, сигнатура, категория, критичность, направление трафика. Позволяет быстро ответить на вопрос «что происходит прямо сейчас».

- **Детальные карточки инцидентов:**

при открытии алерта показывается полный JSON-документ события, включая вложенные поля Suricata (alert, http, dns, tls, flow и др.), а также связанные события вокруг выбранного временного диапазона.

- **Корреляция событий:**

EveBox группирует однотипные срабатывания (например, множество алертов по одному и тому же IP или сигнатуре) и позволяет рассматривать их как одну проблему, а не сотни отдельных записей.

- **Управление жизненным циклом алерта:**

алерты можно подтверждать (ack), помечать как ложные (false positive), добавлять комментарии и теги, что формирует историю работы по каждому инциденту.

За счёт этих возможностей EveBox превращает поток сырых событий Suricata в управляемый список задач для SOC/IR-команд — с возможностью отследить, кто и когда принял решение по тому или иному инциденту.

Связь с OpenSearch: `/etc/suricata/opensearch.conf`

EveBox в NICE.SSA не хранит копию событий у себя — он напрямую работает с индексами OpenSearch, в которые Vector пишет EVE-события Suricata. Параметры подключения к OpenSearch вынесены в файл `/etc/suricata/opensearch.conf`, который используется как единый источник настроек для нескольких компонентов (EveBox, вспомогательные скрипты и др.).

Пример содержания `/etc/suricata/opensearch.conf` (логика, могут быть дополнительные

поля):

```
# /etc/suricata/opensearch.conf (пример)
...

OPENSEARCH_URL="[http://127.0.0.1:9200](http://127.0.0.1:9200)"
OPENSEARCH_USERNAME="suricata"
OPENSEARCH_PASSWORD="changeme"
OPENSEARCH_INDEX_PATTERN="suricata-events-*
```

На базе этих переменных при запуске EveBox можно задавать параметры подключения:

```
# Пример запуска EveBox с параметрами из opensearch.conf
./etc/suricata/opensearch.conf

INDEX_PREFIX="${OPENSEARCH_INDEX_PATTERN%%-*}"

EVEBOX_ELASTICSEARCH_URL="$OPENSEARCH_URL"
EVEBOX_ELASTICSEARCH_USERNAME="$OPENSEARCH_USERNAME"
EVEBOX_ELASTICSEARCH_PASSWORD="$OPENSEARCH_PASSWORD"
evebox server
-k
--index "$INDEX_PREFIX"
--host 0.0.0.0
```

В результате EveBox видит те же индексы, что и OpenSearch Dashboards: любые изменения в потоке событий или структуре индексов сразу отражаются и в визуализации, и в интерфейсе расследования.

...

Фильтры и работа с алертами

В EveBox используются гибкие фильтры, которые помогают быстро отобрать интересующие инциденты:

- **По времени:**

быстрые диапазоны (последние 15 минут, 1 час, сутки) и произвольные интервалы — удобно для расследования вчерашних инцидентов или проверки конкретного окна времени.

- **По IP/подсетям:**

поиск по источнику или назначению, фильтрация по подсетям (например, только трафик из DMZ или конкретного сегмента в Яндекс Cloud).

- **По сигнатурам и категориям:**

фильтрация по идентификатору правила (`signature_id`), имени сигнатуры (`signature`) или категории (`category`) — полезно для анализа эффективности конкретного набора правил.

- **По уровню критичности:**

сортировка и фильтрация по `severity`, чтобы в первую очередь обрабатывать высокоприоритетные инциденты.

SOC-аналитик может создавать и сохранять наборы фильтров для типичных сценариев охоты: например, «подозрительный исходящий трафик в интернет», «атаки на веб-приложения», «подозрительные DNS-запросы» или «активность из сегмента разработчиков».

Теги, комментарии и отчётность

Одно из ключевых преимуществ EveBox — возможность «очеловечить» поток алертов, дополняя его метаданными, понятными команде безопасности и аудиторам:

- **Теги:**

к каждому инциденту можно добавлять теги (например, `prod`, `test`, `false-positive`, `under-investigation`, `escalated`). Это облегчает поиск и группировку инцидентов по их статусу и контексту.

- **Комментарии:**

аналитики могут фиксировать в карточке алерта свои наблюдения, гипотезы, результаты проверки (например, «IP принадлежит нашему сканеру», «трафик разрешён политикой», «подозрение на компрометацию»).

- **Отметки ask / закрытие инцидента:**

после завершения расследования инцидент можно пометить как обработанный. Это снижает шум и позволяет сосредоточиться на новых/активных событиях.

- **Отчёты:**

на основе отфильтрованных наборов событий EveBox позволяет выгружать данные (JSON/CSV) для последующей обработки или включения в формальные отчёты, а также использовать эти наборы как базу для плейбуков IR.

Таким образом, EveBox выступает связующим звеном между «сырыми» событиями в OpenSearch и процессами реагирования: здесь рождаются пометки, которые объясняют, что именно произошло, кто это анализировал и какое решение было принято.

EveBox в NICE.SSA — это операционный центр управления алертами Suricata: он показывает события в удобном для человека виде, позволяет накладывать на них смысловой слой (теги, комментарии, статусы) и фиксирует историю реагирования. При этом все данные остаются в OpenSearch, что обеспечивает консистентность с дашбордами и внешними интеграциями.

...

3.5. Arkime

Arkime в NICE.SSA отвечает за долговременную запись сетевого трафика (PCAP) и удобный веб-интерфейс для его анализа. Если Suricata даёт «события о том, что произошло», то Arkime позволяет дословно увидеть, как именно это происходило на уровне пакетов: запросы, ответы, заголовки протоколов, полезную нагрузку и полные сессии.

...

Назначение Arkime: анализ и просмотр PCAP

В составе NICE.SSA Arkime выполняет три ключевые задачи:

- **Запись трафика в PCAP:**

отдельный capture-процесс Arkime принимает копию того же зеркалируемого трафика, что и Suricata, и сохраняет его в виде PCAP-файлов на диск. Запись может быть непрерывной или с ограничениями по объёму/времени.

- **Индексация метаданных сессий:**

для каждого потока (flow/session) Arkime извлекает метаданные (IP, порты, протокол, время начала/окончания, байты, пакеты, теги) и сохраняет их в поисковом backend (OpenSearch). Это позволяет быстро находить нужные сессии без сканирования всего PCAP.

- **Интерактивный просмотр и экспорт:**

через веб-интерфейс Arkime можно просматривать списки сессий, фильтровать их по любым полям, раскрывать содержимое пакетов, следить за потоками и экспортировать интересующие фрагменты в отдельные PCAP-файлы.

Arkime превращает трафик в «сетевой видеорегистратор»: если Suricata подняла алерт, с Arkime всегда можно вернуться назад и посмотреть, как именно выглядел сетевой диалог, который к нему привёл.

Включение записи трафика и связь Arkime с OpenSearch

В NICE.SSA Arkime настраивается так, чтобы:

- capture-процесс слушал тот же интерфейс, что и Suricata (интерфейс зеркалируемого трафика);
- PCAP-файлы укладывались в выделенный каталог/том (обычно `/var/lib/arkime/raw`);
- метаданные сессий сохранялись в OpenSearch в отдельные индексы (например, `arkime-sessions-*`);
- веб-интерфейс Arkime был доступен через Nginx по защищённому URI (например, `/arkime`).

Типичные шаги включения записи трафика:

1. Определить интерфейс, на который приходит зеркалируемый трафик (например, `eth1`).
2. Указать этот интерфейс в конфигурации Arkime capture.
3. Создать каталог для PCAP и настроить права.
4. Настроить подключение к OpenSearch для индексации сессий.
5. Запустить capture-сервис Arkime и убедиться, что сессии появляются в Web UI.

Пример фрагмента конфигурации Arkime (логика; имена файлов и параметры могут отличаться):

```
# /etc/arkime/config.ini (фрагмент примера)

[default]
interface=eth1
pcapDir=/var/lib/arkime/raw
esHost=127.0.0.1:9200
esIndex=arkime-sessions
rotateIndex=daily

# Опционально: ограничение скорости записи/чтения, фильтры BPF и др.
```

После настройки Arkime начинает:

- принимать пакеты с интерфейса `eth1`,

- писать PCAP-файлы в `/var/lib/arkime/raw`,
- создавать записи о сессиях в индексах OpenSearch (`arkime-sessions-YYYY.MM.DD` или аналогичных).

Рекомендации по объёму хранения PCAP и метаданных

PCAP — самый «тяжёлый» тип данных в NICE.SSA, поэтому важно заранее спланировать объём хранения и политику ротации. Основные ориентиры:

- **Отдельный диск/том под PCAP:**

рекомендуется вынести `/var/lib/arkime/raw` на отдельный диск или файловую систему. Это предотвращает ситуацию, когда PCAP заполняет системный том и нарушает работу других компонентов.

- **Оценка среднего трафика:**

в расчётах удобно исходить из дневного объёма (ГБ/сутки). Например, при 50 ГБ/сутки и желаемом хранении в 7 дней нужен объём порядка 350 ГБ только под PCAP (без учёта запаса).

- **Ротация по объёму и времени:**

Arkime поддерживает автоматическое удаление старых файлов при достижении лимита по объёму или времени. Рекомендуется задавать безопасный предел (например, использовать не более 70–80% выделенного тома).

- **Хранение метаданных:**

метаданные сессий (индексы OpenSearch) занимают намного меньше места, чем сами PCAP. Для большинства сценариев достаточно хранить PCAP относительно короткий период (дни/недели), а метаданные — более длительный (месяцы), что позволяет выполнять аналитические запросы без доступа к полному трафику.

- **Гранулярность записи:**

при необходимости можно ограничивать объём PCAP с помощью фильтров BPF (например, не записывать сильно шумные или малозначимые протоколы) и тем самым снизить нагрузку на диск.

В документации NICE.SSA обычно приводятся примерные профили: «минимальный», «средний», «расширенный» — с типичными объёмами трафика, рекомендуемым размером диска под PCAP и сроками хранения для разных типов организаций (тестовые стенды, небольшие компании, крупные облачные проекты).

Связка Suricata + Arkime в расследованиях

Практический сценарий работы с Arkime в NICE.SSA обычно выглядит так:

1. Suricata генерирует алерт (событие в `suricata-events-*`).
2. Аналитик находит этот алерт в OpenSearch Dashboards или EveBox.
3. По IP, портам и временному диапазону аналитик открывает соответствующую сессию в Arkime.
4. В Arkime он изучает содержимое пакетов, заголовки и полезную нагрузку.
5. При необходимости экспортирует PCAP или отдельный поток для дополнительного анализа.

Это позволяет перейти от высокоуровневого описания инцидента («подозрительный HTTP-трафик», «атака на веб-приложение») к конкретным техническим деталям, которые можно показать разработчикам, внешним аудиторам или использовать при реконструкции цепочки атаки.

Arkime в NICE.SSA — это слой «сетевой памяти» платформы: он хранит не только факты, что что-то произошло, но и полную картину того, как именно это выглядело на уровне пакетов. При грамотной настройке объёмов и ротации он становится незаменимым инструментом для глубоких расследований и ретроспективного анализа сложных инцидентов.

...

`::contentReference[oaicite:0]{index=0}`

3.6. Scirius

Scirius в NICE.SSA — это центр управления правилами Suricata: через него выполняется подключение и обновление внешних источников, поиск и анализ сигнатур, включение/выключение категорий, оформление локальных правил и финальный деплой единого ruleset'a на сенсор. Задача Scirius — сделать работу с правилами прозрачной, воспроизводимой и удобной для SOC/DevSecOps-команд.

...

Назначение Scirius: управление правилами и сигнатурами

В NICE.SSA Scirius решает сразу несколько практических задач, связанных с жизненным циклом правил Suricata:

- **Управление источниками правил:**
подключение и отключение внешних feeds (ET, коммерческие подписки, собственные репозитории), настройка расписания обновления, контроль версий

и статусов.

- **Поиск и анализ сигнатур:**

удобный поиск по `sid`, имени сигнатуры, категории, полям протокола; просмотр исходного текста правила и встроенных комментариев; оценка того, что конкретно правило детектирует и как может влиять на шумность.

- **Включение/выключение и тюнинг правил:**

массовое включение/отключение целых категорий, `fine-tune` отдельных правил, перевод действия из `alert` в `drop` (актуально для IPS-режимов), управление приоритетами.

- **Локальные правила:**

создание и сопровождение собственных сигнатур для внутренних сервисов, специфических протоколов и локальных политик, с хранением в отдельных `rule`-файлах.

В отличие от ручного редактирования отдельных `.rules`-файлов, Scirius предоставляет контролируемый процесс с возможностью отката, историей изменений и наглядным интерфейсом выбора, какие именно правила попадут в итоговый `ruleset` сенсора.

Общий поток: синхронизация, компиляция, деплой

В терминах NICE.SSA работа Scirius с правилами может быть описана как цепочка: «получить → отфильтровать → скомпилировать → развернуть».

1. **Получить правила:**

Scirius обновляет подключённые источники (локальные/внешние) через интеграцию с `suricata-update` или собственные механизмы загрузки.

2. **Отфильтровать и настроить:**

администратор в UI решает, какие категории включить/выключить, какие правила переопределить, какие локальные сигнатуры добавить.

3. **Скомпилировать результирующий набор:**

на основе выбранных источников и настроек формируется единый `ruleset`, который будет применён к Suricata.

4. **Развернуть и активировать:**

скомпилированный набор правил записывается в каталог `/etc/suricata/rules/` (или его подкаталоги), после чего иницируется перезапуск/перечитывание правил Suricata через `systemd`.

Таким образом, Scirius становится «оркестратором» правил: он сам не анализирует трафик, но управляет тем, как именно это делает Suricata, и какие сигнатуры будут

действовать на сенсоре в каждый момент времени.

Синхронизация правил и загрузка внешних источников

В NICE.SSA Scirius работает поверх стандартной экосистемы `suricata-update` и дополнительных конфигураций, отвечающих за источники правил. Типичный сценарий:

- В UI Scirius администратор подключает источники: публичные наборы, коммерческие subscriptions, собственные Git/HTTP-репозитории с правилами компании.
- Для каждого источника задаются параметры: URL, тип аутентификации, приоритет, включён/выключен.
- По расписанию (или вручную) Scirius инициирует обновление: фактически запускается `suricata-update` с нужным набором *enable/disable-source*.
- Выгруженные из источников `.rules`-файлы складываются в рабочие каталоги `/var/lib/suricata/update/` и далее используются при сборке итогового ruleset'a.

Пример логики на стороне CLI (для понимания того, что делает Scirius под капотом):

```
# Обновить список источников и правила
...

sudo suricata-update

# Включить/выключить конкретные источники (пример)

sudo suricata-update enable-source mycompany-rules
sudo suricata-update disable-source et-pro
```

Разница в том, что Scirius позволяет управлять этим процессом через веб-интерфейс, видеть статус источников и результат обновлений, а также комбинировать их с локальными настройками без необходимости вручную редактировать конфиги или запускать команды на сенсоре.

...

Компиляция и деплой правил на сенсор Suricata

После того как источники обновлены и администратор выбрал нужные категории/правила, Scirius формирует результирующий набор и деплоит его в окружение Suricata.

Типичный pipeline:

1. Сбор правил:

Scirius объединяет правила из внешних источников и локальных файлов (`local.rules`, `override.rules` и др.), учитывая настройки включения/отключения.

2. Проверка и компиляция:

выполняется базовая проверка синтаксиса (чтобы исключить невалидные правила), возможно — разбиение по логическим наборам (`policy`, `malware`, `web-attacks` и т. д.).

3. Запись в `/etc/suricata/rules/`:

итоговые `.rules`-файлы помещаются в `/etc/suricata/rules/` (либо в структуры подкаталогов, если используется отдельное разделение по `policy/role`).

4. Перезапуск/перечитывание Suricata:

Scirius инициирует обновление правил на сенсоре — через `suricatactl` или `systemd`:

```
# Примерный шаг деплоя с точки зрения сенсора
...

sudo systemctl restart suricata.service

# либо (если поддерживается reload без полной перезагрузки)

sudo suricatactl reload-rules
```

В документации NICE.SSA обычно фиксируется, как часто и каким образом следует выполнять цикл «обновить □ протестировать □ развернуть» для правил — например, регулярные обновления внешних наборов по расписанию + ручной триггер деплоя после проверки на стенде.

...

Scirius превращает управление правилами Suricata из набора разрозненных скриптов и текстовых файлов в понятный процесс: источники подключаются, сигнатуры ищутся и анализируются, локальные правила документируются, а итоговый набор разворачивается на сенсор по воспроизводимому pipeline'у. Именно от этой «гигиены правил» зависит, насколько точным и полезным станет NICE.SSA в реальной эксплуатации.

...

::contentReference[oaicite:0]{index=0}

3.7. NGINX

NGINX в NICE.SSA — это единая точка входа ко всем пользовательским интерфейсам платформы. Он принимает HTTPS-подключения от операторов и аналитиков, терминирует TLS, выполняет аутентификацию (Basic Auth или интеграция с внешним IAM) и проксирует запросы к внутренним сервисам: OpenSearch Dashboards, EveBox, Scirius и Arkime.

...

Единая точка входа и маршрутизация UI

В типовой конфигурации NICE.SSA все интерфейсы доступны с одного доменного имени, например: `sensor.example.ru`. NGINX маршрутизирует запросы по URI:

- `https://sensor.example.ru/` — OpenSearch Dashboards (основной портал и дашборды);
- `https://sensor.example.ru/evebox` — EveBox (инциденты и алерты);
- `https://sensor.example.ru/scirius` — Scirius (управление правилами Suricata);
- `https://sensor.example.ru/arkime` — Arkime (PCAP и сессии).

Логика проста: снаружи существует один «вход в сенсор», изнутри — несколько приложений, каждое отвечает за свою задачу. Это:

- упрощает коммуникацию с пользователями (один URL вместо набора портов и адресов);
- упрощает управление сертификатами (один домен/набор SAN);
- позволяет централизованно внедрять политики доступа и логирования.

Упрощённый пример server-блока с маршрутизацией:

```
server {
listen 443 ssl http2;
server_name sensor.example.ru;

ssl_certificate /etc/nginx/tls/sensor.crt;
ssl_certificate_key /etc/nginx/tls/sensor.key;

# По умолчанию — OpenSearch Dashboards
location / {
    proxy_pass http://opensearch-dashboards:5601/;
    include /etc/nginx/snippets/proxy-common.conf;
}

location /evebox/ {
```

```

    proxy_pass http://evebox:5636/;
    include /etc/nginx/snippets/proxy-common.conf;
}

location /scirius/ {
    proxy_pass http://scirius:8008/;
    include /etc/nginx/snippets/proxy-common.conf;
}

location /arkime/ {
    proxy_pass http://arkime-ui:8005/;
    include /etc/nginx/snippets/proxy-common.conf;
}
...
}

```

...

HTTPS, Basic Auth и интеграция с Yandex IAM

NGINX в NICE.SSA отвечает за безопасность фронтального слоя:

- **HTTPS (TLS-терминация):**

на уровне NGINX настраиваются сертификаты (самоподписанные, корпоративные, доверенные CA), версии протокола TLS, наборы шифров, HSTS и другие параметры, отвечающие требованиям политики безопасности.

- **Basic Auth:**

самый простой и часто достаточный вариант аутентификации — базовая авторизация с использованием `htpasswd`. Она может применяться как ко всему серверу, так и к отдельным локациям (например, закрыть Arkime отдельным логином).

- **Интеграция с Yandex IAM / внешним SSO:**

возможно построение схемы, где NGINX работает совместно с внешним OIDC/OAuth2-прокси (например, отдельный сервис- «auth-gateway»), который проверяет токены Yandex Cloud или корпоративного IdP. После успешной проверки в NGINX приходят только запросы с валидным заголовком (например, `X-User`), а доступ для остальных блокируется.

Пример включения Basic Auth (схема, реальные пути могут отличаться):

```

server {
    listen 443 ssl http2;
    server_name sensor.example.ru;

```

```

ssl_certificate /etc/nginx/tls/sensor.crt;
ssl_certificate_key /etc/nginx/tls/sensor.key;

# Общая базовая аутентификация для всех UI
auth_basic "NICE.SSA restricted";
auth_basic_user_file /etc/nginx/htpasswd-nice-ssa;

location / {
    proxy_pass http://opensearch-dashboards:5601/;
    include /etc/nginx/snippets/proxy-common.conf;
}

location /evebox/ {
    proxy_pass http://evebox:5636/;
    include /etc/nginx/snippets/proxy-common.conf;
}

# При необходимости можно усилить правила по отдельным разделам
location /arkime/ {
    # Дополнительное ограничение по IP
    allow 10.0.0.0/24;
    deny all;

    proxy_pass http://arkime-ui:8005/;
    include /etc/nginx/snippets/proxy-common.conf;
}
'''
}

```

'''

Для интеграции с Yandex IAM или другим внешним SSO обычно используется связка:

- отдельный OAuth2/OIDC-прокси (docker-сервис в том же стеке NICE.SSA);
- NGINX, отправляющий неаутентифицированные запросы на этот прокси (через `auth_request`);
- передача в backend-интерфейсы информации о пользователе и его ролях через заголовки.

В результате правила доступа к сенсору заводятся в одном месте — в IAM/IdP организации, а NGINX выступает как технический `enforcement point`, который пропускает внутрь только аутентифицированных и авторизованных пользователей.

Эксплуатация, логирование и защита фронта

Помимо маршрутизации и аутентификации, NGINX в NICE.SSA выполняет ряд

эксплуатационных функций:

- **Централизованное логирование HTTP-доступа:**
access- и error-логи фиксируют, кто и когда обращался к Dashboards, EveBox, Scirius и Arkime, с каких IP, с какими кодами ответа. Эти логи можно дополнительно отправлять в OpenSearch/Vector или внешнюю систему логирования.
- **Ограничение запросов и rate limiting:**
защита UI от грубых переборов паролей и скриптовых атак за счёт ограничений на количество запросов с одного адреса.
- **Фильтрация по IP/сетям:**
возможность разрешить доступ к сенсору только с внутренних адресов, VPN-сегментов или jump-host'ов SOC.
- **Дополнительные заголовки безопасности:**
HSTS, X-Content-Type-Options, X-Frame-Options и другие заголовки, снижающие риск атак на уровне браузера.

В итоге NGINX становится «лицом» NICE.SSA в сети: через него проходят все соединения к интерфейсам анализа и управления, он же обеспечивает шифрование, проверку пользователей и базовую защиту от атак на веб-слой.

Внутри NICE.SSA множество сервисов — Suricata, Vector, OpenSearch, EveBox, Arkime, Scirius, Dashboards. Для пользователя же это один портал: <https://sensor.example.ru>. Эту иллюзию простоты и безопасности создаёт именно NGINX, аккуратно скрывая внутреннюю сложность стека.

...

```
::contentReference[oaicite:0]{index=0}
```