Программный комплекс обнаружения и реагирования на мошеннические голосовые вызовы на основе анализа SIP/RTP-трафика с применением технологий машинного обучения «Волна»

Руководство администратора безопасности

МОСКВА, 2026

Содержание

1. Общие сведения о ПО

1.1 Описание платформы

ПО «Программный комплекс обнаружения и реагирования на мошеннические голосовые вызовы на основе анализа SIP/RTP-трафика с применением технологий машинного обучения «Волна»» (далее - Платформа "Волна") представляет собой систему обнаружения и предотвращения мошеннических звонков в режиме реального времени. Платформа предназначена для анализа VoIP-трафика с использованием технологий машинного обучения для выявления социальной инженерии и голосового мошенничества (vishing).

1.2 Компоненты безопасности платформы

Платформа "Волна" использует комплексный подход к обеспечению безопасности:

1.2.1 HashiCorp Vault

HashiCorp Vault является центральным компонентом системы управления секретами платформы. Vault обеспечивает:

- Безопасное хранение паролей, ключей API и других конфиденциальных данных
- Динамическое управление секретами через API
- Аудит всех операций с секретами
- Ротацию секретов без перезапуска приложений
- Шифрование данных в покое и при передаче

Интеграция Vault с компонентами платформы:

- Все микросервисы (CDR, Classifier, Notificator, Importer) используют Vault для получения конфигурации и секретов
- Аутентификация через AppRole механизм
- Автоматическая синхронизация настроек при изменении секретов

1.2.2 Аутентификация и авторизация

- **JWT-токены** с алгоритмом HS256 для аутентификации пользователей
- Срок действия токенов: 5 минут (настраиваемый)

- Ролевая модель доступа (read, write, admin, export, audio_access, critical)
- Хеширование паролей алгоритмом bcrypt

1.2.3 Шифрование и защита данных

- Интеграция с HashiCorp Vault для секретов
- Шифрование конфиденциальной информации
- CORS-защита для веб-интерфейса
- TLS/SSL для всех внешних соединений

1.2.4 Ограничение частоты запросов

- Алгоритм скользящего окна для защиты от DDoS-атак
- Различные лимиты для разных эндпоинтов
- Защита от brute-force атак

1.3 Роли и ответственность администратора безопасности

Администратор безопасности платформы "Волна" отвечает за:

1. **Управление секретами:**
    - Настройка и поддержка HashiCorp Vault
    - Ротация паролей и секретов согласно политике
    - Управление политиками доступа к секретам
2. **Мониторинг безопасности:**
    - Отслеживание подозрительной активности
    - Анализ логов доступа и аудита
    - Реагирование на инциденты безопасности
3. **Обновление компонентов:**
    - Оценка безопасности обновлений
    - Безопасное развертывание обновлений
    - Проверка целостности после обновления
4. **Реагирование на инциденты:**
    - Обнаружение компрометации
    - Изоляция затронутых компонентов
    - Восстановление работоспособности
    - Постмортем анализ
5. **Аудит и соответствие:**
    - Регулярный аудит безопасности
    - Документирование процедур безопасности

- Обеспечение соответствия требованиям безопасности

1.4.Требования к квалификации для демонстрации

Учитывая особенности заявляемой Платформы «Волна», требования к установке и эксплуатации, а также возможность оказания сервисной поддержки исключительно силами компетентных специалистов разработчика и правообладателя, предлагаем провести демонстрацию функциональных характеристик, системы хранения исходного кода и процессов эксплуатации программного обеспечения в формате видеоконференцсвязи.

Демонстрация может быть организована по согласованию с экспертом Минцифры в рабочем порядке, в любое удобное время и с использованием любого подходящего технического решения для ВКС.

Контактные данные ответственного лица за организацию демонстрации программного обеспечения:
(ФИО, должность, телефон, e-mail).

---

2. Обновление компонентов ПО с учетом безопасности

2.1 Подготовка к обновлению

Перед началом обновления компонентов платформы необходимо выполнить комплексную проверку безопасности:

2.1.1 Оценка безопасности обновления

```
# 1. Проверка changelog на наличие исправлений безопасности
# Изучите release notes новой версии на предмет:
# - Исправленных уязвимостей (CVE)
# - Изменений в механизмах безопасности
# - Несовместимых изменений в API

# 2. Проверка подписи образов Docker
docker trust inspect registry.af.internal/antifraud/cloud/cdr:v1.2.0

# 3. Проверка целостности образов
docker pull
```

```
registry.af.internal/antifraud/cloud/cdr:v1.2.0
docker images --digests | grep cdr
```

2.1.2 Резервное копирование перед обновлением

Критически важно создать резервные копии всех компонентов безопасности:

```
# 1. Резервное копирование Vault
export VAULT_ADDR='https://vault.af.internal'
export VAULT_TOKEN='your-root-token'
vault operator export -format=json >
/backup/vault/vault_backup_$(date +%Y%m%d_%H%M%S).json

# Шифрование резервной копии
gpg --encrypt --recipient security@example.com
/backup/vault/vault_backup_*.json

# 2. Резервное копирование конфигурации безопасности
tar -czf /backup/config/security_config_$(date
+%Y%m%d).tar.gz \
  /opt/demo/config/docker-compose.stack.yaml \
  /opt/demo/config/stack.env \
  /etc/vault.d/

# 3. Резервное копирование сертификатов и ключей
tar -czf /backup/certs/certificates_$(date
+%Y%m%d).tar.gz \
  /etc/ssl/certs/demo.* \
  /etc/ssl/private/demo.*
```

2.1.3 Проверка текущего состояния безопасности

```
# 1. Проверка статуса всех сервисов
docker stack services demo

# 2. Проверка активных подключений к Vault
vault list auth/token/accessors | head -20

# 3. Проверка логов на подозрительную активность
docker service logs demo_cdr --since 24h | grep -i
```

```
"unauthorized\|forbidden\|error"

# 4. Проверка актуальности секретов
vault kv get demo/settings | grep -i
"password\|secret\|key"
```

2.2 Безопасное обновление сервисов в Docker Swarm

2.2.1 Обновление с проверкой целостности

```
# 1. Обновление образа с указанием конкретной версии
(не latest!)
docker service update \
  --image
registry.af.internal/antifraud/cloud/cdr:v1.2.0 \
  --update-delay 30s \
  --update-parallelism 1 \
  --update-failure-action rollback \
  demo_cdr

# 2. Мониторинг процесса обновления
watch -n 5 'docker service ps demo_cdr --no-trunc'

# 3. Проверка healthcheck после обновления
for i in {1..12}; do
  curl -f http://localhost:8800/healthcheck && break
  sleep 5
done
```

2.2.2 Обновление через docker-compose.stack.yaml

Рекомендуемый способ для комплексных обновлений:

```
# 1. Создание резервной копии текущей конфигурации
cp docker-compose.stack.yaml docker-
compose.stack.yaml.backup_$(date +%Y%m%d)

# 2. Обновление версий образов в файле
# Замените все :latest на конкретные версии, например:
# image:
registry.af.internal/antifraud/cloud/cdr:v1.2.0
```

```
# 3. Проверка синтаксиса файла
docker-compose -f docker-compose.stack.yaml config >
/dev/null

# 4. Развертывание обновленного стека
docker stack deploy -c docker-compose.stack.yaml demo

# 5. Проверка статуса обновления
docker stack ps demo --no-trunc
```

2.2.3 Проверка безопасности после обновления

После обновления необходимо проверить:

```
# 1. Проверка подключения к Vault
docker exec -it $(docker ps -q -f name=demo_cdr) \
  vault read -format=json demo/settings

# 2. Проверка аутентификации сервисов
docker service logs demo_cdr --tail 50 | grep -i vault

# 3. Проверка работы healthcheck эндпоинтов
curl -f http://localhost:8800/healthcheck
curl -f http://localhost:6061/healthcheck
curl -f http://localhost:8803/healthcheck
curl -f http://localhost:6062/healthcheck

# 4. Проверка метрик безопасности в Prometheus
curl
'http://localhost:9090/api/v1/query?query=up{job=~"demo
-.*"}'

# 5. Проверка отсутствия ошибок в логах
docker service logs demo_cdr --since 10m | grep -i
"error\|exception\|failed"
```

2.3 Откат изменений при обнаружении уязвимостей

В случае обнаружения проблем безопасности после обновления:

```
# 1. Немедленный откат к предыдущей версии
docker service rollback demo_cdr

# 2. Или явное указание предыдущей безопасной версии
docker service update \
  --image
registry.af.internal/antifraud/cloud/cdr:v1.1.0 \
  demo_cdr

# 3. Проверка статуса отката
docker service ps demo_cdr --no-trunc

# 4. Изоляция обновленного компонента (если требуется)
docker service update --replicas 0 demo_cdr

# 5. Документирование инцидента
# Зафиксируйте:
# - Версию с уязвимостью
# - Обнаруженную проблему
# - Время обнаружения
# - Выполненные действия
```

2.4 Обновление инфраструктурных компонентов

2.4.1 Обновление HashiCorp Vault

Критически важно: Обновление Vault требует особой осторожности из-за центральной роли в безопасности платформы.

```
# 1. Проверка совместимости версий
# Убедитесь, что новая версия Vault совместима с
используемыми клиентами

# 2. Резервное копирование Vault (обязательно!)
vault operator export -format=json >
vault_backup_before_update_$(date +%Y%m%d).json
gpg --encrypt --recipient security@example.com
vault_backup_before_update_*.json

# 3. Проверка текущей версии
vault version
```

```
# 4. Обновление Vault (зависит от способа развертывания)
# Если Vault в Docker:
docker service update --image hashicorp/vault:1.15.0 demo_vault

# 5. Проверка работоспособности после обновления
vault status
vault kv get demo/settings

# 6. Проверка подключения сервисов к обновленному Vault
docker service logs demo_cdr --tail 100 | grep -i vault
```

2.4.2 Обновление ClickHouse

```
# 1. Резервное копирование данных
docker exec clickhouse clickhouse-client --query "
BACKUP DATABASE cdr TO Disk('backups',
'backup_before_update_$(date +%Y%m%d).zip')"

# 2. Обновление образа
docker service update --image clickhouse/clickhouse-server:23.8 demo_clickhouse

# 3. Проверка версии после обновления
docker exec clickhouse clickhouse-client --query "SELECT version()"

# 4. Проверка целостности данных
docker exec clickhouse clickhouse-client --query "SELECT count() FROM cdr.calls"
```

2.4.3 Обновление Redpanda/Kafka

```
# 1. Проверка статуса кластера перед обновлением
docker exec redpanda rpk cluster info

# 2. Обновление образа
docker service update --image
```

```
docker.redpanda.com/redpandadata/redpanda:v23.2.0
demo_redpanda

# 3. Проверка статуса после обновления
docker exec redpanda rpk cluster health
docker exec redpanda rpk cluster info
```

2.5 Контрольный список безопасного обновления

Перед обновлением:

- ✓ Изучены release notes на предмет исправлений безопасности
- ✓ Проверена подпись и целостность образов
- ✓ Созданы резервные копии Vault, конфигурации и сертификатов
- ✓ Проверено текущее состояние безопасности
- ✓ Запланировано окно обслуживания
- ✓ Уведомлены заинтересованные стороны

Во время обновления:

- ✓ Обновление выполняется с указанием конкретных версий (не latest)
- ✓ Используется стратегия rolling update
- ✓ Мониторится процесс обновления
- ✓ Проверяются healthcheck эндпоинты

После обновления:

- ✓ Проверено подключение к Vault
- ✓ Проверена аутентификация всех сервисов
- ✓ Проверены healthcheck эндпоинты
- ✓ Проверены метрики в системе мониторинга
- ✓ Проверены логи на наличие ошибок
- ✓ Выполнено тестирование основных функций
- ✓ Документированы изменения

3. Диагностика и локализация проблем Vault

3.1 Мониторинг состояния Vault

3.1.1 Проверка статуса Vault

```
# Базовая проверка статуса
vault status
```

```
# Ожидаемый вывод должен содержать:
# - Sealed: false (Vault должен быть unsealed)
# - Version: версия Vault
# - Cluster Name: имя кластера
# - Cluster ID: идентификатор кластера
```

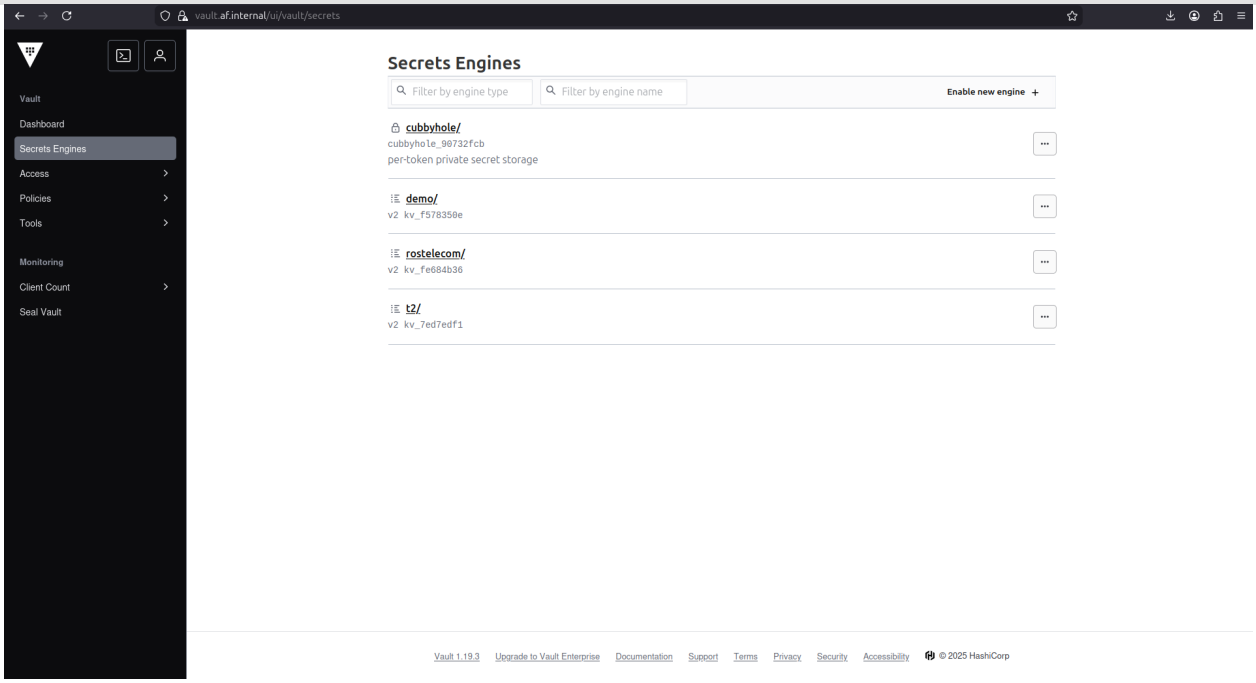

Рисунок 1 – Vault. Хранилище секретов

3.1.2 Проверка доступности Vault

```
# Проверка доступности через HTTP/HTTPS
curl -k ${VAULT_ADDR}/v1/sys/health

# Ожидаемый ответ:
#
{"initialized":true,"sealed":false,"standby":false,"ver
sion":"1.15.0",...}

# Проверка доступности из контейнеров сервисов
docker exec -it $(docker ps -q -f name=demo_cdr) \
  curl -k ${VAULT_ADDR}/v1/sys/health
```

3.1.3 Мониторинг метрик Vault

```
# Просмотр метрик производительности
curl ${VAULT_ADDR}/v1/sys/metrics?format=prometheus
```

```
# Ключевые метрики для мониторинга:
# - vault_core_unsealed - статус unsealed (должен быть 1)
# - vault_token_count - количество активных токенов
# - vault_audit_log_request_count - количество запросов аудита
# - vault_expire_num_leases - количество истекающих lease
```

3.2 Диагностика проблем подключения

3.2.1 Проблемы с сетевой доступностью

Симптомы:

- Сервисы не могут подключиться к Vault
- Таймауты при обращении к Vault
- Ошибки "connection refused" или "no route to host"

Диагностика:

```
# 1. Проверка доступности Vault из хоста
curl -v -k ${VAULT_ADDR}/v1/sys/health

# 2. Проверка доступности из контейнера сервиса
docker exec -it $(docker ps -q -f name=demo_cdr) \
  curl -v -k ${VAULT_ADDR}/v1/sys/health

# 3. Проверка DNS разрешения
docker exec -it $(docker ps -q -f name=demo_cdr) \
  nslookup vault.af.internal

# 4. Проверка сетевых подключений
docker exec -it $(docker ps -q -f name=demo_cdr) \
  nc -zv vault.af.internal 8200

# 5. Проверка сетей Docker
docker network inspect demo-backend-network | grep -A 10 vault
```

Решение:

```
# Если Vault недоступен из контейнеров:
# 1. Проверьте, что Vault находится в той же сети
```

```
docker network inspect demo-backend-network

# 2. Переподключите Vault к сети
docker service update --network-add demo-backend-
network demo_vault

# 3. Перезапустите сервисы для применения изменений
docker service update --force demo_cdr
```

3.2.2 Проблемы с SSL/TLS сертификатами

Симптомы:

- Ошибки "certificate verify failed"
- Ошибки "self-signed certificate"
- Ошибки "hostname mismatch"

Диагностика:

```
# 1. Проверка сертификата Vault
openssl s_client -connect vault.af.internal:8200 -
showcerts

# 2. Проверка переменной VAULT_SKIP_VERIFY
docker service inspect demo_cdr --pretty | grep
VAULT_SKIP_VERIFY

# 3. Проверка CA сертификата в контейнере
docker exec -it $(docker ps -q -f name=demo_cdr) \
  ls -la /etc/ssl/certs/ | grep vault
```

Решение:

```
# Временное решение (только для тестирования):
docker service update \
  --env-add VAULT_SKIP_VERIFY=true \
  demo_cdr

# Правильное решение - добавить CA сертификат в
контейнеры:
# 1. Скопировать CA сертификат
docker cp /path/to/vault-ca.crt
demo_cdr:/etc/ssl/certs/vault-ca.crt
```

```
# 2. Обновить переменные окружения
docker service update \
  --env-add VAULT_CACERT=/etc/ssl/certs/vault-ca.crt \
  demo_cdr
```

3.3 Анализ логов Vault

3.3.1 Просмотр логов Vault

```
# Если Vault в Docker контейнере:
docker logs vault-container-id --tail 100 -f

# Если Vault как системный сервис:
journalctl -u vault -f

# Просмотр логов через Docker Swarm:
docker service logs demo_vault --tail 100 -f
```

3.3.2 Анализ типичных ошибок

Ошибка: "permission denied"

```
# Проверка политик доступа
vault policy read antifraud-policy

# Проверка токена сервиса
vault token lookup <token>

# Проверка AppRole настроек
vault read auth/approle/role/antifraud
```

Ошибка: "no such path"

```
# Проверка существования пути
vault kv list demo/

# Проверка mount point
vault secrets list | grep demo

# Проверка правильности пути в конфигурации
docker service inspect demo_cdr --pretty | grep
VAULT_SECRET_PATH
```

Ошибка: "token expired"

```
# Проверка TTL токена
vault token lookup

# Проверка настроек AppRole
vault read auth/approle/role/antifraud | grep -i ttl

# Решение: обновить Secret ID
vault write -f auth/approle/role/antifraud/secret-id
```

3.4 Проблемы с AppRole и токенами

3.4.1 Диагностика проблем AppRole

Проверка конфигурации AppRole:

```
# 1. Проверка существования AppRole
vault list auth/approle/role

# 2. Просмотр настроек AppRole
vault read auth/approle/role/antifraud

# 3. Проверка Role ID
vault read auth/approle/role/antifraud/role-id

# 4. Проверка Secret ID (требует root токен)
vault list auth/approle/role/antifraud/secret-id
```

3.4.2 Проблемы с аутентификацией

Симптомы:

- Сервисы не могут аутентифицироваться в Vault
- Ошибки **"invalid role_id"** или **"invalid secret_id"**
- Ошибки **"permission denied"** после аутентификации

Диагностика:

```
# 1. Проверка переменных окружения в сервисе
docker service inspect demo_cdr --pretty | grep -A 5
VAULT_ROLE_ID
docker service inspect demo_cdr --pretty | grep -A 5
VAULT_SECRET_ID

# 2. Тестовая аутентификация с Role ID и Secret ID
```

```
export VAULT_ROLE_ID=$(docker service inspect demo_cdr
--format '{{range
.Spec.TaskTemplate.ContainerSpec.Env}}{{if eq .Name
"VAULT_ROLE_ID"}}{{.Value}}{{end}}{{end}}')
export VAULT_SECRET_ID=$(docker service inspect
demo_cdr --format '{{range
.Spec.TaskTemplate.ContainerSpec.Env}}{{if eq .Name
"VAULT_SECRET_ID"}}{{.Value}}{{end}}{{end}}')

vault write auth/approle/login \
  role_id=${VAULT_ROLE_ID} \
  secret_id=${VAULT_SECRET_ID}

# 3. Проверка полученного токена
export VAULT_TOKEN=$(vault write -field=token
auth/approle/login role_id=${VAULT_ROLE_ID}
secret_id=${VAULT_SECRET_ID})
vault token lookup
```

Решение:

```
# Если Role ID или Secret ID неверны:
# 1. Получить правильный Role ID
vault read auth/approle/role/antifraud/role-id

# 2. Сгенерировать новый Secret ID
vault write -f auth/approle/role/antifraud/secret-id

# 3. Обновить переменные окружения в сервисе
docker service update \
  --env-add VAULT_ROLE_ID=<correct-role-id> \
  --env-add VAULT_SECRET_ID=<new-secret-id> \
  demo_cdr
```

3.4.3 Проблемы с политиками доступа

Диагностика:

```
# 1. Проверка политики, привязанной к AppRole
vault read auth/approle/role/antifraud | grep
token_policies
```

```
# 2. Просмотр содержимого политики
vault policy read antifraud-policy

# 3. Тест доступа с токеном сервиса
export VAULT_TOKEN=<service-token>
vault kv get demo/settings
```

Решение:

```
# Если политика не предоставляет необходимые права:
# 1. Обновить политику
vault policy write antifraud-policy - <<EOF
path "demo/settings" {
  capabilities = ["read"]
}
path "demo/settings/*" {
  capabilities = ["read"]
}
EOF

# 2. Применить политику к AppRole
vault write auth/approle/role/antifraud \
  token_policies="antifraud-policy"

# 3. Перезапустить сервисы для получения нового токена
docker service update --force demo_cdr
```

3.5 Восстановление работоспособности Vault

3.5.1 Vault запечатан (Sealed)

Симптомы:

- Все запросы к Vault возвращают ошибку **"Vault is sealed"**
- Сервисы не могут получить секреты

Решение:

```
# 1. Распечатывание Vault (требует unseal ключи)
# Обычно требуется 3 из 5 unseal ключей
vault operator unseal <unseal-key-1>
vault operator unseal <unseal-key-2>
vault operator unseal <unseal-key-3>
```

```
# 2. Проверка статуса после распечатывания
vault status

# 3. Автоматическое распечатывание (если настроено)
# Проверьте наличие auto-unseal конфигурации
```

3.5.2 Восстановление из резервной копии

```
# 1. Остановка сервисов, использующих Vault
docker service scale demo_cdr=0
docker service scale demo_notificator=0
docker service scale demo_classifier=0
docker service scale demo_importer=0

# 2. Восстановление из резервной копии
vault operator import vault_backup_20250101.json

# 3. Распечатывание Vault
vault operator unseal <unseal-keys>

# 4. Проверка восстановленных секретов
vault kv get demo/settings

# 5. Запуск сервисов
docker service scale demo_cdr=3
docker service scale demo_notificator=2
docker service scale demo_classifier=1
docker service scale demo_importer=2
```

3.5.3 Пересоздание AppRole

Если AppRole был случайно удален или поврежден:

```
# 1. Включение AppRole (если отключен)
vault auth enable approle

# 2. Создание политики
vault policy write antifraud-policy - <<EOF
path "demo/settings" {
  capabilities = ["read"]
```

```
}
path "demo/settings/*" {
  capabilities = ["read"]
}
EOF

# 3. Создание AppRole
vault write auth/approle/role/antifraud \
  token_policies="antifraud-policy" \
  token_ttl=1h \
  token_max_ttl=4h \
  bind_secret_id=true

# 4. Получение Role ID
vault read auth/approle/role/antifraud/role-id

# 5. Генерация Secret ID
vault write -f auth/approle/role/antifraud/secret-id

# 6. Обновление переменных окружения в сервисах
docker service update \
  --env-add VAULT_ROLE_ID=<role-id> \
  --env-add VAULT_SECRET_ID=<secret-id> \
  demo_cdr
```

3.6 Контрольный список диагностики Vault

При проблемах с подключением:

- Проверен статус Vault (vault status)
- ✓ Проверена доступность через HTTP/HTTPS
- ✓ Проверена сетевая доступность из контейнеров
- ✓ Проверены SSL/TLS сертификаты
- ✓ Проверены переменные окружения VAULT_ADDR

При проблемах с аутентификацией:

- ✓ Проверены Role ID и Secret ID
- ✓ Проверена конфигурация AppRole
- ✓ Проверены политики доступа
- ✓ Проверен срок действия токенов
- ✓ Проверены логи Vault на ошибки

При проблемах с доступом к секретам:

- ✓ Проверено существование пути к секретам
- ✓ Проверены права доступа в политике
- ✓ Проверена правильность mount point
- ✓ Проверены логи доступа

4. Парольная политика и работа с секретами

4.1 Парольная политика платформы

Платформа "Волна" использует строгую парольную политику для обеспечения безопасности всех компонентов системы.

4.1.1 Общие требования к паролям

Минимальные требования:

- Длина пароля: не менее 16 символов
- Обязательное использование заглавных и строчных букв
- Обязательное использование цифр
- Обязательное использование специальных символов (!@#$%^&*()_+-=[]{}|;:,.<>?)
- Запрет использования словарных слов и простых последовательностей
- Запрет повторного использования последних 5 паролей

Рекомендации:

- Использование генераторов случайных паролей
- Хранение паролей только в Vault
- Никогда не передавать пароли по незащищенным каналам связи
- Использование уникальных паролей для каждого компонента

4.1.2 Парольная политика для различных компонентов

Базы данных (ClickHouse, PostgreSQL):

- Минимальная длина: **20 символов**
- Обязательная ротация: **каждые 90 дней**
- Использование случайных паролей, сгенерированных криптографически стойким генератором

API ключи и токены:

- Минимальная длина: 32 символа
- Обязательная ротация: каждые 180 дней

- Использование UUID v4 или криптографически стойких случайных строк

AppRole Secret ID (Vault):

- Автоматическая генерация Vault
- Обязательная ротация: каждые 30 дней
- Немедленная ротация при подозрении на компрометацию

Пароли пользователей системы:

- Минимальная длина: 16 символов
- Обязательная ротация: каждые 90 дней
- Использование bcrypt хеширования (cost factor: 12)

4.2 Требования к сложности паролей

4.2.1 Генерация безопасных паролей

```
# Генерация пароля с использованием OpenSSL (рекомендуется)
openssl rand -base64 32

# Генерация пароля с использованием /dev/urandom
cat /dev/urandom | tr -dc 'a-zA-Z0-9!@#$%^&*()_+-=[]{}|;:,.<>?' | fold -w 32 | head -n 1

# Генерация пароля через Vault
vault write -field=password sys/generate-password/32 format=base64

# Генерация пароля для базы данных (20+ символов)
openssl rand -base64 24
```

4.2.2 Проверка сложности пароля

```
#!/bin/bash
# Скрипт проверки сложности пароля
PASSWORD=$1

if [ ${#PASSWORD} -lt 16 ]; then
  echo "ОШИБКА: Пароль слишком короткий (минимум 16 символов)"
  exit 1
fi
```

```
if ! [[ $PASSWORD =~ [A-Z] ]]; then
  echo "ОШИБКА: Пароль должен содержать заглавные
буквы"
  exit 1
fi

if ! [[ $PASSWORD =~ [a-z] ]]; then
  echo "ОШИБКА: Пароль должен содержать строчные буквы"
  exit 1
fi

if ! [[ $PASSWORD =~ [0-9] ]]; then
  echo "ОШИБКА: Пароль должен содержать цифры"
  exit 1
fi

if ! [[ $PASSWORD =~ [!@#$%^&*()_+\-=\[\]{}|;:,.\<\>?]
]]; then
  echo "ОШИБКА: Пароль должен содержать специальные
символы"
  exit 1
fi

echo "Пароль соответствует требованиям безопасности"
```

4.3 Ротация паролей и секретов

4.3.1 Политика ротации секретов

Рекомендуемая частота ротации:

- Пароли баз данных: каждые 90 дней
- API ключи: каждые 180 дней
- Secret ID для AppRole: каждые 30 дней
- TLS сертификаты: согласно сроку действия (обычно 365 дней)
- Пароли пользователей: каждые 90 дней
- Root токены Vault: немедленно после использования, затем уничтожение

4.3.2 Процедура ротации пароля ClickHouse

```
#!/bin/bash
# Скрипт безопасной ротации пароля ClickHouse

set -e

# 1. Генерация нового пароля
NEW_PASSWORD=$(openssl rand -base64 24)
echo "Сгенерирован новый пароль для ClickHouse"

# 2. Обновление пароля в ClickHouse
docker exec -it clickhouse clickhouse-client --query "
ALTER USER admin IDENTIFIED BY '${NEW_PASSWORD}'"

# 3. Обновление пароля в Vault
export VAULT_ADDR='https://vault.af.internal'
export VAULT_TOKEN='your-admin-token'

vault kv put demo/settings
clickhouse_password="${NEW_PASSWORD}"

# 4. Синхронизация настроек в сервисах
echo "Синхронизация настроек в сервисах..."
curl -X PUT http://localhost:8800/settings/sync
curl -X PUT http://localhost:8803/settings/sync
curl -X PUT http://localhost:6061/settings/sync

# 5. Проверка работоспособности
echo "Проверка работоспособности..."
curl -f http://localhost:8800/healthcheck || exit 1
curl -f http://localhost:8803/healthcheck || exit 1

# 6. Удаление пароля из переменных окружения
unset NEW_PASSWORD

echo "Ротация пароля ClickHouse завершена успешно"
```

4.3.3 Процедура ротации Secret ID для AppRole

```
#!/bin/bash
# Скрипт ротации Secret ID для AppRole

set -e

export VAULT_ADDR='https://vault.af.internal'
export VAULT_TOKEN='your-admin-token'

# 1. Генерация нового Secret ID
NEW_SECRET_ID=$(vault write -field=secret_id -f
auth/approle/role/antifraud/secret-id)
echo "Сгенерирован новый Secret ID"

# 2. Обновление Secret ID в сервисах (постепенно, для
минимизации downtime)
SERVICES=("demo_cdr" "demo_notificator"
"demo_classifier" "demo_importer")

for service in "${SERVICES[@]}"; do
  echo "Обновление Secret ID в сервисе ${service}..."

  # Получение текущего Role ID
  ROLE_ID=$(docker service inspect ${service} --format
\
    '{{range .Spec.TaskTemplate.ContainerSpec.Env}}{{if
eq .Name "VAULT_ROLE_ID"}}{{.Value}}{{end}}{{end}}')

  # Обновление Secret ID
  docker service update \
    --env-add VAULT_SECRET_ID=${NEW_SECRET_ID} \
    ${service}

  # Ожидание готовности сервиса
  sleep 10

  # Проверка работоспособности
```

```
  docker service ps ${service} --no-trunc | grep -q
"Running" || {
    echo "ОШИБКА: Сервис ${service} не запустился после
обновления"
    exit 1
  }
done

echo "Ротация Secret ID завершена успешно"
```

4.3.4 Автоматизация ротации секретов

Настройка cron для автоматической ротации:

```
# Добавить в crontab для автоматической ротации Secret
ID (каждые 30 дней)
0 2 1 * * /opt/scripts/rotate_vault_secret_id.sh >>
/var/log/vault_rotation.log 2>&1

# Добавить в crontab для автоматической ротации паролей
БД (каждые 90 дней)
0 3 1 */3 * /opt/scripts/rotate_db_passwords.sh >>
/var/log/db_rotation.log 2>&1
```

4.4 Управление секретами через Vault

4.4.1 Хранение секретов в Vault

Рисунок 2 - Управление секретами в Vault

Базовые операции с секретами:

```
# Просмотр секрета
vault kv get demo/settings

# Создание/обновление секрета
vault kv put demo/settings \
    kafka_url="redpanda:9092" \
    clickhouse_host="clickhouse" \
    clickhouse_port="8123" \
    clickhouse_database="cdr" \
    clickhouse_user="admin" \
    clickhouse_password="secure_password_here" \
    s3_endpoint="minio:9000" \
    s3_access_key="minioadmin" \
    s3_secret_key="minioadmin"

# Обновление отдельного поля секрета
vault kv patch demo/settings
clickhouse_password="new_password"

# Удаление секрета (осторожно!)
vault kv delete demo/settings

# Просмотр всех секретов в пути
vault kv list demo/
```

4.4.2 Структура хранения секретов

Рекомендуемая структура:

```
demo/
├── settings/                # Основные настройки
подключений
│   ├── kafka_url
│   ├── clickhouse_*
│   ├── s3_*
│   └── ...
├── api_keys/                # API ключи внешних сервисов
│   ├── telegram_bot_token
│   ├── smtp_password
```

```
│   └── ...
├── certificates/          # TLS сертификаты
│   ├── demo_cert
│   ├── demo_key
│   └── ca_cert
└── database/              # Пароли баз данных
    ├── clickhouse_admin
    ├── postgres_admin
    └── ...
```

4.4.3 Версионирование секретов

Vault автоматически версионирует секреты в KV v2:

```
# Просмотр всех версий секрета
vault kv get -versions demo/settings

# Просмотр конкретной версии
vault kv get -version=5 demo/settings

# Восстановление предыдущей версии
vault kv rollback -version=5 demo/settings
```

4.5 Политики доступа к секретам

4.5.1 Создание политик доступа

Политика для чтения настроек:

```
vault policy write read-settings - <<EOF
# Разрешить чтение основных настроек
path "demo/settings" {
  capabilities = ["read"]
}

# Разрешить чтение метаданных
path "demo/settings" {
  capabilities = ["list"]
}
EOF
```

Политика для записи настроек:

```
vault policy write write-settings - <<EOF
# Разрешить чтение и запись настроек
```

```
path "demo/settings" {
  capabilities = ["read", "create", "update"]
}

# Разрешить удаление (только для администраторов)
path "demo/settings" {
  capabilities = ["delete"]
}
EOF
```

Политика для работы с API ключами:

```
vault policy write api-keys - <<EOF
# Разрешить чтение API ключей
path "demo/api_keys/*" {
  capabilities = ["read"]
}

# Запретить удаление API ключей
path "demo/api_keys/*" {
  capabilities = ["deny"]
}
EOF
```

4.5.2 Привязка политик к AppRole

```
# Создание AppRole с политикой только для чтения
vault write auth/approle/role/antifraud-readonly \
    token_policies="read-settings" \
    token_ttl=1h \
    token_max_ttl=4h \
    bind_secret_id=true

# Создание AppRole с политикой для записи (только для
административных сервисов)
vault write auth/approle/role/antifraud-admin \
    token_policies="write-settings,api-keys" \
    token_ttl=30m \
    token_max_ttl=2h \
    bind_secret_id=true
```

4.5.3 Принцип наименьших привилегий

Рекомендации:

- Каждый сервис должен иметь доступ только к необходимым секретам
- Использование отдельных AppRole для разных типов сервисов
- Минимальный TTL токенов (1 час для обычных сервисов, 30 минут для административных)
- Регулярный аудит политик доступа

Пример конфигурации для разных сервисов:

```
# CDR-сервис: только чтение настроек подключений
vault write auth/approle/role/antifraud-cdr \
    token_policies="read-settings" \
    token_ttl=1h

# Notificator: чтение настроек и API ключей для
уведомлений
vault write auth/approle/role/antifraud-notificator \
    token_policies="read-settings,api-keys" \
    token_ttl=1h

# API-сервис: полный доступ (требует особой
осторожности)
vault write auth/approle/role/antifraud-api \
    token_policies="write-settings,api-keys" \
    token_ttl=30m \
    token_max_ttl=2h
```

4.6 Аудит доступа к секретам

4.6.1 Включение аудита в Vault

```
# Включение файлового аудита
vault audit enable file
file_path=/var/log/vault_audit.log \
    log_raw=false \
    hmac_accessor=true \
    format=json
```

```
# Включение syslog аудита
vault audit enable syslog tag="vault" facility="AUTH" \
    log_raw=false \
    hmac_accessor=true

# Просмотр включенных устройств аудита
vault audit list
```

4.6.2 Анализ логов аудита

```
# Просмотр логов аудита в реальном времени
tail -f /var/log/vault_audit.log | jq

# Поиск всех операций чтения секретов
grep '"operation":"read"' /var/log/vault_audit.log | jq

# Поиск операций записи/обновления
grep '"operation":"update"' /var/log/vault_audit.log |
jq

# Поиск неудачных попыток доступа
grep '"error"' /var/log/vault_audit.log | jq

# Статистика доступа к секретам за последний час
grep "$(date -d '1 hour ago' +%Y-%m-%dT%H)"
/var/log/vault_audit.log | \
  jq -r '.request.path' | sort | uniq -c | sort -rn
```

4.6.3 Мониторинг подозрительной активности

Индикаторы подозрительной активности:

- Множественные неудачные попытки аутентификации
- Доступ к секретам вне обычных часов работы
- Доступ с необычных IP-адресов
- Попытки доступа к секретам, к которым сервис не должен иметь доступ
- Необычно большое количество запросов к Vault

Скрипт мониторинга:

```
#!/bin/bash
# Скрипт анализа подозрительной активности в Vault

AUDIT_LOG="/var/log/vault_audit.log"
LAST_HOUR=$(date -d '1 hour ago' +%Y-%m-%dT%H)

echo "Анализ активности Vault за последний час..."

# Неудачные попытки аутентификации
FAILED_AUTH=$(grep "$LAST_HOUR" $AUDIT_LOG | grep -c
'"error"')
if [ $FAILED_AUTH -gt 10 ]; then
  echo "ПРЕДУПРЕЖДЕНИЕ: Обнаружено $FAILED_AUTH
неудачных попыток аутентификации"
fi

# Доступ к критическим секретам
CRITICAL_ACCESS=$(grep "$LAST_HOUR" $AUDIT_LOG | \
  grep -E
'"path":"demo/(settings|api_keys|certificates)"' | wc -
l)
echo "Доступ к критическим секретам: $CRITICAL_ACCESS"

# Список уникальных клиентов
UNIQUE_CLIENTS=$(grep "$LAST_HOUR" $AUDIT_LOG | \
  jq -r '.request.client_token' | sort -u | wc -l)
echo "Уникальных клиентов: $UNIQUE_CLIENTS"
```

4.6.4 Регулярный аудит доступа

Ежедневные проверки:

- Просмотр логов на наличие ошибок доступа
- Проверка количества активных токенов
- Анализ паттернов доступа к секретам

Еженедельные проверки:

- Полный анализ логов аудита
- Проверка соответствия политик доступа принципу наименьших привилегий
- Анализ подозрительной активности

Ежемесячные проверки:

- Аудит всех политик доступа
- Проверка актуальности секретов
- Анализ эффективности ротации секретов

5. Алгоритм действий в случае компрометации

5.1 Классификация инцидентов безопасности

5.1.1 Уровни критичности инцидентов

Критический уровень (P1 - Немедленное реагирование):

- Компрометация Vault (утечка root токена или unseal ключей)
- Компрометация базы данных с персональными данными
- Получение несанкционированного доступа к production окружению
- Обнаружение активной атаки на платформу
- Утечка секретов в публичное пространство

Высокий уровень (P2 - Реагирование в течение 1 часа):

- Компрометация одного из микросервисов
- Подозрение на утечку паролей или API ключей
- Обнаружение подозрительной активности в логах
- Несанкционированный доступ к конфигурационным файлам

Средний уровень (P3 - Реагирование в течение 4 часов):

- Обнаружение уязвимостей в компонентах
- Подозрительная активность, требующая расследования
- Нарушение политик безопасности

Низкий уровень (P4 - Реагирование в течение 24 часов):

- Информационные сообщения о потенциальных угрозах
- Рекомендации по улучшению безопасности

5.1.2 Типы инцидентов

Компрометация учетных данных:

- Утечка паролей или токенов
- Компрометация AppRole credentials
- Несанкционированный доступ к учетным записям

Компрометация инфраструктуры:

- Взлом серверов или контейнеров
- Компрометация сетевой инфраструктуры

- Установка вредоносного ПО

Утечка данных:

- Несанкционированный доступ к базам данных
- Утечка конфиденциальной информации
- Экспорт данных без авторизации

Нарушение доступности:

- DDoS атаки
- Атаки типа "отказ в обслуживании"
- Саботаж инфраструктуры

5.2 Процедура обнаружения компрометации

5.2.1 Индикаторы компрометации

Индикаторы в логах Vault:

- Множественные неудачные попытки аутентификации
- Доступ к секретам с необычных IP-адресов
- Доступ к секретам вне обычных часов работы
- Попытки доступа к секретам, к которым нет прав
- Необычно большое количество запросов

Индикаторы в логах приложений:

- Ошибки аутентификации и авторизации
- Подозрительные запросы к API
- Необычные паттерны доступа к данным
- Ошибки подключения к базам данных

Индикаторы в системе мониторинга:

- Необычная сетевая активность
- Необычное использование ресурсов
- Аномальное поведение сервисов
- Срабатывание алертов безопасности

5.2.2 Процедура обнаружения

```
#!/bin/bash
# Скрипт обнаружения подозрительной активности

echo "=== Проверка подозрительной активности ==="

# 1. Проверка неудачных попыток аутентификации в Vault
FAILED_AUTH=$(grep "$(date -d '1 hour ago' +%Y-%m-
%dT%H)" /var/log/vault_audit.log | \
```

```
  grep -c '"error"')
if [ $FAILED_AUTH -gt 20 ]; then
  echo "КРИТИЧНО: Обнаружено $FAILED_AUTH неудачных
попыток аутентификации в Vault"
fi

# 2. Проверка доступа с необычных IP
UNUSUAL_IPS=$(grep "$(date -d '1 hour ago' +%Y-%m-
%dT%H)" /var/log/vault_audit.log | \
  jq -r '.request.remote_address' | sort -u)
KNOWN_IPS=("10.0.0.0/8" "192.168.0.0/16")
# Проверка IP против списка известных...

# 3. Проверка доступа к критическим секретам
CRITICAL_ACCESS=$(grep "$(date -d '1 hour ago' +%Y-%m-
%dT%H)" /var/log/vault_audit.log | \
  grep -E
'"path":"demo/(settings|api_keys|certificates|database)
"' | wc -l)
if [ $CRITICAL_ACCESS -gt 100 ]; then
  echo "ПРЕДУПРЕЖДЕНИЕ: Обнаружено $CRITICAL_ACCESS
обращений к критическим секретам"
fi

# 4. Проверка активных токенов
ACTIVE_TOKENS=$(vault list auth/token/accessors | wc -
l)
if [ $ACTIVE_TOKENS -gt 100 ]; then
  echo "ПРЕДУПРЕЖДЕНИЕ: Обнаружено $ACTIVE_TOKENS
активных токенов"
fi

# 5. Проверка ошибок в логах сервисов
for service in demo_cdr demo_notificator
demo_classifier; do
  ERRORS=$(docker service logs ${service} --since 1h
2>&1 | grep -ci "unauthorized\|forbidden\|error")
  if [ $ERRORS -gt 50 ]; then
```

```
    echo "ПРЕДУПРЕЖДЕНИЕ: Обнаружено $ERRORS ошибок в
сервисе ${service}"
  fi
done
```

5.3 Изоляция затронутых компонентов

5.3.1 Немедленные действия при обнаружении компрометации

Шаг 1: Оценка масштаба

```
# Определение затронутых компонентов
# 1. Проверка статуса всех сервисов
docker stack services demo

# 2. Проверка активных подключений
docker service ps demo --no-trunc

# 3. Анализ сетевой активности
docker network inspect demo-backend-network
```

Шаг 2: Изоляция затронутых компонентов

```
# Если скомпрометирован конкретный сервис:
# 1. Немедленная остановка сервиса
docker service scale demo_cdr=0

# 2. Изоляция сети (если требуется)
docker service update --network-rm demo-backend-network
demo_cdr

# 3. Блокировка доступа на уровне firewall
# (зависит от инфраструктуры)

# Если скомпрометирован Vault:
# 1. Немедленное запечатывание Vault
vault operator seal

# 2. Остановка всех сервисов, зависящих от Vault
docker service scale demo_cdr=0
docker service scale demo_notificator=0
```

```
docker service scale demo_classifier=0
docker service scale demo_importer=0

# 3. Блокировка сетевого доступа к Vault
```

5.3.2 Изоляция на уровне сети

```
# Блокировка доступа к затронутому узлу
# (пример для Docker Swarm)

# 1. Дренирование узла (перемещение задач на другие узлы)
docker node update --availability drain <compromised-node-id>

# 2. Блокировка узла
docker node update --label-add status=quarantine <compromised-node-id>

# 3. Изоляция сети узла (требует настройки firewall)
# Блокировка всех входящих и исходящих соединений кроме управления
```

5.4 Ротация всех секретов и паролей

5.4.1 Процедура экстренной ротации секретов

Критически важно: При подозрении на компрометацию необходимо немедленно ротировать ВСЕ секреты.

```
#!/bin/bash
# Скрипт экстренной ротации всех секретов

set -e

export VAULT_ADDR='https://vault.af.internal'
export VAULT_TOKEN='your-admin-token'

echo "=== Начало экстренной ротации секретов ==="

# 1. Ротация всех паролей баз данных
echo "Ротация паролей баз данных..."
```

```
# ClickHouse
NEW_CH_PASSWORD=$(openssl rand -base64 24)
docker exec -it clickhouse clickhouse-client --query \
  "ALTER USER admin IDENTIFIED BY '${NEW_CH_PASSWORD}'"
vault kv put demo/settings
clickhouse_password="${NEW_CH_PASSWORD}"

# PostgreSQL
NEW_PG_PASSWORD=$(openssl rand -base64 24)
docker exec -it postgres psql -U admin -d antifraud -c
\
  "ALTER USER admin WITH PASSWORD
'${NEW_PG_PASSWORD}';"
vault kv put demo/settings
postgres_password="${NEW_PG_PASSWORD}"

# 2. Ротация всех Secret ID для AppRole
echo "Ротация Secret ID для всех AppRole..."

ROLES=("antifraud" "antifraud-cdr" "antifraud-
notificator" "antifraud-classifier")

for role in "${ROLES[@]}"; do
  echo "Ротация Secret ID для ${role}..."
  NEW_SECRET_ID=$(vault write -field=secret_id -f
auth/approle/role/${role}/secret-id)

  # Обновление в соответствующих сервисах
  # (требует знания соответствия ролей и сервисов)
done

# 3. Ротация API ключей
echo "Ротация API ключей..."

# Telegram Bot Token
NEW_TELEGRAM_TOKEN="<новый токен от BotFather>"
vault kv put demo/api_keys
```

```
telegram_bot_token="${NEW_TELEGRAM_TOKEN}"

# SMTP пароль
NEW_SMTP_PASSWORD=$(openssl rand -base64 16)
vault kv put demo/api_keys
smtp_password="${NEW_SMTP_PASSWORD}"

# 4. Ротация S3 credentials
echo "Ротация S3 credentials..."
NEW_S3_ACCESS_KEY=$(openssl rand -hex 16)
NEW_S3_SECRET_KEY=$(openssl rand -hex 32)
vault kv put demo/settings \
  s3_access_key="${NEW_S3_ACCESS_KEY}" \
  s3_secret_key="${NEW_S3_SECRET_KEY}"

# 5. Синхронизация настроек во всех сервисах
echo "Синхронизация настроек..."
SERVICES=("demo_cdr" "demo_notificator"
"demo_classifier" "demo_importer")

for service in "${SERVICES[@]}"; do
  echo "Синхронизация ${service}..."
  # Определение порта сервиса и вызов API синхронизации
  # curl -X PUT http://localhost:<port>/settings/sync
done

# 6. Очистка переменных окружения
unset NEW_CH_PASSWORD NEW_PG_PASSWORD NEW_SECRET_ID
unset NEW_TELEGRAM_TOKEN NEW_SMTP_PASSWORD
unset NEW_S3_ACCESS_KEY NEW_S3_SECRET_KEY

echo "=== Экстренная ротация секретов завершена ==="
```

Управление секретами в Vault:

Критически важно: Если есть подозрение на компрометацию root токена:

```
# 1. Создание нового root токена
NEW_ROOT_TOKEN=$(vault operator generate-root -init |
```

```
grep "Nonce" | awk '{print $2}')

# 2. Использование unseal ключей для генерации (требует
3 из 5 ключей)
# Повторить для каждого unseal ключа:
vault operator generate-root

# 3. После получения нового root токена:
# - Обновить переменную VAULT_TOKEN
# - Уничтожить старый root токен
vault token revoke <old-root-token>

# 4. Обновить хранилище root токена (если используется)
```

5.4.3 Ротация unseal ключей Vault

Внимание: Ротация unseal ключей требует особой осторожности.

```
# 1. Генерация новых unseal ключей
vault operator rekey -init -key-shares=5 -key-
threshold=3

# 2. Предоставление существующих unseal ключей (требует
3 из 5)
vault operator rekey <unseal-key-1>
vault operator rekey <unseal-key-2>
vault operator rekey <unseal-key-3>

# 3. После получения новых ключей:
# - Безопасно сохранить новые ключи
# - Уничтожить старые ключи
# - Обновить хранилище ключей
```

5.5 Анализ логов и аудита

5.5.1 Сбор доказательств компрометации

```
#!/bin/bash
# Скрипт сбора доказательств компрометации

INCIDENT_ID=$(date +%Y%m%d_%H%M%S)
```

```
EVIDENCE_DIR="/var/incidents/${INCIDENT_ID}"

mkdir -p ${EVIDENCE_DIR}

echo "Сбор доказательств инцидента ${INCIDENT_ID}..."

# 1. Экспорт логов Vault
echo "Экспорт логов Vault..."
cp /var/log/vault_audit.log
${EVIDENCE_DIR}/vault_audit.log
grep "$(date -d '24 hours ago' +%Y-%m-%d)"
/var/log/vault_audit.log > \
  ${EVIDENCE_DIR}/vault_audit_last_24h.log

# 2. Экспорт логов сервисов
echo "Экспорт логов сервисов..."
for service in demo_cdr demo_notificator
demo_classifier demo_importer; do
  docker service logs ${service} --since 24h >
${EVIDENCE_DIR}/${service}_logs.txt
done

# 3. Экспорт конфигурации
echo "Экспорт конфигурации..."
cp /opt/demo/config/docker-compose.stack.yaml
${EVIDENCE_DIR}/
cp /opt/demo/config/stack.env ${EVIDENCE_DIR}/

# 4. Экспорт списка активных токенов
echo "Экспорт списка активных токенов..."
vault list auth/token/accessors >
${EVIDENCE_DIR}/active_tokens.txt

# 5. Экспорт политик доступа
echo "Экспорт политик доступа..."
vault policy list > ${EVIDENCE_DIR}/policies.txt
for policy in $(vault policy list); do
  vault policy read ${policy} >
```

```
${EVIDENCE_DIR}/policy_${policy}.txt
done

# 6. Создание архива доказательств
tar -czf ${EVIDENCE_DIR}.tar.gz ${EVIDENCE_DIR}/
gpg --encrypt --recipient security@example.com
${EVIDENCE_DIR}.tar.gz

echo "Доказательства собраны в
${EVIDENCE_DIR}.tar.gz.gpg"
```

5.5.2 Анализ временной шкалы событий

```
# Построение временной шкалы событий из логов Vault
grep "$(date -d '24 hours ago' +%Y-%m-%d)"
/var/log/vault_audit.log | \
  jq -r '[.time, .request.operation, .request.path,
.request.remote_address] | @csv' | \
  sort > /tmp/timeline.csv

# Анализ паттернов доступа
grep "$(date -d '24 hours ago' +%Y-%m-%d)"
/var/log/vault_audit.log | \
  jq -r '.request.path' | sort | uniq -c | sort -rn >
/tmp/access_patterns.txt
```

5.6 Восстановление работоспособности

5.6.1 Поэтапное восстановление

Этап 1: Восстановление Vault

```
# 1. Распечатывание Vault (если был запечатан)
vault operator unseal <unseal-key-1>
vault operator unseal <unseal-key-2>
vault operator unseal <unseal-key-3>

# 2. Проверка статуса
vault status

# 3. Проверка доступности секретов
vault kv get demo/settings
```

Этап 2: Восстановление инфраструктурных сервисов

```
# 1. Проверка и восстановление баз данных
docker exec clickhouse clickhouse-client --query
"SELECT version()"
docker exec postgres psql -U admin -d antifraud -c
"SELECT version();"

# 2. Проверка Kafka/Redpanda
docker exec redpanda rpk cluster info

# 3. Проверка MinIO
curl http://localhost:9000/minio/health/live
```

Этап 3: Восстановление сервисов платформы

```
# 1. Постепенный запуск сервисов
docker service scale demo_cdr=1
sleep 30
curl -f http://localhost:8800/healthcheck || {
  echo "ОШИБКА: CDR сервис не запустился"
  exit 1
}

docker service scale demo_notificator=1
sleep 30
curl -f http://localhost:8803/healthcheck || {
  echo "ОШИБКА: Notificator сервис не запустился"
  exit 1
}

# И так далее для остальных сервисов...

# 2. Масштабирование до нормального уровня
docker service scale demo_cdr=3
docker service scale demo_notificator=2
docker service scale demo_classifier=1
docker service scale demo_importer=2
```

5.6.2 Проверка целостности после восстановления

```
# 1. Проверка всех healthcheck эндпоинтов
for port in 8800 6061 8803 6062; do
  curl -f http://localhost:${port}/healthcheck || echo
"ОШИБКА: Порт ${port}"
done

# 2. Проверка подключения к зависимостям
docker exec -it $(docker ps -q -f name=demo_cdr) \
  vault read -format=json demo/settings

# 3. Проверка метрик в Prometheus
curl
'http://localhost:9090/api/v1/query?query=up{job=~"demo
-.*"}'

# 4. Тестирование основных функций
# (зависит от конкретных функций платформы)
```

5.7 Постмортем анализ

5.7.1 Структура постмортем отчета

Обязательные разделы отчета:

1. **Резюме инцидента:**
   - Дата и время обнаружения
   - Дата и время устранения
   - Уровень критичности
   - Краткое описание
2. **Хронология событий:**
   - Временная шкала событий
   - Ключевые моменты обнаружения и реагирования
3. **Причина инцидента:**
   - Корневая причина
   - Способ проникновения
   - Использованные уязвимости
4. **Затронутые компоненты:**
   - Список затронутых сервисов
   - Объем затронутых данных

- Потенциальный ущерб

5. **Действия по устранению:**
   - Выполненные действия
   - Ротированные секреты
   - Изменения в конфигурации
6. **Рекомендации:**
   - Меры по предотвращению повторения
   - Улучшения процессов безопасности
   - Обновления политик безопасности

5.7.2 Шаблон постмортем отчета

```
# Постмортем отчет: [Название инцидента]

**Инцидент ID:** INC-YYYYMMDD-HHMMSS
**Дата:** YYYY-MM-DD
**Уровень критичности:** P1/P2/P3/P4

## Резюме

[Краткое описание инцидента, что произошло, когда было
обнаружено, когда устранено]

## Хронология событий

| Время | Событие |
|-------|---------|
| HH:MM | Обнаружение инцидента |
| HH:MM | Начало реагирования |
| HH:MM | Изоляция затронутых компонентов |
| HH:MM | Ротация секретов |
| HH:MM | Восстановление работоспособности |

## Причина инцидента

[Описание корневой причины, способа проникновения,
использованных уязвимостей]

## Затронутые компоненты
```

```
- [Список затронутых сервисов]
- [Объем затронутых данных]
- [Потенциальный ущерб]

## Действия по устранению

1. [Действие 1]
2. [Действие 2]
3. [Действие 3]

## Рекомендации

1. [Рекомендация 1]
2. [Рекомендация 2]
3. [Рекомендация 3]

## Участники

- [Имя] - Роль
- [Имя] - Роль
```

5.8 Уведомление заинтересованных сторон

5.8.1 Процедура уведомления

Немедленное уведомление (P1 инциденты):

- Руководство службы безопасности
- Руководство IT-отдела
- Менеджер платформы
- On-call инженер

Уведомление в течение 1 часа (P2 инциденты):

- Руководство службы безопасности
- Менеджер платформы
- Команда разработки

Уведомление в течение 4 часов (P3 инциденты):

- Менеджер платформы
- Команда разработки

5.8.2 Шаблон уведомления

```
#!/bin/bash
# Скрипт отправки уведомления об инциденте

INCIDENT_ID="INC-$(date +%Y%m%d-%H%M%S)"
SEVERITY="P1"  # P1/P2/P3/P4
SUBJECT="[ВОЛНА] Инцидент безопасности ${INCIDENT_ID} -
${SEVERITY}"

BODY="Обнаружен инцидент безопасности в платформе
'Волна'.

Инцидент ID: ${INCIDENT_ID}
Уровень критичности: ${SEVERITY}
Время обнаружения: $(date)

Краткое описание:
[Описание инцидента]

Затронутые компоненты:
[Список компонентов]

Текущий статус:
[Статус реагирования]

Действия:
[Список выполненных действий]

Следующие шаги:
[Планируемые действия]
"

# Отправка уведомления (пример через email)
echo "$BODY" | mail -s "$SUBJECT" \
  security-team@example.com \
  it-manager@example.com \
  platform-manager@example.com
```

5.8.3 Регистрация инцидента

```
# Создание записи об инциденте в системе отслеживания
# (пример для системы типа Jira, ServiceNow и т.д.)

INCIDENT_DATA=$(cat <<EOF
{
  "summary": "Инцидент безопасности - ${INCIDENT_ID}",
  "description": "${BODY}",
  "priority": "${SEVERITY}",
  "labels": ["security", "incident", "vault"]
}
EOF
)

# Отправка в систему отслеживания
# curl -X POST https://tracking-
system.example.com/api/incidents \
#   -H "Content-Type: application/json" \
#   -d "${INCIDENT_DATA}"
```

6. Мониторинг безопасности

6.1 Настройка мониторинга безопасности

6.1.1 Интеграция Vault с Prometheus

Vault предоставляет метрики в формате Prometheus для мониторинга:

```
# Проверка доступности метрик Vault
curl ${VAULT_ADDR}/v1/sys/metrics?format=prometheus

# Настройка Prometheus для сбора метрик Vault
```

Конфигурация Prometheus (prometheus.yml):

```
scrape_configs:
  - job_name: 'vault'
    static_configs:
      - targets: ['vault.af.internal:8200']
    metrics_path: '/v1/sys/metrics'
    params:
```

```
      format: ['prometheus']
    scheme: https
    tls_config:
      insecure_skip_verify: true  # Используйте CA
сертификат в production
```

6.1.2 Ключевые метрики безопасности для мониторинга

Метрики доступности:

- vault_core_unsealed - статус unsealed (должен быть 1)
- vault_core_sealed - статус sealed (должен быть 0)

Метрики аутентификации:

- vault_token_count - количество активных токенов
- vault_token_create - количество созданных токенов
- vault_token_lookup - количество запросов информации о токенах

Метрики доступа к секретам:

- vault_kv_get - количество операций чтения секретов
- vault_kv_put - количество операций записи секретов
- vault_kv_delete - количество операций удаления секретов

Метрики производительности:

- vault_audit_log_request_count - количество запросов аудита
- vault_expire_num_leases - количество истекающих lease

6.1.3 Дашборд безопасности в Grafana

Создание дашборда для мониторинга безопасности:

```
{
  "dashboard": {
    "title": "Волна - Мониторинг безопасности",
    "panels": [
      {
        "title": "Статус Vault",
        "targets": [{
          "expr": "vault_core_unsealed",
          "legendFormat": "Vault Unsealed"
```

```
        }],
        "type": "stat",
        "thresholds": {
          "steps": [
            {"value": 0, "color": "red"},
            {"value": 1, "color": "green"}
          ]
        }
      },
      {
        "title": "Активные токены",
        "targets": [{
          "expr": "vault_token_count",
          "legendFormat": "Active Tokens"
        }],
        "type": "graph"
      },
      {
        "title": "Операции с секретами",
        "targets": [
          {"expr": "rate(vault_kv_get[5m])",
"legendFormat": "Reads"},
          {"expr": "rate(vault_kv_put[5m])",
"legendFormat": "Writes"},
          {"expr": "rate(vault_kv_delete[5m])",
"legendFormat": "Deletes"}
        ],
        "type": "graph"
      },
      {
        "title": "Неудачные попытки аутентификации",
        "targets": [{
          "expr":
"rate(vault_token_create{error=\"true\"}[5m])",
          "legendFormat": "Failed Auth"
        }],
        "type": "graph"
      }
```

```
    ]
  }
}
```

6.2 Алерты безопасности

6.2.1 Настройка алертов Prometheus

Создание правил алертов (alert_rules_security.yml):

```
groups:
  - name: security_alerts
    interval: 30s
    rules:
      # Алерт: Vault запечатан
      - alert: VaultSealed
        expr: vault_core_sealed == 1
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "Vault запечатан"
          description: "Vault был запечатан и
недоступен для операций"

      # Алерт: Слишком много активных токенов
      - alert: TooManyActiveTokens
        expr: vault_token_count > 200
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "Обнаружено слишком много активных
токенов"
          description: "Количество активных токенов: {{
$value }}"

      # Алерт: Высокий уровень операций записи в
секреты
      - alert: HighSecretWriteRate
        expr: rate(vault_kv_put[5m]) > 10
```

```
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "Высокий уровень операций записи в
секреты"
          description: "Скорость записи: {{ $value }}
операций/сек"

      # Алерт: Операции удаления секретов
      - alert: SecretDeletionDetected
        expr: rate(vault_kv_delete[5m]) > 0
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "Обнаружены операции удаления
секретов"
          description: "Скорость удаления: {{ $value }}
операций/сек"

      # Алерт: Множественные неудачные попытки
аутентификации
      - alert: HighFailedAuthRate
        expr:
rate(vault_token_create{error="true"}[5m]) > 5
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "Высокий уровень неудачных попыток
аутентификации"
          description: "Скорость неудачных попыток: {{
$value }} попыток/сек"

      # Алерт: Необычный доступ к критическим секретам
      - alert: UnusualCriticalSecretAccess
        expr: |
```

```
rate(vault_kv_get{path=~"demo/(settings|api_keys|certif
icates|database).*"}[5m]) > 20
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "Необычный уровень доступа к
критическим секретам"
          description: "Скорость доступа: {{ $value }}
операций/сек"
```

6.2.2 Интеграция алертов с системами уведомлений

Настройка уведомлений в Grafana:

1. Перейдите в Alerting → Notification channels
2. Добавьте каналы уведомлений:
    - Email для критических алертов
    - Slack/Telegram для предупреждений
    - PagerDuty для экстренных ситуаций

Пример настройки Email:

```
notifications:
  - name: security_critical
    type: email
    settings:
      addresses: security-team@example.com
      subject: "[КРИТИЧНО] {{ .GroupLabels.alertname
}}"
      singleEmail: true
```

6.3 Анализ подозрительной активности

6.3.1 Автоматический анализ логов аудита

```
#!/bin/bash
# Скрипт автоматического анализа подозрительной
активности

AUDIT_LOG="/var/log/vault_audit.log"
ALERT_THRESHOLD_FAILED_AUTH=20
ALERT_THRESHOLD_CRITICAL_ACCESS=50
```

```
ALERT_THRESHOLD_UNUSUAL_TIME=10

LAST_HOUR=$(date -d '1 hour ago' +%Y-%m-%dT%H)
CURRENT_HOUR=$(date +%Y-%m-%dT%H)

echo "=== Анализ подозрительной активности за последний
час ==="

# 1. Анализ неудачных попыток аутентификации
FAILED_AUTH=$(grep -E "${LAST_HOUR}|${CURRENT_HOUR}"
${AUDIT_LOG} | \
  grep -c '"error"')

if [ $FAILED_AUTH -gt $ALERT_THRESHOLD_FAILED_AUTH ];
then
  echo "ПРЕДУПРЕЖДЕНИЕ: Обнаружено $FAILED_AUTH
неудачных попыток аутентификации"
  echo "Детали:"
  grep -E "${LAST_HOUR}|${CURRENT_HOUR}" ${AUDIT_LOG} |
\
    grep '"error"' | \
    jq -r '[.time, .request.remote_address, .error] |
@csv' | \
    head -10
fi

# 2. Анализ доступа к критическим секретам
CRITICAL_ACCESS=$(grep -E
"${LAST_HOUR}|${CURRENT_HOUR}" ${AUDIT_LOG} | \
  grep -E
'"path":"demo/(settings|api_keys|certificates|database)
"' | \
  wc -l)

if [ $CRITICAL_ACCESS -gt
$ALERT_THRESHOLD_CRITICAL_ACCESS ]; then
  echo "ПРЕДУПРЕЖДЕНИЕ: Обнаружено $CRITICAL_ACCESS
обращений к критическим секретам"
```

```
  echo "Детали:"
  grep -E "${LAST_HOUR}|${CURRENT_HOUR}" ${AUDIT_LOG} |
\
    grep -E
'"path":"demo/(settings|api_keys|certificates|database)
"' | \
    jq -r '[.time, .request.path,
.request.remote_address] | @csv' | \
    head -10
fi

# 3. Анализ доступа вне рабочих часов (например, с
22:00 до 06:00)
UNUSUAL_TIME_ACCESS=$(grep -E
"${LAST_HOUR}|${CURRENT_HOUR}" ${AUDIT_LOG} | \
  jq -r 'select(.time | test("T(22|23|0[0-5])")) |
.time' | wc -l)

if [ $UNUSUAL_TIME_ACCESS -gt
$ALERT_THRESHOLD_UNUSUAL_TIME ]; then
  echo "ПРЕДУПРЕЖДЕНИЕ: Обнаружено $UNUSUAL_TIME_ACCESS
обращений вне рабочих часов"
fi

# 4. Анализ доступа с необычных IP-адресов
# (требует списка известных IP-адресов)
KNOWN_IPS_FILE="/etc/vault/known_ips.txt"
if [ -f "$KNOWN_IPS_FILE" ]; then
  UNUSUAL_IPS=$(grep -E "${LAST_HOUR}|${CURRENT_HOUR}"
${AUDIT_LOG} | \
    jq -r '.request.remote_address' | \
    while read ip; do
      grep -q "^${ip}$" "$KNOWN_IPS_FILE" || echo "$ip"
    done | sort -u | wc -l)

  if [ $UNUSUAL_IPS -gt 0 ]; then
    echo "ПРЕДУПРЕЖДЕНИЕ: Обнаружен доступ с необычных
IP-адресов"
```

```
  fi
fi

# 5. Анализ операций удаления секретов
DELETIONS=$(grep -E "${LAST_HOUR}|${CURRENT_HOUR}"
${AUDIT_LOG} | \
  grep -c '"operation":"delete"')

if [ $DELETIONS -gt 0 ]; then
  echo "КРИТИЧНО: Обнаружено $DELETIONS операций
удаления секретов"
  echo "Детали:"
  grep -E "${LAST_HOUR}|${CURRENT_HOUR}" ${AUDIT_LOG} |
\
    grep '"operation":"delete"' | \
    jq -r '[.time, .request.path,
.request.remote_address] | @csv'
fi
```

6.3.2 Мониторинг доступа к API

```
#!/bin/bash
# Скрипт мониторинга подозрительной активности в API

# Анализ логов API на подозрительные запросы
for service in demo_cdr demo_notificator
demo_classifier; do
  echo "=== Анализ сервиса ${service} ==="

  # Неудачные попытки аутентификации
  FAILED_AUTH=$(docker service logs ${service} --since
1h 2>&1 | \
    grep -ci "401\|unauthorized\|forbidden")

  if [ $FAILED_AUTH -gt 10 ]; then
    echo "ПРЕДУПРЕЖДЕНИЕ: $FAILED_AUTH неудачных
попыток аутентификации"
  fi
```

```
  # Подозрительные паттерны запросов
  SUSPICIOUS_PATTERNS=$(docker service logs ${service}
--since 1h 2>&1 | \
    grep -ciE "(sql injection|script|eval|exec|union
select)")

  if [ $SUSPICIOUS_PATTERNS -gt 0 ]; then
    echo "КРИТИЧНО: Обнаружены подозрительные паттерны
запросов"
  fi
done
```

6.4 Регулярный аудит безопасности

6.4.1 Ежедневный аудит

Контрольный список ежедневного аудита:

```
#!/bin/bash
# Скрипт ежедневного аудита безопасности

echo "=== Ежедневный аудит безопасности ==="
DATE=$(date +%Y-%m-%d)

# 1. Проверка статуса Vault
echo "1. Проверка статуса Vault..."
vault status | grep -q "Sealed.*false" || {
  echo "ОШИБКА: Vault запечатан!"
  exit 1
}

# 2. Проверка количества активных токенов
echo "2. Проверка активных токенов..."
ACTIVE_TOKENS=$(vault list auth/token/accessors
2>/dev/null | wc -l)
echo "Активных токенов: $ACTIVE_TOKENS"

# 3. Проверка неудачных попыток аутентификации за
последние 24 часа
echo "3. Проверка неудачных попыток аутентификации..."
FAILED_AUTH=$(grep "$(date -d '24 hours ago' +%Y-%m-
```

```
%d)" /var/log/vault_audit.log | \
  grep -c '"error"')
echo "Неудачных попыток за 24 часа: $FAILED_AUTH"

# 4. Проверка доступа к критическим секретам
echo "4. Проверка доступа к критическим секретам..."
CRITICAL_ACCESS=$(grep "$(date -d '24 hours ago' +%Y-
%m-%d)" /var/log/vault_audit.log | \
  grep -E
'"path":"demo/(settings|api_keys|certificates|database)
"' | wc -l)
echo "Обращений к критическим секретам:
$CRITICAL_ACCESS"

# 5. Проверка операций удаления
echo "5. Проверка операций удаления..."
DELETIONS=$(grep "$(date -d '24 hours ago' +%Y-%m-%d)"
/var/log/vault_audit.log | \
  grep -c '"operation":"delete"')
if [ $DELETIONS -gt 0 ]; then
  echo "ПРЕДУПРЕЖДЕНИЕ: Обнаружено $DELETIONS операций
удаления"
fi

# 6. Проверка актуальности секретов
echo "6. Проверка актуальности секретов..."
# (требует знания политики ротации)

echo "=== Аудит завершен ==="
```

6.4.2 Еженедельный аудит

Дополнительные проверки для еженедельного аудита:

```
#!/bin/bash
# Скрипт еженедельного аудита безопасности

echo "=== Еженедельный аудит безопасности ==="

# 1. Анализ политик доступа
```

```
echo "1. Анализ политик доступа..."
vault policy list | while read policy; do
  echo "Политика: $policy"
  vault policy read $policy | grep -E
"path|capabilities"
done

# 2. Проверка AppRole конфигураций
echo "2. Проверка AppRole конфигураций..."
vault list auth/approle/role | while read role; do
  echo "AppRole: $role"
  vault read auth/approle/role/$role | grep -E
"token_policies|token_ttl"
done

# 3. Анализ паттернов доступа за неделю
echo "3. Анализ паттернов доступа..."
grep "$(date -d '7 days ago' +%Y-%m-%d)"
/var/log/vault_audit.log | \
  jq -r '.request.path' | sort | uniq -c | sort -rn |
head -20

# 4. Проверка соответствия принципу наименьших
привилегий
echo "4. Проверка соответствия принципу наименьших
привилегий..."
# (требует ручного анализа политик)

echo "=== Еженедельный аудит завершен ==="
```

6.4.3 Ежемесячный аудит

Комплексный ежемесячный аудит:

```
#!/bin/bash
# Скрипт ежемесячного аудита безопасности

MONTH=$(date +%Y-%m)
REPORT_FILE="/var/audit/monthly_security_audit_${MONTH}
.txt"
```

```
{
  echo "=== Ежемесячный аудит безопасности за ${MONTH}
==="
  echo ""

  # 1. Статистика доступа
  echo "1. Статистика доступа:"
  echo "   Всего операций: $(grep "${MONTH}"
/var/log/vault_audit.log | wc -l)"
  echo "   Операций чтения: $(grep "${MONTH}"
/var/log/vault_audit.log | grep -c
'"operation":"read"')"
  echo "   Операций записи: $(grep "${MONTH}"
/var/log/vault_audit.log | grep -c
'"operation":"write"')"
  echo "   Операций удаления: $(grep "${MONTH}"
/var/log/vault_audit.log | grep -c
'"operation":"delete"')"
  echo ""

  # 2. Анализ неудачных попыток
  echo "2. Анализ неудачных попыток аутентификации:"
  grep "${MONTH}" /var/log/vault_audit.log | grep
'"error"' | \
    jq -r '.request.remote_address' | sort | uniq -c |
sort -rn | head -10
  echo ""

  # 3. Топ-10 наиболее часто используемых секретов
  echo "3. Топ-10 наиболее часто используемых
секретов:"
  grep "${MONTH}" /var/log/vault_audit.log | \
    jq -r '.request.path' | sort | uniq -c | sort -rn |
head -10
  echo ""

  # 4. Анализ времени доступа
```

```
  echo "4. Распределение доступа по времени суток:"
  grep "${MONTH}" /var/log/vault_audit.log | \
    jq -r '.time' | cut -d'T' -f2 | cut -d':' -f1 |
sort | uniq -c
  echo ""

  # 5. Рекомендации
  echo "5. Рекомендации:"
  echo "   - Проверить актуальность всех секретов"
  echo "   - Проверить соответствие политик доступа
принципу наименьших привилегий"
  echo "   - Обновить список известных IP-адресов"
  echo "   - Проверить актуальность ротации секретов"

} > ${REPORT_FILE}

echo "Отчет сохранен в ${REPORT_FILE}"
```

7. Резервное копирование и восстановление секретов

7.1 Резервное копирование Vault

7.1.1 Автоматическое резервное копирование

Критически важно: Регулярное резервное копирование Vault является обязательным для обеспечения возможности восстановления после сбоев.

```
#!/bin/bash
# Скрипт автоматического резервного копирования Vault

set -e

BACKUP_DIR="/backup/vault"
DATE=$(date +%Y%m%d_%H%M%S)
RETENTION_DAYS=90

export VAULT_ADDR='https://vault.af.internal'
export VAULT_TOKEN='your-admin-token'

# Создание директории для резервных копий
mkdir -p ${BACKUP_DIR}
```

```
echo "=== Начало резервного копирования Vault ==="

# 1. Проверка статуса Vault
vault status | grep -q "Sealed.*false" || {
  echo "ОШИБКА: Vault запечатан, невозможно создать
резервную копию"
  exit 1
}

# 2. Экспорт всех секретов
echo "Экспорт секретов..."
vault operator export -format=json >
${BACKUP_DIR}/vault_backup_${DATE}.json

# 3. Проверка целостности резервной копии
if [ ! -s ${BACKUP_DIR}/vault_backup_${DATE}.json ];
then
  echo "ОШИБКА: Резервная копия пуста или не создана"
  exit 1
fi

# 4. Шифрование резервной копии
echo "Шифрование резервной копии..."
gpg --encrypt --recipient security@example.com \
  ${BACKUP_DIR}/vault_backup_${DATE}.json

# 5. Удаление незашифрованной копии
rm ${BACKUP_DIR}/vault_backup_${DATE}.json

# 6. Проверка целостности зашифрованной копии
if [ ! -f ${BACKUP_DIR}/vault_backup_${DATE}.json.gpg
]; then
  echo "ОШИБКА: Зашифрованная резервная копия не
создана"
  exit 1
fi
```

```
# 7. Создание контрольной суммы
sha256sum ${BACKUP_DIR}/vault_backup_${DATE}.json.gpg >
\
  ${BACKUP_DIR}/vault_backup_${DATE}.json.gpg.sha256

# 8. Удаление старых резервных копий
echo "Удаление резервных копий старше ${RETENTION_DAYS}
дней..."
find ${BACKUP_DIR} -name "vault_backup_*.json.gpg" -
mtime +${RETENTION_DAYS} -delete
find ${BACKUP_DIR} -name
"vault_backup_*.json.gpg.sha256" -mtime
+${RETENTION_DAYS} -delete

# 9. Копирование на внешний носитель (если настроено)
# rsync -av ${BACKUP_DIR}/ remote-backup-
server:/backups/vault/

echo "=== Резервное копирование завершено ==="
echo "Резервная копия:
${BACKUP_DIR}/vault_backup_${DATE}.json.gpg"
```

7.1.2 Настройка автоматического резервного копирования

Настройка cron для ежедневного резервного копирования:

```
# Добавить в crontab (ежедневно в 2:00)
0 2 * * * /opt/scripts/backup_vault.sh >>
/var/log/vault_backup.log 2>&1
```

Настройка cron для еженедельного полного резервного копирования:

```
# Добавить в crontab (каждое воскресенье в 1:00)
0 1 * * 0 /opt/scripts/backup_vault_full.sh >>
/var/log/vault_backup_full.log 2>&1
```

7.1.3 Резервное копирование через API

Альтернативный способ через API:

```
#!/bin/bash
# Резервное копирование через Vault API
```

```
BACKUP_DIR="/backup/vault"
DATE=$(date +%Y%m%d_%H%M%S)
VAULT_ADDR='https://vault.af.internal'
VAULT_TOKEN='your-admin-token'

# Экспорт через API
curl \
  --header "X-Vault-Token: ${VAULT_TOKEN}" \
  --request GET \
  ${VAULT_ADDR}/v1/sys/export \
  > ${BACKUP_DIR}/vault_backup_api_${DATE}.json

# Шифрование
gpg --encrypt --recipient security@example.com \
  ${BACKUP_DIR}/vault_backup_api_${DATE}.json

rm ${BACKUP_DIR}/vault_backup_api_${DATE}.json
```

7.2 Шифрование резервных копий

7.2.1 Использование GPG для шифрования

Настройка GPG ключей:

```
# 1. Генерация GPG ключа для шифрования резервных копий
gpg --full-generate-key
# Выбрать:
# - RSA and RSA (default)
# - 4096 bits
# - Срок действия: 0 (без срока)
# - Имя и email: Security Team <security@example.com>

# 2. Экспорт публичного ключа для распространения
gpg --armor --export security@example.com >
vault_backup_public_key.asc

# 3. Импорт публичного ключа на серверах резервного
копирования
gpg --import vault_backup_public_key.asc
```

Шифрование резервной копии:

```
# Шифрование для одного получателя
gpg --encrypt --recipient security@example.com
vault_backup.json

# Шифрование для нескольких получателей
gpg --encrypt \
  --recipient security@example.com \
  --recipient admin@example.com \
  vault_backup.json

# Шифрование с симметричным ключом (менее безопасно, но проще)
gpg --symmetric --cipher-algo AES256 vault_backup.json
```

7.2.2 Проверка целостности зашифрованных резервных копий

```
#!/bin/bash
# Скрипт проверки целостности резервных копий

BACKUP_DIR="/backup/vault"

echo "=== Проверка целостности резервных копий ==="

for backup in ${BACKUP_DIR}/vault_backup_*.json.gpg; do
  if [ -f "${backup}" ]; then
    echo "Проверка: $(basename ${backup})"

    # Проверка контрольной суммы
    if [ -f "${backup}.sha256" ]; then
      sha256sum -c "${backup}.sha256" || {
        echo "ОШИБКА: Контрольная сумма не совпадает"
        continue
      }
    fi

    # Проверка возможности расшифровки (без фактической расшифровки)
    gpg --verify ${backup} 2>&1 | grep -q "Good
```

```
signature" || {
      echo "ПРЕДУПРЕЖДЕНИЕ: Не удалось проверить
подпись"
    }

    echo "OK"
  fi
done

echo "=== Проверка завершена ==="
```

7.3 Восстановление секретов

7.3.1 Процедура восстановления из резервной копии

Внимание: Восстановление из резервной копии должно выполняться с особой осторожностью, так как может перезаписать текущие данные.

```
#!/bin/bash
# Скрипт восстановления Vault из резервной копии

set -e

BACKUP_FILE=$1

if [ -z "$BACKUP_FILE" ]; then
  echo "Использование: $0 <путь_к_резервной_копии>"
  echo "Пример: $0
/backup/vault/vault_backup_20250101.json.gpg"
  exit 1
fi

export VAULT_ADDR='https://vault.af.internal'
export VAULT_TOKEN='your-admin-token'

echo "=== Восстановление Vault из резервной копии ==="
echo "Резервная копия: ${BACKUP_FILE}"

# 1. Проверка существования резервной копии
if [ ! -f "${BACKUP_FILE}" ]; then
  echo "ОШИБКА: Резервная копия не найдена"
```

```
  exit 1
fi

# 2. Расшифровка резервной копии
echo "Расшифровка резервной копии..."
DECRYPTED_FILE="/tmp/vault_backup_decrypted_$(date
+%Y%m%d_%H%M%S).json"
gpg --decrypt ${BACKUP_FILE} > ${DECRYPTED_FILE}

# 3. Проверка целостности расшифрованного файла
if [ ! -s ${DECRYPTED_FILE} ]; then
  echo "ОШИБКА: Расшифрованный файл пуст или поврежден"
  rm -f ${DECRYPTED_FILE}
  exit 1
fi

# 4. Проверка статуса Vault
echo "Проверка статуса Vault..."
vault status | grep -q "Sealed.*false" || {
  echo "ОШИБКА: Vault запечатан, необходимо распечатать
перед восстановлением"
  rm -f ${DECRYPTED_FILE}
  exit 1
}

# 5. Подтверждение восстановления
echo ""
echo "ВНИМАНИЕ: Восстановление перезапишет текущие
данные в Vault!"
echo "Убедитесь, что у вас есть актуальная резервная
копия текущего состояния."
read -p "Продолжить восстановление? (yes/no): " CONFIRM

if [ "$CONFIRM" != "yes" ]; then
  echo "Восстановление отменено"
  rm -f ${DECRYPTED_FILE}
  exit 0
fi
```

```
# 6. Остановка сервисов, использующих Vault
(рекомендуется)
echo ""
read -p "Остановить сервисы перед восстановлением?
(yes/no): " STOP_SERVICES

if [ "$STOP_SERVICES" = "yes" ]; then
  echo "Остановка сервисов..."
  docker service scale demo_cdr=0
  docker service scale demo_notificator=0
  docker service scale demo_classifier=0
  docker service scale demo_importer=0
  sleep 10
fi

# 7. Восстановление из резервной копии
echo "Восстановление секретов..."
vault operator import ${DECRYPTED_FILE}

# 8. Удаление расшифрованного файла
rm -f ${DECRYPTED_FILE}

# 9. Проверка восстановленных секретов
echo "Проверка восстановленных секретов..."
vault kv get demo/settings || {
  echo "ПРЕДУПРЕЖДЕНИЕ: Не удалось проверить
восстановленные секреты"
}

# 10. Запуск сервисов (если были остановлены)
if [ "$STOP_SERVICES" = "yes" ]; then
  echo "Запуск сервисов..."
  docker service scale demo_cdr=3
  docker service scale demo_notificator=2
  docker service scale demo_classifier=1
  docker service scale demo_importer=2
```

```
  # Ожидание готовности сервисов
  sleep 30

  # Проверка работоспособности
  for port in 8800 6061 8803 6062; do
    curl -f http://localhost:${port}/healthcheck || {
      echo "ПРЕДУПРЕЖДЕНИЕ: Сервис на порту ${port} не
отвечает"
    }
  done
fi

echo "=== Восстановление завершено ==="
```

7.3.2 Восстановление отдельных секретов

Восстановление конкретного секрета без полного восстановления:

```
#!/bin/bash
# Восстановление отдельного секрета из резервной копии

BACKUP_FILE=$1
SECRET_PATH=$2

if [ -z "$BACKUP_FILE" ] || [ -z "$SECRET_PATH" ]; then
  echo "Использование: $0 <резервная_копия>
<путь_к_секрету>"
  echo "Пример: $0 /backup/vault/vault_backup.json.gpg
demo/settings"
  exit 1
fi

export VAULT_ADDR='https://vault.af.internal'
export VAULT_TOKEN='your-admin-token'

# Расшифровка резервной копии
DECRYPTED_FILE="/tmp/vault_backup_temp.json"
gpg --decrypt ${BACKUP_FILE} > ${DECRYPTED_FILE}
```

```
# Извлечение конкретного секрета
SECRET_DATA=$(jq -r ".data.${SECRET_PATH}"
${DECRYPTED_FILE})

if [ "$SECRET_DATA" = "null" ] || [ -z "$SECRET_DATA"
]; then
  echo "ОШИБКА: Секрет не найден в резервной копии"
  rm -f ${DECRYPTED_FILE}
  exit 1
fi

# Восстановление секрета
echo "Восстановление секрета ${SECRET_PATH}..."
echo "${SECRET_DATA}" | vault kv put ${SECRET_PATH} -

# Удаление временного файла
rm -f ${DECRYPTED_FILE}

echo "Секрет ${SECRET_PATH} восстановлен"
```

7.3.3 Контрольный список восстановления

Перед восстановлением:

- ✓ Создана резервная копия текущего состояния Vault
- ✓ Проверена целостность резервной копии для восстановления
- ✓ Получено подтверждение от руководства на восстановление
- ✓ Запланировано окно обслуживания
- ✓ Уведомлены заинтересованные стороны

Во время восстановления:

- ✓ Остановлены сервисы, использующие Vault (если требуется)
- ✓ Выполнено восстановление из резервной копии
- ✓ Проверена целостность восстановленных данных

После восстановления:

- ✓ Проверена работоспособность всех сервисов

- ✓ Проверены healthcheck эндпоинты
- ✓ Проверены метрики в системе мониторинга
- ✓ Выполнено тестирование основных функций
- ✓ Документирован процесс восстановления

8. Приложения

8.1 Контрольные списки безопасности

8.1.1 Ежедневный контрольный список

Утренняя проверка:

- ✓ Проверка статуса Vault (vault status)
- ✓ Проверка критических алертов в системе мониторинга
- ✓ Просмотр логов Vault на наличие ошибок за последние 24 часа
- ✓ Проверка количества активных токенов
- ✓ Проверка работоспособности всех сервисов платформы

Вечерняя проверка:

- ✓ Анализ подозрительной активности за день
- ✓ Проверка метрик безопасности в Grafana
- ✓ Проверка успешности резервного копирования (если запланировано)
- ✓ Документирование обнаруженных проблем

8.1.2 Еженедельный контрольный список

- ✓ Полный анализ логов аудита Vault за неделю
- ✓ Проверка соответствия политик доступа принципу наименьших привилегий
- ✓ Анализ паттернов доступа к секретам
- ✓ Проверка актуальности секретов (срок действия, необходимость ротации)
- ✓ Проверка обновлений безопасности компонентов
- ✓ Обновление документации при необходимости

8.1.3 Ежемесячный контрольный список

- ✓ Комплексный аудит безопасности
- ✓ Ротация секретов согласно политике
- ✓ Проверка и обновление политик доступа
- ✓ Анализ эффективности мер безопасности
- ✓ Обновление списка известных IP-адресов
- ✓ Проверка актуальности резервных копий
- ✓ Обновление планов реагирования на инциденты
- ✓ Обучение команды новым процедурам безопасности

8.1.4 Контрольный список при обновлении компонентов

Перед обновлением:

- ✓ Изучены release notes на предмет исправлений безопасности
- ✓ Проверена подпись и целостность образов
- ✓ Созданы резервные копии Vault, конфигурации и сертификатов
- ✓ Проверено текущее состояние безопасности
- ✓ Запланировано окно обслуживания
- ✓ Уведомлены заинтересованные стороны

Во время обновления:

- ✓ Обновление выполняется с указанием конкретных версий
- ✓ Используется стратегия rolling update
- ✓ Мониторится процесс обновления
- ✓ Проверяются healthcheck эндпоинты

После обновления:

- ✓ Проверено подключение к Vault
- ✓ Проверена аутентификация всех сервисов
- ✓ Проверены healthcheck эндпоинты
- ✓ Проверены метрики в системе мониторинга
- ✓ Проверены логи на наличие ошибок
- ✓ Выполнено тестирование основных функций
- ✓ Документированы изменения

8.1.5 Контрольный список при инциденте безопасности

Немедленные действия:

- ✓ Оценка масштаба и критичности инцидента
- ✓ Уведомление команды безопасности и руководства
- ✓ Изоляция затронутых компонентов
- ✓ Сбор доказательств инцидента
- ✓ Документирование временной шкалы событий

Действия по устранению:

- ✓ Ротация всех затронутых секретов
- ✓ Блокировка компрометированных учетных записей
- ✓ Восстановление работоспособности системы
- ✓ Проверка целостности данных

После устранения:

- ✓ Проверка работоспособности всех компонентов
- ✓ Анализ причин инцидента

- ✓ Составление постмортем отчета
- ✓ Внедрение мер по предотвращению повторения
- ✓ Обновление процедур безопасности

8.2 Полезные команды Vault

8.2.1 Базовые команды

```
# Проверка статуса Vault
vault status

# Проверка версии Vault
vault version

# Получение помощи по команде
vault <command> -help

# Автодополнение (добавить в ~/.bashrc или ~/.zshrc)
complete -C /usr/local/bin/vault vault
```

8.2.2 Работа с секретами

```
# Просмотр секрета
vault kv get demo/settings

# Просмотр секрета в формате JSON
vault kv get -format=json demo/settings

# Просмотр конкретного поля секрета
vault kv get -field=clickhouse_password demo/settings

# Создание/обновление секрета
vault kv put demo/settings key1=value1 key2=value2

# Обновление отдельного поля
vault kv patch demo/settings new_key=new_value

# Удаление секрета
vault kv delete demo/settings

# Удаление всех версий секрета
```

```
vault kv destroy demo/settings

# Просмотр метаданных секрета
vault kv metadata get demo/settings

# Просмотр всех версий секрета
vault kv get -versions demo/settings

# Восстановление предыдущей версии
vault kv rollback -version=5 demo/settings

# Список всех секретов в пути
vault kv list demo/
```

8.2.3 Работа с политиками

```
# Список всех политик
vault policy list

# Просмотр политики
vault policy read <policy-name>

# Создание/обновление политики из файла
vault policy write <policy-name> policy.hcl

# Создание политики из stdin
vault policy write <policy-name> - <<EOF
path "demo/settings" {
  capabilities = ["read"]
}
EOF

# Удаление политики
vault policy delete <policy-name>
```

8.2.4 Работа с AppRole

```
# Список всех AppRole
vault list auth/approle/role
```

```
# Просмотр настроек AppRole
vault read auth/approle/role/<role-name>

# Создание AppRole
vault write auth/approle/role/<role-name> \
  token_policies="<policy-name>" \
  token_ttl=1h \
  token_max_ttl=4h

# Получение Role ID
vault read auth/approle/role/<role-name>/role-id

# Генерация нового Secret ID
vault write -f auth/approle/role/<role-name>/secret-id

# Просмотр Secret ID (требует root токен)
vault list auth/approle/role/<role-name>/secret-id

# Удаление Secret ID
vault write auth/approle/role/<role-name>/secret-
id/destroy \
  secret_id=<secret-id>

# Аутентификация с Role ID и Secret ID
vault write auth/approle/login \
  role_id=<role-id> \
  secret_id=<secret-id>
```

8.2.5 Работа с токенами

```
# Просмотр информации о текущем токене
vault token lookup

# Просмотр информации о конкретном токене
vault token lookup <token>

# Создание нового токена
vault token create
```

```
# Создание токена с политикой
vault token create -policy=<policy-name>

# Создание токена с TTL
vault token create -ttl=1h

# Обновление TTL токена
vault token renew <token>

# Отзыв токена
vault token revoke <token>

# Отзыв токена и всех дочерних токенов
vault token revoke -mode=path <token>

# Список всех токенов (требует root токен)
vault list auth/token/accessors

# Просмотр информации о токене по accessor
vault token lookup -accessor <accessor>
```

8.2.6 Аудит и мониторинг

```
# Список устройств аудита
vault audit list

# Включение файлового аудита
vault audit enable file
file_path=/var/log/vault_audit.log

# Включение syslog аудита
vault audit enable syslog tag="vault"

# Отключение аудита
vault audit disable file

# Просмотр метрик
curl ${VAULT_ADDR}/v1/sys/metrics?format=prometheus
```

```
# Просмотр информации о производительности
vault operator perf-standby
```

8.2.7 Управление Vault

```
# Распечатывание Vault
vault operator unseal <unseal-key>

# Запечатывание Vault
vault operator seal

# Генерация нового root токена
vault operator generate-root -init

# Ротация unseal ключей
vault operator rekey -init -key-shares=5 -key-
threshold=3

# Экспорт данных Vault
vault operator export -format=json > backup.json

# Импорт данных Vault
vault operator import backup.json
```

8.2.8 Диагностика

```
# Проверка здоровья Vault
curl ${VAULT_ADDR}/v1/sys/health

# Проверка готовности Vault
curl ${VAULT_ADDR}/v1/sys/health?standbyok=true

# Просмотр информации о кластере
vault operator members

# Просмотр информации о производительности
vault operator perf-standby

# Проверка подключения к Vault из контейнера
docker exec -it <container-id> vault read demo/settings
```

8.3 Полезные скрипты

8.3.1 Скрипт проверки безопасности

```
#!/bin/bash
# Скрипт комплексной проверки безопасности

echo "=== Проверка безопасности платформы Волна ==="

# Проверка статуса Vault
echo "1. Проверка Vault..."
vault status | grep -q "Sealed.*false" && echo "   ✓
Vault распечатан" || echo "   ✗ Vault запечатан!"

# Проверка активных токенов
TOKEN_COUNT=$(vault list auth/token/accessors
2>/dev/null | wc -l)
echo "2. Активных токенов: $TOKEN_COUNT"
[ $TOKEN_COUNT -lt 100 ] && echo "   ✓ Количество
токенов в норме" || echo "   ⚠ Слишком много токенов"

# Проверка неудачных попыток аутентификации за
последние 24 часа
FAILED=$(grep "$(date -d '24 hours ago' +%Y-%m-%d)"
/var/log/vault_audit.log 2>/dev/null | grep -c
'"error"' || echo "0")
echo "3. Неудачных попыток за 24 часа: $FAILED"
[ $FAILED -lt 20 ] && echo "   ✓ Количество ошибок в
норме" || echo "   ⚠ Много неудачных попыток"

# Проверка работоспособности сервисов
echo "4. Проверка сервисов..."
for port in 8800 6061 8803 6062; do
  curl -sf http://localhost:${port}/healthcheck >
/dev/null && \
    echo "   ✓ Сервис на порту ${port} работает" || \
    echo "   ✗ Сервис на порту ${port} не отвечает"
done
```

```
echo "=== Проверка завершена ==="
```

8.3.2 Скрипт ротации секретов

```
#!/bin/bash
# Скрипт ротации секретов согласно политике

export VAULT_ADDR='https://vault.af.internal'
export VAULT_TOKEN='your-admin-token'

# Ротация Secret ID для всех AppRole
for role in $(vault list auth/approle/role 2>/dev/null
| tail -n +3); do
  echo "Ротация Secret ID для ${role}..."
  vault write -f auth/approle/role/${role}/secret-id >
/dev/null
done

echo "Ротация Secret ID завершена"
```

8.4 Контакты службы безопасности

8.4.1 Контакты для экстренных ситуаций

Критические инциденты (P1):

- **Email экстренных уведомлений:** security-emergency@rt-solar.ru

Высокоприоритетные инциденты (P2):

- **Email службы безопасности:** soc@rt-solar.ru

Обычные запросы:

- **Email службы безопасности:** security@rt-solar.ru
- **Служба поддержки:** support@rt-solar.ru

8.4.2 Процедура обращения

При обнаружении инцидента безопасности:

1. **Немедленно:**
   - Оцените критичность инцидента
   - Изолируйте затронутые компоненты (если возможно)
   - Свяжитесь с on-call инженером безопасности
2. **В течение 1 часа:**

- Соберите доказательства инцидента
- Документируйте временную шкалу событий
- Отправьте уведомление в службу безопасности

3. **После устранения:**
   - Составьте постмортем отчет
   - Внедрите меры по предотвращению повторения
   - Обновите документацию

Заключение

Данная инструкция администратора безопасности содержит основные процедуры обеспечения безопасности платформы "Волна". Регулярное следование рекомендациям и контрольным спискам обеспечит высокий уровень безопасности платформы и минимизирует риски возникновения инцидентов.

Ключевые принципы безопасности:

1. **Принцип наименьших привилегий** - каждый компонент должен иметь доступ только к необходимым ресурсам
2. **Регулярная ротация секретов** - все пароли и ключи должны регулярно обновляться
3. **Мониторинг и аудит** - все операции с секретами должны логироваться и мониториться
4. **Готовность к инцидентам** - наличие планов реагирования и регулярное их тестирование
5. **Резервное копирование** - регулярное создание и проверка резервных копий критических данных

При возникновении вопросов или проблем, не описанных в данной инструкции, обращайтесь в службу безопасности.