API ??? ????????? ???????? ????????? ??????????? ?
???????? ???????????

Для всех версий KEDR до версии 7.1.7

> *Примечание: данная статья является примером реализации скрипта, который позволяет делать API запрос к Central Node для сбора событий телеметрии и отправки полученных данных в коллектор SIEM по TCP.*
>
> *Справка на API: https://support.kaspersky.ru/kata/7.1/248949*

Шаг 1: копирование скрипта на внешнюю систему

Внешней системой может выступать любая рабочая станция или сервер, включая непосредственно коллектор SIEM. Заполните обязательные поля с UUID внешней системы, путями до TLS-сертификата и ключа, а также адресом и портом SIEM системы. Генерация UUID и TLS-сертификата с ключом указаны на шагах 2 и 3.

Сам скрипт:

```
import json
import logging
import os
import socket
import sys
import time
import urllib.parse
from pathlib import Path

import requests

# КОНФИГУРАЦИЯ — ОБЯЗАТЕЛЬНО ЗАМЕНИТЕ
EXTERNAL_SYSTEM_ID = "<идентификатор external_system_id>"  # Сгенерированный UUID
KATA_HOST = "<IP адрес или FQDN Central Node>"
KATA_PORT = 443  # Оставить без изменений
```

```
CERT_FILE = "<путь к файлу TLS-сертификата>"
KEY_FILE = "<путь к файлу закрытого ключа>"

SIEM_HOST = "<IP адрес или FQDN SIEM>"
SIEM_PORT = 9000  # порт TCP

# ПАРАМЕТРЫ ЗАПРОСА (опционально)
# MAX_EVENTS: от 1 до 72000 (максимум зависит от вашей инсталляции)
MAX_EVENTS = 20000

# FILTER: оставьте пустым для всех событий
# Примеры (см. список полей ниже):
# FILTER = "Ioa.Severity == 'High'"
# FILTER = "RemoteIp == '192.168.10.50'"
# FILTER = "Ioa.Rules.Techniques contains 'T1059'"
FILTER = ""

# MAX_TIMEOUT: максимум 300 секунд (PT300S). None — использовать значение по умолчанию.
MAX_TIMEOUT = None  # или 60, 120, 300

# НЕ МЕНЯЙТЕ НИЖЕ
API_URL = f"https://{KATA_HOST}:{KATA_PORT}/kata/events_api/v1/{EXTERNAL_SYSTEM_ID}/events"
TOKEN_FILE = "continuation_token.txt"
LOG_DIR = "logs"
LOG_FILE = f"{LOG_DIR}/kata_to_siem.log"

Path(LOG_DIR).mkdir(exist_ok=True)
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(message)s",
    handlers=[
        logging.FileHandler(LOG_FILE, encoding="utf-8"),
        logging.StreamHandler(sys.stdout)
    ]
)
logger = logging.getLogger("kata-to-siem")

requests.packages.urllib3.disable_warnings(requests.packages.urllib3.exceptions.InsecureReques
tWarning)
```

```
def send_to_siem(events: list) -> bool:
    if not events:
        return True
    try:
        data = "\n".join(json.dumps(e, ensure_ascii=False) for e in events) + "\n"
        with socket.create_connection((SIEM_HOST, SIEM_PORT), timeout=15) as sock:
            sock.sendall(data.encode("utf-8"))
#ДЛЯ ТЕСТА ЗАКОММЕНТИРОВАТЬ ВЕРХНИЕ СТРОКИ ДО try И РАСКОММЕНТИРОВАТЬ ЭТИ ДЛЯ ЗАПИСИ В ФАЙЛ
(укажите корректный путь):
#        with open("/home/admin/kata_events.ndjson", "a", encoding="utf-8") as f:
#            for e in events:
#                f.write(json.dumps(e, ensure_ascii=False) + "\n")

        logger.info(f"Отправлено {len(events)} событий в SIEM")
        return True
    except Exception as e:
        logger.error(f"Ошибка отправки в SIEM: {e}")
        return False


def fetch_events(token: str | None) -> dict | None:
    params = {"max_events": str(MAX_EVENTS)}

    if token is None:
        if FILTER:
            params["filter"] = FILTER
        if MAX_TIMEOUT is not None:
            safe_timeout = min(MAX_TIMEOUT, 300)
            params["max_timeout"] = f"PT{safe_timeout}S"

    if token:
        params["continuation_token"] = token

    try:
        response = requests.get(
            API_URL,
            params=params,
            cert=(CERT_FILE, KEY_FILE),
            verify=False,
```

```
            timeout=310
        )
        if response.status_code == 404 and "Subscription not found" in response.text:
            logger.error("Токен устарел или подпись удалена. Сброс токена.")
            return {"reset_token": True}

        if response.status_code == 200:
            return response.json()
        else:
            logger.error(f"API error {response.status_code}: {response.text[:300]}")
            return None

    except Exception as e:
        logger.exception(f"Ошибка при запросе к KATA API: {e}")
        return None


def load_token() -> str | None:
    if os.path.exists(TOKEN_FILE):
        with open(TOKEN_FILE, "r", encoding="utf-8") as f:
            return f.read().strip()
    return None


def save_token(token: str):
    with open(TOKEN_FILE, "w", encoding="utf-8") as f:
        f.write(token)
    logger.info("Токен сохранён")


def main():
    logger.info("Запуск интеграции KATA → SIEM")
    if FILTER:
        logger.info(f"Фильтр: {FILTER}")
    if MAX_TIMEOUT:
        logger.info(f"Макс. время ожидания: {MAX_TIMEOUT} сек")

    token = load_token()

    while True:
```

```
        try:
            result = fetch_events(token)
            if result and result.get("reset_token"):
                logger.info("Сброс токена. Начинаем с начала.")
                token = None
                if os.path.exists(TOKEN_FILE):
                    os.remove(TOKEN_FILE)
                time.sleep(5)
                continue

            events = result.get("events", [])
            new_token = result.get("continuationToken")

            if not events:
                logger.info("Очередь пуста. Ожидание...")
                time.sleep(30)
                continue

            logger.info(f"Получено {len(events)} событий")

            if send_to_siem(events):
                if new_token:
                    save_token(new_token)
                    token = new_token
                else:
                    logger.warning("continuationToken отсутствует в ответе")
            else:
                logger.warning("События не отправлены — токен НЕ обновлён")

            time.sleep(1 if len(events) == MAX_EVENTS else 5)

        except KeyboardInterrupt:
            logger.info("Остановка по запросу пользователя")
            sys.exit(0)
        except Exception as e:
            logger.exception(f"Критическая ошибка: {e}")
            time.sleep(10)


if __name__ == "__main__":
```

```
    main()
```

Шаг 2: получение UUID внешней системы

На **Windows** откройте PowerShell и выполните:

```
[guid]::NewGuid().ToString()
```

На **Linux** выполните в терминале:

```
cat /proc/sys/kernel/random/uuid
```

или

```
python3 -c "import uuid; print(uuid.uuid4())"
```

Шаг 3: генерация самоподписанного TLS-сертификата и ключа

Предварительно установив OpenSSL выполнить команду:

```
openssl req -x509 -newkey rsa:2048 -keyout kata.key -out kata.crt -days 365 -nodes
```

Шаг 4: тестирование и дальнейшее использование

Для тестирования, запустите скрипт вручную, выполнив **python3 /path/to/script.py**

После первого запуска необходимо будет перейти в веб-интерфейс Central Node под учетной записью с правами Администратора в раздел Внешние системы и подтвердить подключение. Далее повторить запуск скрипта.

По завершению тестирования можно создать минималистичный сервис на внешней системы, чтобы скрипт запускался после загрузки системы. Для этого в **/etc/systemd/system/** создайте **kata-siem.service** со следующим содержимым (отредактировать под себя, изменив пользователя, рабочий каталог и параметры запуска):

```
[Unit]
Description=KATA to SIEM Forwarder
After=network.target

[Service]
Type=simple
User=root
```

```
WorkingDirectory=/opt/kata-siem
ExecStart=/usr/bin/python3 /opt/kata-siem/kata_to_siem.py
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal

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

Далее выполнить:

```
sudo systemctl daemon-reload
sudo systemctl enable --now kata-siem
```

Revision #3
Created 5 November 2025 08:58:51 by Кирилл
Updated 15 April 2026 07:26:43 by Кирилл