УТВЕРЖДЕНО

RU.86432418.00002-01-01 90 01-1 - ЛУ

Программное обеспечение «Deckhouse Stronghold»

Руководство администратора

RU.86432418.00002-01-01 90 01-1

Листов 101

2025

Содержание

Список используемых обозначений и сокращений

ОС	Операционная система
ПО	Программное обеспечение
ПО «Deckhouse Stronghold», Stronghold, Изделие	Программное обеспечение «Deckhouse Stronghold»
ОЗУ	Оперативное запоминающее устройство
Снэпшот	Моментальная копия хранимых данных
ФСБ России	Федеральная служба безопасности Российской Федерации
ЦС	Центр сертификации
Эндпойнт	Конечная точка
API	Application Programming Interface (прикладной программный интерфейс)
CIDR	Classless Inter-Domain Routing (бесклассовая адресация)
CLI	Command Line Interface (интерфейс командной строки)
DNS	Domain Name System (система доменных имен)
HA	Режим высокой доступности
HTTP	Hypertext Transfer Protocol (протокол передачи гипертекста)
OIDC	Open ID connect (протокол проверки подлинности)
PEM	Privacy Enhanced Mail (текстовый формат файлов, который используется для хранения криптографических данных, таких как сертификаты и закрытые ключи)
JSON	JavaScript Object Notation (текстовый формат обмена данными, основанный на JavaScript)
JWT	JSON Web Token (стандарт для создания токенов доступа, основанный на формате JSON)
JWKS	JSON Web Key Set (набор открытых ключей, которые используются для проверки токена)
TLS	Transport layer security (протокол защиты транспортного уровня)
TTL	Time to live (предельный период заданного времени)
UI	User Interface (пользовательский интерфейс)
USB	Universal Serial Bus (интерфейс для подключения периферийных устройств к вычислительной технике)

1. Действия по приемке поставленного средства

Приемка поставленного ПО «Deckhouse Stronghold» осуществляется в соответствии с указаниями, содержащимися в документе «Программное обеспечение «Stronghold». Технические условия. RU.86432418.00002-01 ТУ 02».

Перед началом эксплуатации для обнаружения любого расхождения между оригиналом ПО «Deckhouse Stronghold» и версией, полученной Заказчиком, проводится процедура приемки.

Приемка поставленного ПО «Deckhouse Stronghold» включает следующие процедуры:

- проверка упаковки;
- проверка маркировки;
- проверка комплектности;
- проверка целостности.

Проверка упаковки, маркировки и комплектности проводится методом визуального осмотра. При осмотре упаковки проверяется:

- наименование и адрес отправителя (изготовителя);
- отсутствие механических повреждений упаковки.

Если на упаковке имеются значительные повреждения, которые могут свидетельствовать о нарушении целостности упаковки и ее содержимого, а также о неправомерном вскрытии конверта третьими лицами, или отправителем не является Акционерное общество «Флант» (адрес: Акционерное общество «Флант», 115088, г. Москва, ул. Угрешская, д. 12, стр. 4, офис 47А), такой комплект поставки признается бракованным и отсылается обратно отправителю. Если по результатам проверки целостности упаковки нарушений выявлено не было – проводится проверка упаковки и маркировки.

Проверка маркировки подразумевает проверку наличия во вкладыше следующих данных:

- наименование изделия;
- децимальный номер;
- логотип предприятия-производителя;
- год выпуска;
- серийный номер;
- адрес, номер телефона, адрес электронной почты, ссылка на сайт предприятия-производителя;
- краткая информация об изделии и его основных функциональных возможностях.

На этапе приемки должна выполняться проверка заполнения и подписания ответственными разделов «Свидетельство о приемке» и «Свидетельство об упаковке и маркировке» в Формуляре, поставляемом на бумажном носителе.

Если по результатам проверки маркировки обнаружено отсутствие какого-либо элемента маркировки, такой комплект поставки признается бракованным и отсылается обратно отправителю. Если по результатам проверки маркировки нарушений выявлено не было, проводится проверка комплектности поставки.

Проверка комплектности поставки проводится методом сравнения состава полученной поставки требованиям, указанным в п. 4.1 Формуляра. Если по результатам проверки комплектности поставки расхождения с Формуляром отсутствуют, далее производится проверка целостности.

В рамках проверки целостности должны быть проведены:

- проверка целостности USB флэш-накопителя;
- проверка целостности дистрибутива ПО «Deckhouse Stronghold»;
- проверка целостности установленного ПО «Deckhouse Stronghold».

Проверка целостности USB флэш-накопителя выполняется путем визуального определения отсутствия каких-либо механических или иных повреждений. Если по результатам проверки на USB флэш-накопителе обнаружены повреждения, такой комплект поставки признается бракованным и отсылается обратно отправителю.

Проверка целостности ПО «Deckhouse Stronghold» на USB флэш-накопителе информации выполняется путем снятия контрольных сумм с помощью с использованием утилиты gostsum из состава сертифицированной ОС и сравнения их с контрольными суммами, приведенными в Электронном приложении к Формуляру. Для контейнерного исполнения: «Электронные приложения» – «Deckhouse Stronghold. Формуляр» – «distr_cs_01.txt», в виде дистрибутива для установки на ОС: «Электронные приложения» – «Deckhouse Stronghold. Формуляр» – «distr_cs_02.txt». Если по результатам проверки контрольная сумма дистрибутива, записанного на USB флэш-накопитель информации, соответствует значению контрольной суммы, приведенной в Формуляре, требуется выполнить установку ПО «Deckhouse Stronghold» в соответствии с разделом 3 настоящего документа. Если по результатам проверки контрольная сумма дистрибутива ПО «Deckhouse Stronghold», записанного на USB флэш-накопитель информации, не соответствует значению контрольной суммы, приведенной в Формуляре, такой комплект поставки признается бракованным и отсылается обратно отправителю.

Проверка целостности установленного ПО «Deckhouse Stronghold» выполняется путем снятия контрольных сумм с исполняемых файлов с помощью утилиты gostsum из состава сертифицированной ОС и сравнения полученных контрольных сумм с контрольными суммами, приведенными в Электронном приложении к Формуляру. Для контейнерного исполнения: «Электронные приложения» – «Deckhouse Stronghold. Формуляр» – «bins_cs_01.txt», в виде дистрибутива для установки на ОС: «Электронные приложения» – «Deckhouse Stronghold. Формуляр» – «bins_cs_02.txt». Если по результатам проверки контрольные суммы исполняемых файлов установленного ПО «Deckhouse Stronghold» соответствуют значениям контрольных сумм, приведенных в Формуляре, полученный экземпляр ПО «Deckhouse Stronghold» считается соответствующим оригиналу ПО «Deckhouse Stronghold». Если по результатам проверки контрольные суммы исполняемых файлов установленного ПО «Deckhouse Stronghold» не соответствуют значениям контрольных сумм, приведенным в Формуляре, такой комплект поставки признается бракованным и отсылается обратно отправителю.

В случае, если по всем указанным процедурам установлено полное соответствие, ПО «Deckhouse Stronghold» признается годным к эксплуатации.

2. Действия по безопасной установке и настройке ПО «Deckhouse Stronghold»

При установке и настройке ПО «Deckhouse Stronghold» для обеспечения безопасности необходимо выполнение следующих условий:

- инсталляция ПО «Deckhouse Stronghold» должна осуществляться в защищенной инфраструктуре, работником соответствующей квалификации, имеющим права администратора с присвоенными ему идентификационными данными для работы в среде функционирования ПО «Deckhouse Stronghold»;
- действия, проводимые при инсталляции ПО «Deckhouse Stronghold», а также при инициализации ПО «Deckhouse Stronghold», подлежат логированию;
- установка и конфигурирование ПО «Deckhouse Stronghold» должны осуществляться в соответствии с эксплуатационной документацией;
- должно обеспечиваться предотвращение несанкционированного доступа к идентификаторам и паролям пользователей сервиса и привилегированных пользователей (администраторов информационной безопасности) ПО «Deckhouse Stronghold»;
- должно обеспечиваться предотвращение несанкционированного доступа к идентификаторам и паролям администраторов среды виртуализации, которые необходимы для установки и настройки ПО «Deckhouse Stronghold».

3. Установка и настройка

Перед установкой необходимо убедиться, что выполнены минимальные требования к аппаратным и программным средствам. ПО «Deckhouse Stronghold» может поставляться в виде дистрибутива для установки на ОС или в контейнере. В случае поставки Изделия в виде дистрибутива необходимо выбрать одну из трех ОС, представленных в столбце «Требования к программной части» Таблицы 1. В случае поставки ПО «Deckhouse Stronghold» в контейнерном исполнении Изделие должно функционировать в ПО «Deckhouse Platform Certified Security Edition» (не ниже 1.67).

Таблица 1. Минимальные требования к аппаратным и программным средствам

Изделие	**Требования к аппаратной части**	**Требования к программной части**
ПО «Deckhouse Stronghold»	− Архитектура процессора x86-64; − не менее 2 ядер CPU; − не менее 2 GB RAM; − не менее 10 ГБ дискового пространства.	− ОС РЕД ОС (не ниже 7.3); − ОС Astra Linux Special Edition (не ниже 1.7); − ОС Альт не ниже 8 СП.

Перед установкой необходимо проверить следующие условия:

- на сервер скопирован файл дистрибутива ПО «Deckhouse Stronghold» по пути /opt/stronghold/stronghold;
- права на файл установлены следующие:

```
chown stronghold:stronghold /opt/stronghold/stronghold
chmod 500 /opt/stronghold/stronghold
```

- команда доступна без указания полного пути:

```
ln -s /opt/stronghold/stronghold /usr/bin/stronghold
```

- создан systemd-unit;
- наличие сертификатов для каждого узла в кластере Raft, а также сертификата корневого центра сертификации.

Сертификаты узлов используются для идентификации перед другими узлами кластера.

Наличие корневого сертификата нужно для того, чтобы несколько узлов кластера с помощью него могли проверить сертификаты друг друга.

Справочник администратора, содержащий дополнительную информацию о ПО «Deckhouse Stronghold», приведен в электронном приложении к настоящему документу, каталог «Электронные приложения» - «Deckhouse Stronghold. Руководство администратора» - «Руководство.pdf».

3.1. Установка и настройка ПО в режиме высокой доступности

ПО «Deckhouse Stronghold» поддерживает мультисерверный режим для обеспечения высокой доступности (HA, high availability). Этот режим автоматически включается при использовании хранилища данных, которое его поддерживает, и защищает систему от сбоев за счет работы нескольких серверов ПО.

Чтобы определить, поддерживает ли ваше хранилище данных режим высокой доступности, запустите сервер и проверьте, выводится ли сообщение «HA available» рядом с информацией о хранилище. Если сообщение выводится, ПО «Deckhouse Stronghold» будет автоматически использовать режим HA.

Для обеспечения высокой доступности один из узлов ПО «Deckhouse Stronghold» получает блокировку в системе хранения данных. Затем этот узел становится активным, в то время как остальные узлы переходят в режим ожидания. Если резервные узлы получают запросы, они либо перенаправляют их, либо переадресовывают клиентов в соответствии с настройками и текущим состоянием кластера.

Для развертывания ПО «Deckhouse Stronghold» в режиме HA с интегрированным хранилищем Raft вам понадобятся как минимум три сервера ПО «Deckhouse Stronghold». В противном случае не получится достичь кворума и распечатать хранилище.

3.1.1. Предварительная подготовка инфраструктуры

Здесь и далее предполагается, что сервис запущен под пользователем stronghold и такой пользователь существует. Если требуется запустить сервис под другим пользователем, укажите необходимое имя пользователя.

1. Создайте файл /etc/systemd/system/stronghold.service

```
[Unit]
Description=Stronghold service
Documentation=https://deckhouse.ru/products/stronghold/
After=network.target
```

```
[Service]
Type=simple
ExecStart=/opt/stronghold/stronghold server -config=/opt/stronghold/config.hcl
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=5
User=stronghold
Group=stronghold
LimitNOFILE=65536
CapabilityBoundingSet=CAP_IPC_LOCK
AmbientCapabilities=CAP_IPC_LOCK
SecureBits=noroot

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

2. Выполните команду systemctl daemon-reload.
3. Включите автозапуск сервиса командой systemctl enable stronghold.service.
4. Создайте каталог /opt/stronghold/data и установите для него следующие права доступа:

```
mkdir -p /opt/stronghold/data
chown stronghold:stronghold /opt/stronghold/data
chmod 0700 /opt/stronghold/data
```

3.1.2. Подготовка необходимых сертификатов

Для настройки TLS необходимо наличие следующего набора сертификатов и ключей, расположенных в каталоге /opt/stronghold/tls:

- Сертификат корневого центра сертификации, подписавшего сертификат Stronghold TLS. В данном сценарии он будет иметь имя stronghold-ca.pem.
- Сертификаты узлов Raft. В данном сценарии в кластер будет добавлено три узла, для которых будут созданы следующие сертификаты:
    - node-1-cert.pem;
    - node-2-cert.pem;
    - node-3-cert.pem.
- Закрытые ключи сертификатов узлов:

- node-1-key.pem;
- node-2-key.pem;
- node-3-key.pem.

В этом сценарии будет создан корневой сертификат, а также набор самоподписанных сертификатов для каждого узла.

Хотя самоподписанные сертификаты можно использовать для экспериментов с развертыванием и запуском ПО, настоятельно рекомендуется заменить их сертификатами, созданными и подписанными соответствующим центром сертификации.

Порядок действий:

1. На первом узле перейдите в каталог /opt/stronghold/tls/ (создайте каталог при необходимости):

```
mkdir -p /opt/stronghold/tls
cd /opt/stronghold/tls/
```

2. Сгенерируйте ключ для корневого сертификата:

```
openssl genrsa 2048 > stronghold-ca-key.pem
```

3. Выпустите корневой сертификат:

```
openssl req -new -x509 -nodes -days 3650 -key stronghold-ca-key.pem -out stronghold-ca.pem


Country Name (2 letter code) [XX]:RU
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:Moscow
Organization Name (eg, company) [Default Company Ltd]:MyOrg
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server hostname) []:demo.tld
```

Атрибуты сертификата указаны для примера.

4. Чтобы выпустить сертификаты узлов, создайте конфигурационные файлы, содержащие поле subjectAltName (SAN). Например, для узла raft-node-1 файл будет выглядеть следующим образом:

```
cat << EOF > node-1.cnf
[v3_ca]
subjectAltName = @alt_names
[alt_names]
DNS.1 = raft-node-1.demo.tld
```

```
IP.1 = 10.20.30.10
IP.2 = 127.0.0.1
EOF
```

В поле SAN должны быть указаны корректные FQDN и IP-адрес соответствующего узла.

Создайте отдельный конфигурационный файл для каждого узла, который планируется добавить в кластер.

5. Для каждого узла сформируйте файл запроса:

```
openssl req -newkey rsa:2048 -nodes -keyout node-1-key.pem -out node-1-csr.pem -subj "/CN=raft-node-1.demo.tld"
openssl req -newkey rsa:2048 -nodes -keyout node-2-key.pem -out node-2-csr.pem -subj "/CN=raft-node-2.demo.tld"
openssl req -newkey rsa:2048 -nodes -keyout node-3-key.pem -out node-3-csr.pem -subj "/CN=raft-node-3.demo.tld"
```

6. Выпустите сертификаты на основании запросов:

```
openssl x509 -req -set_serial 01 -days 3650 -in node-1-csr.pem -out node-1-cert.pem -CA stronghold-ca.pem -CAkey stronghold-ca-key.pem -extensions v3_ca -extfile ./node-1.cnf
openssl x509 -req -set_serial 01 -days 3650 -in node-2-csr.pem -out node-2-cert.pem -CA stronghold-ca.pem -CAkey stronghold-ca-key.pem -extensions v3_ca -extfile ./node-2.cnf
openssl x509 -req -set_serial 01 -days 3650 -in node-3-csr.pem -out node-3-cert.pem -CA stronghold-ca.pem -CAkey stronghold-ca-key.pem -extensions v3_ca -extfile ./node-3.cnf
```

Для автоматического подключения узлов скопируйте на каждый узел:

- файл сертификата узла;
- файл ключа узла;
- файл корневого сертификата.

Например:

```
scp ./node-2-key.pem ./node-2-cert.pem ./stronghold-ca.pem  raft-node-2.demo.tld:/opt/stronghold/tls
scp ./node-3-key.pem ./node-3-cert.pem ./stronghold-ca.pem  raft-node-3.demo.tld:/opt/stronghold/tls
```

Если каталог /opt/stronghold/tls на целевых узлах отсутствует, создайте его.

3.1.3. Развертывание кластера с Raft

1. Подключитесь к первому серверу, на котором будет выполняться инициализация кластера ПО «Deckhouse Stronghold».
2. Добавьте разрешающие правила на сетевом экране для TCP-портов 8200 и 8201. Пример для firewalld:

```
firewall-cmd --add-port=8200/tcp --permanent
firewall-cmd --add-port=8201/tcp --permanent
firewall-cmd --reload
```

Вы можете использовать и любые другие порты, указав их в конфигурационном файле /opt/stronghold/config.hcl.

3. Создайте файл /opt/stronghold/config.hcl для конфигурации Raft. Если каталог /etc/stronghold/ не существует, создайте его. Добавьте в файл следующее содержимое, заменив значения соответствующих параметров своими:

```
ui = true
cluster_addr  = "https://10.20.30.10:8201"
api_addr      = "https://10.20.30.10:8200"
disable_mlock = true

listener "tcp" {
 address       = "0.0.0.0:8200"
 tls_cert_file      = "/opt/stronghold/tls/node-1-cert.pem"
 tls_key_file       = "/opt/stronghold/tls/node-1-key.pem"
}

storage "raft" {
 path = "/opt/stronghold/data"
 node_id = "raft-node-1"

 retry_join {
  leader_tls_servername   = "raft-node-1.demo.tld"
  leader_api_addr         = "https://10.20.30.10:8200"
  leader_ca_cert_file     = "/opt/stronghold/tls/stronghold-ca.pem"
  leader_client_cert_file = "/opt/stronghold/tls/node-1-cert.pem"
  leader_client_key_file  = "/opt/stronghold/tls/node-1-key.pem"
 }
 retry_join {
  leader_tls_servername   = "raft-node-2.demo.tld"
  leader_api_addr         = "https://10.20.30.11:8200"
  leader_ca_cert_file     = "/opt/stronghold/tls/stronghold-ca.pem"
  leader_client_cert_file = "/opt/stronghold/tls/node-1-cert.pem"
  leader_client_key_file  = "/opt/stronghold/tls/node-1-key.pem"
 }
 retry_join {
```

```
  leader_tls_servername   = "raft-node-3.demo.tld"
  leader_api_addr         = "https://10.20.30.12:8200"
  leader_ca_cert_file     = "/opt/stronghold/tls/stronghold-ca.pem"
  leader_client_cert_file = "/opt/stronghold/tls/node-1-cert.pem"
  leader_client_key_file  = "/opt/stronghold/tls/node-1-key.pem"
 }
}
```

4. Запустите ПО «Deckhouse Stronghold»:

```
systemctl start stronghold
```

5. Выполните инициализацию кластера ПО «Deckhouse Stronghold»:

```
stronghold operator init -ca-cert /opt/stronghold/tls/stronghold-ca.pem
```

Вы можете передать параметры -key-shares и -key-threshold, чтобы определить, на сколько частей будет разбит ключ и сколько из них будет достаточно для распечатывания хранилища. По умолчанию key-shares=5, a key-threshold=3.

6. Чтобы распечатать кластер, выполните необходимое количество раз команду: stronghold operator unseal -ca-cert /opt/stronghold/tls/stronghold-ca.pem
   и введите ключи распечатки, полученные на предыдущем шаге. Если вы не меняли параметр -key-threshold, то ввести нужно 3 части ключа.
7. Повторите настройку на остальных узлах кластера. Для этого укажите в файле /opt/stronghold/config.hcl в параметрах cluster_addr и api_addr соответствующие IP-адреса узлов. Пропустите шаг с инициализацией и сразу переходите к шагу распечатки кластера.
8. Проверьте работу кластера:

```
stronghold status -ca-cert /opt/stronghold/tls/stronghold-ca.pem
Key                Value
---                -----
Seal Type          shamir
Initialized        true
Sealed             false
Total Shares       5
Threshold          3
Version            1.15.2
Build Date         2025-03-07T16:10:46Z
```

```
Storage Type          raft
Cluster Name          stronghold-cluster-a3fcc270
Cluster ID            f682968d-5e6c-9ad4-8303-5aecb259ca0b
HA Enabled            true
HA Cluster            https://10.20.30.10:8201
HA Mode               active
Active Node Address   https://10.20.30.10:8200
Raft Committed Index  40
Raft Applied Index    40
```

3.2. Поддержка HSM режима

ПО «Deckhouse Stronghold» поддерживает шифрование ключей с помощью аппаратных модулей шифрования, таких как TPM2, Rutoken ЭЦП 3.0, JaCarta и т.д., имеющих поддержку pkcs11. Также поддерживаются SoftHSM.

Для использования автоматического распечатывания через pkcs11 потребуется создать ключи в HSM и сконфигурировать ПО «Deckhouse Stronghold» на использование этих ключей.

3.2.1. Предварительная настройка

Для HSM требуются следующие программные пакеты, не входящие в поставку ПО «Deckhouse Stronghold» и являющиеся частью среды исполнения:

- библиотека интеграции HSM, совместимая с pkcs11. ПО «Deckhouse Stronghold» ориентировано на версию PKCS#11 2.2 или выше. В зависимости от конкретного HSM некоторые функции могут выполняться вручную;
- библиотека [GNU libltdl] (https://www.gnu.org/software/libtool/manual/html_node/Using-libltdl.html) — перед подключением необходимо проверить совместимость с архитектурой серверов, на которые утсанавливается библиотека.

В примере ниже показана настройка распечатки с помощью HSM pkcs11 через файл конфигурации ПО «Deckhouse Stronghold» с указанием всех требуемых значений:

```
```
seal "pkcs11" {
  lib = "/usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so"
  token_label = "my_token"
  pin = "4321"
  key_label = "my-rsa-key"
```
```

```
 rsa_oaep_hash = "sha1"
}
```

3.2.2. Параметры `pkcs11`

Эти параметры относятся к разделу `seal` в файле конфигурации ПО «Deckhouse Stronghold»:

- `lib` `(string: <required>)`: Путь к файлу общего объекта библиотеки PKCS#11. Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_LIB`. (В зависимости от HSM, значение параметра `lib`может быть либо двоичным значением, либо динамической библиотекой, и для его использования могут потребоваться другие библиотеки в зависимости от используемого типа HSM);
- `slot` `(string: <slot or token label required>)`: Номер слота для использования, указанный как строка (например `"2305843009213693953"`). Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_SLOT`(слоты как правило перечислены как шестнадцатерично-десятичные значения в утилите настройки ОС, но эта конфигурация использует их десятичный эквивалент. Например, при использовании командной строки HSM `pkcs11-tool`слот, перечисленный как `0x2000000000000001`в шестнадцатеричном формате, эквивалентен `2305843009213693953`в десятичном формате; эти значения могут быть перечислены короче или отличаться в зависимости от используемого HSM);
- `token_label` `(string: <slot or token label required>)`: Метка токена слота для использования. Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_TOKEN_LABEL`;
- `pin` `(string: <required>)`: PIN-код для входа. Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_PIN`. _Если задано через переменную среды, его необходимо будет переустановить при перезапуске ПО «Deckhouse Stronghold».
- `key_label` `(string: <required>)`: Метка используемого ключа. Если ключ не существует и генерация включена, это метка, которая будет присвоена сгенерированному ключу. Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_KEY_LABEL`;
- `key_id` `(string: "")`: Идентификатор используемого ключа. Значение должно быть шестнадцатеричной строкой (например, "0x33333435363434373537"). Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_KEY_ID`;

- `hmac_key_label` `(string: <required>)`: Метка ключа для использования при хешировании HMAC. Если Stronghold попытается создать его, он вызовет функцию CKK_GENERIC_SECRET_KEY. Если ключ не существует и генерация включена, это метка, которая будет присвоена сгенерированному ключу. Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_HMAC_KEY_LABEL`;
- `hmac_key_id` `(string: "")`: Идентификатор ключа HMAC для использования. Значение должно быть шестнадцатеричной строкой (например, "0x33333435363434373537"). Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_HMAC_KEY_ID`;
- `mechanism` `(string: <best available>)`: Механизм зашифровывания/расшифровывания для использования, указанный как десятичная или шестнадцатеричная (с префиксом `0x`) строка. Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_MECHANISM`. В настоящее время поддерживаемые механизмы (в порядке приоритета):
    - `0x1085` `CKM_AES_CBC_PAD`(Требуется механизм HMAC)
    - `0x1082` `CKM_AES_CBC`(Требуется механизм HMAC)
    - `0x1087` `CKM_AES_GCM`
    - `0x0009` `CKM_RSA_PKCS_OAEP`
    - `0x0001` `CKM_RSA_PKCS`

  CKM_RSA_PKCS указывает схему заполнения PKCS #1 v1.5, которая считается менее безопасной, чем OAEP. По возможности рекомендуется использовать CKM_RSA_PKCS_OAEP вместо CKM_RSA_PKCS.
- `hmac_mechanism` `(string: "0x0251")`: Механизм зашифровывания/расшифровывания, указанный как десятичная или шестнадцатеричная (с префиксом `0x`) строка. В настоящее время поддерживается только `0x0251`(соответствует `CKM_SHA256_HMAC`из спецификации). Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_HMAC_MECHANISM`. Это значение требуется только для определенных механизмов;
- `generate_key` `(string: "false")`: Если во время инициализации ПО «Deckhouse Stronghold» не сможет обнаружить существующий ключ с меткой, указанной в `key_label`, он дает команду сгенерировать ключ. Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_GENERATE_KEY`. ПО «Deckhouse Stronghold» может не иметь возможности успешно генерировать ключи при любых

обстоятельствах, например, если для создания ключей подходящего типа требуются фирменные расширения поставщика (после создания первоначального ключа после инициализации кластера рекомендуется отключить этот флаг, чтобы предотвратить непреднамеренное создание ключей в будущем);

- `force_rw_session` `(string: "false")`: Принудительно открывать сеанс чтения-записи для HSM для всех операций. Это логическое значение, выраженное в виде строки (например, `"true"`). Может также задаваться с помощью переменной окружения `STRONGHOLD_HSM_FORCE_RW_SESSION`;
- `max_parallel` `(int: 1)`- Количество одновременных запросов, которые могут поступать в HSM в любой момент времени;
- `disabled` `(string: "")`: Установите это значение, `true` если вы проводите миграцию из конфигурации автоматической распечатки. В противном случае установите значение `false`.

3.2.3. SoftHSM2

Установка libsofthsm2 и pkcs11-tool осуществляется с использованием следующей команды:

```
apt install libsofthsm2 opensc
```

Далее необходимо создать конфигурацию для softhsm2:

```
mkdir /home/stronghold/softhsm
cd softhsm
echo "directories.tokendir = /home/stronghold/softhsm/" > /home/stronghold/softhsm2.conf
```

Создание ключа в HSM осуществляется следующим образом:

```
$ export SOFTHSM2_CONF=/home/stronghold/softhsm2.conf
$ HSMLIB="/usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so"
$ pkcs11-tool --module $HSMLIB --init-token --so-pin 1234 --init-pin --pin 4321 --label my_token --login
Using slot 0 with a present token (0x0)
Token successfully initialized
User PIN successfully initialized
$ pkcs11-tool --module $HSMLIB -L
Available slots:
Slot 0 (0xe6829d3): SoftHSM slot ID 0xe6829d3
  token label        : my_token
  token manufacturer : SoftHSM project
```

```
  token model       : SoftHSM v2
  token flags       : login required, rng, token initialized, PIN initialized, other flags=0x20
  hardware version  : 2.6
  firmware version  : 2.6
  serial num        : 6a5468368e6829d3
  pin min/max       : 4/255
Slot 1 (0x1): SoftHSM slot ID 0x1
  token state:   uninitialized
$ pkcs11-tool --module $HSMLIB --login --pin 4321 --keypairgen --key-type rsa:4096 --label "vault-rsa-key"
Using slot 0 with a present token (0xe6829d3)
Key pair generated:
Private Key Object; RSA
  label:      vault-rsa-key
  Usage:      decrypt, sign, signRecover, unwrap
  Access:     sensitive, always sensitive, never extractable, local
Public Key Object; RSA 4096 bits
  label:      vault-rsa-key
  Usage:      encrypt, verify, verifyRecover, wrap
  Access:     local
```

Пример конфигурации ПО «Deckhouse Stronghold» (config.hcl)

```
api_addr="https://0.0.0.0:8200"
log_level = "warn"
ui = true
listener "tcp" {
  address = "0.0.0.0:8200"
  tls_cert_file = "/home/stronghold/cert.pem"
  tls_key_file  = "/home/stronghold/key.pem"
  #tls_require_and_verify_client_cert = true
  #tls_client_ca_file = "ca.crt"
  tls_disable = "false"
}
storage "raft" {
  path = "/home/stronghold/data"
}
seal "pkcs11" {
```

```
  lib = "/usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so"
  token_label = "my_token"
  pin = "4321"
  key_label = "vault-rsa-key"
  rsa_oaep_hash = "sha1"
}
```

Запуск ПО «Deckhouse Stronghold»

```
export SOFTHSM2_CONF=/home/stronghold/softhsm2.conf
stronghold server -config config.hcl
```

3.2.4. Использование Rutoken ЭЦП 3.0

Для использования Rutoken необходимо убедиться, что установлена библиотека librtpkcs11ecp.so. Далее необходимо сгенерировать в токене публичный и приватный ключи, с помощью которых будет проводиться шифрование Root-ключа. Эту операцию можно провести с помощью утилиты pkcs11-tool:

```
$ HSMLIB="/usr/lib/librtpkcs11ecp.so"
$ pkcs11-tool --module $HSMLIB --init-token --so-pin 87654321 \
        --init-pin --pin 12345678 --label my_token --login
$ pkcs11-tool --module $HSMLIB --login --pin 12345678 --keypairgen \
        --key-type rsa:2048 --label "vault-rsa-key"
Using slot 0 with a present token (0x0)
Key pair generated:
Private Key Object; RSA
  label:      vault-rsa-key
  Usage:      decrypt, sign
  Access:     sensitive, always sensitive, never extractable, local
Public Key Object; RSA 2048 bits
  label:      vault-rsa-key
  Usage:      encrypt, verify
  Access:     local
```

После нужно добавить в конфигурацию Stronghold метод распечатки pkcs11

```
...
seal "pkcs11" {
  lib = "/usr/lib/librtpkcs11ecp.so"
```

```
  token_label = "my_token"
  pin = "12345678"
  key_label = "vault-rsa-key"
}
```

Для запуска ПО «Deckhouse Stronghold» используется оператор init:

```
systemctl start stronghold
stronghold operator init
```

Пример отображения статуса:

```
stronghold status
Key                    Value
---                    -----
Recovery Seal Type     shamir
Initialized            true
Sealed                 false
Total Recovery Shares  5
Threshold              3
Version                1.15.2+hsm
Build Date             2025-04-03T13:06:02Z
Storage Type           raft
Cluster Name           stronghold-cluster-6586e287
Cluster ID             d7552773-2e8a-33b6-9c32-6749a4c9af13
HA Enabled             false
```

4. Описание параметров (настроек) безопасности

4.1. Настройка конфигурации

ПО «Deckhouse Stronghold» настраивается с использованием конфигурационного файла в формате HCL или JSON.

Чтобы усилить контроль за доступом к файлам конфигурации, можно включить проверку прав доступа к файлам с помощью переменной окружения VAULT_ENABLE_FILE_PERMISSIONS_CHECK. Если проверка активирована, ПО «Deckhouse Stronghold» проверяет, принадлежат ли директория конфигурации и файлы пользователю, который запускает ПО. Дополнительно проверяется, что у группы или других пользователей нет прав на запись или выполнение этих файлов.

При необходимости администратор может указать пользователя и права доступа к директории плагинов и исполняемым файлам. Для этого используются параметры plugin_file_uid и plugin_file_permissions в конфигурации. По умолчанию проверка прав в ПО отключена.

Пример конфигурации:

```
ui            = true
cluster_addr  = "https://127.0.0.1:8201"
api_addr      = "https://127.0.0.1:8200"
disable_mlock = true


storage "raft" {
  path = "/path/to/raft/data"
  node_id = "raft_node_id"
}


listener "tcp" {
  address       = "127.0.0.1:8200"
  tls_cert_file = "/path/to/full-chain.pem"
  tls_key_file  = "/path/to/private-key.pem"
}


telemetry {
  statsite_address = "127.0.0.1:8125"
  disable_hostname = true
}
```

Для применения новых параметров после изменения файла конфигурации нужно перезапустить сервис ПО командой systemctl restart stronghold.

4.1.1. Описание параметров конфигурации

- storage — обязательный блок параметров. Настройка бэкенда хранилища, где будут сохраняться данные ПО. Для работы ПО «Deckhouse Stronghold» в режиме высокой доступности (HA) нужно использовать бэкенд, который поддерживает семантику координации. Если выбранный бэкенд хранения это позволяет, параметры HA можно указать прямо внутри блока storage. В ином случае следует настроить отдельный параметр ha_storage с бэкендом, который поддерживает HA, вместе с соответствующими параметрами HA. Подробности о параметрах бэкенда хранилища — в разделе 4.3;
- ha_storage — необязательный блок параметров. Настройка бэкенда хранилища, где будет происходить координация ПО в режиме высокой доступности (HA). Указанный бэкенд должен поддерживать HA. Если параметр не установлен, будет выполнена попытка запустить HA на бэкенде, который указан в параметре storage. Если бэкенд хранилища уже поддерживает координацию HA и если специфические параметры HA уже указаны в блоке storage, дополнительная настройка ha_storage не требуется;
- listener — обязательный блок параметров. Настройка параметров прослушивания запросов API ПО. Подробности — в разделе 4.2;
- user_lockout — необязательный блок параметров. Настройка поведения блокировки пользователя при неудачном входе в систему. Подробности — в разделе 4.5;
- cluster_name — необязательный строковый параметр. Указывает идентификатор для кластера ПО «Deckhouse Stronghold». Если значение не указано, оно будет автоматически сгенерировано;
- cache_size — необязательный строковый параметр. Определяет размер кэша чтения, который использует физическая подсистема хранения. Значение указывается в количестве хранимых записей, поэтому общий размер кэша зависит от их размера. По умолчанию 131072;
- disable_cache — необязательный логический параметр. Отключает все кэши в ПО, в том числе кэш чтения, который использует физическая подсистема хранения. Сильно влияет на производительность. По умолчанию false;

- disable_mlock — необязательный логический параметр. Отключает возможность сервера выполнять системный вызов mlock, который предотвращает выгрузку памяти на диск.

Не рекомендуется отключать mlock, если не используется интегрированное хранилище. При отключении mlock стоит соблюдать описанные ниже дополнительные меры безопасности.

При использовании интегрированного хранилища настоятельно рекомендуется отключить mlock. Этот системный вызов плохо совместим с отображаемыми в памяти файлами, такими как те, что создаются в базе данных BoltDB, которую Raft использует для отслеживания состояния.

Использование mlock с интегрированным хранилищем может вызвать нехватку памяти, если объем данных ПО «Deckhouse Stronghold» превышает доступный объем ОЗУ. Дело в том, что отображаемые в память файлы загружаются в резидентную память, а это приводит к загрузке всех данных ПО «Deckhouse Stronghold» в оперативную память. В этом случае, несмотря на то, что данные в BoltDB остаются зашифрованными в состоянии покоя, swap нужно отключить. Это позволит предотвратить выгрузку на диск других конфиденциальных данных ПО, которые находятся в памяти.

В ОС можно дать исполняемому файлу ПО возможность использовать системный вызов mlock без запуска процесса от имени root, выполнив следующую команду:

```
sudo setcap cap_ipc_lock=+ep $(readlink -f $(which stronghold))
```

Каждый плагин запускается как отдельный процесс, поэтому нужно применить аналогичные настройки для каждого плагина в директории plugins. Если вы используете дистрибутив Linux с актуальной версией systemd, то можете добавить следующую директиву в секцию [Service] конфигурационного файла:

```
LimitMEMLOCK=infinity
```

- plugin_directory — необязательный строковый параметр. Определяет каталог, из которого разрешено загружать плагины. Для успешной загрузки плагинов ПО должен иметь разрешение на чтение файлов в этой директории, а значение не может быть символической ссылкой. По умолчанию параметр не задан и имеет значение "";
- plugin_tmpdir — необязательный строковый параметр. Указывает каталог, где ПО может создавать временные файлы для поддержки взаимодействия Unix-сокета с контейнеризированными плагинами. Если значение не задано, ПО «Deckhouse Stronghold» будет использовать каталог по умолчанию для временных файлов. В большинстве случаев настройка этого параметра не требуется, за исключением ситуаций, когда используются контейнеризированные плагины и ПО

разделяет временную папку с другими процессами, например при использовании параметра PrivateTmp в systemd;

- plugin_file_uid — необязательный целочисленный параметр. Идентификатор пользователя (UID) директорий плагинов и исполняемых файлов плагинов в случае, если они принадлежат пользователю, отличному от того, кто запускает ПО «Deckhouse Stronghold»;
- plugin_file_permissions — необязательный строковый параметр. Строка восьмеричных прав доступа для директорий плагинов и исполняемых файлов плагинов в случае, если установлены права на запись или выполнение для группы или других пользователей;
- telemetry — необязательный блок параметров. Указывает систему телеметрии для сбора и отправки статистики;
- default_lease_ttl — необязательный строковый параметр. Определяет срок действия аренды для токенов и секретов по умолчанию. Значение указывается с использованием суффикса времени, например, "40s" (40 секунд) или "1h" (1 час), и не может быть больше, чем max_lease_ttl. По умолчанию 768h;
- max_lease_ttl — необязательный строковый параметр. Определяет максимальный срок действия аренды для токенов и секретов. Значение указывается с использованием суффикса времени, например, "40s" (40 секунд) или "1h" (1 час). Отдельные точки монтирования могут переопределить это значение, настроив точку монтирования с помощью флага max-lease-ttl в командах auth или secret. По умолчанию 768h;
- default_max_request_duration — необязательный строковый параметр. Указывает максимальное стандартное время выполнения запроса, после которого ПО его отменит. Значение указывается с использованием суффикса времени, например, "40s" (40 секунд) или "1h" (1 час) и может быть переопределено для каждого слушателя (listener) через параметр max_request_duration. По умолчанию 90s;
- detect_deadlocks — необязательный строковый параметр. Строка значений, разделенных запятыми. Указывает внутренние взаимоисключающие блокировки, за которыми необходимо следить на предмет потенциальных взаимоблокировок. Поддерживаются значения statelock, quotas и expiration, что приведет к записи "POTENTIAL DEADLOCK:" в лог , если попытка блокировки состояния ядра кажется заблокированной. Включение detect_deadlocks может негативно повлиять на производительность из-за отслеживания каждой попытки блокировки. По умолчанию параметр не задан и имеет значение "";

- raw_storage_endpoint — необязательный логический параметр. Активирует эндпойнт sys/raw с высоким уровнем привилегий. Этот эндпойнт позволяет выполнять дешифрование и шифрование необработанных данных на входе и выходе из защитного барьера. По умолчанию false;
- introspection_endpoint — необязательный логический параметр. Активирует эндпойнт sys/internal/inspect, который позволяет проводить инспекцию определенных подсистем внутри ПО пользователям с root-токеном или привилегиями sudo. По умолчанию false;
- ui — необязательный логический параметр. Активирует встроенный веб-интерфейс пользователя, доступный на всех слушателях (адрес + порт) по пути /ui. При обращении браузеров к стандартному адресу API ПО они будут автоматически перенаправлены на интерфейс. По умолчанию false. Подробности — в разделе 4.4;
- pid_file — необязательный строковый параметр. Указывает путь к файлу, где должен храниться идентификатор процесса (PID) сервера ПО;
- enable_response_header_hostname — необязательный логический параметр. Активирует добавление HTTP-заголовка X-Vault-Hostname во все HTTP-ответы ПО «Deckhouse Stronghold». Этот заголовок будет содержать имя узла ПО «Deckhouse Stronghold», который обработал HTTP-запрос. Наличие этой информации не гарантируется — она будет предоставлена по возможности. Если опция включена, но заголовок X-Vault-Hostname отсутствует в ответе, это может указывать на ошибку при извлечении имени хоста из операционной системы. По умолчанию false;
- enable_response_header_raft_node_id — необязательный логический параметр. Активирует добавление HTTP-заголовка X-Vault-Raft-Node-ID во все HTTP-ответы ПО. Если ПО использует интегрированное хранилище (то есть участвует в кластере Raft), заголовок X-Vault-Raft-Node-ID будет содержать идентификатор узла Raft, который обработал HTTP-запрос. Если же узел ПО «Deckhouse Stronghold» не участвует в кластере Raft, этот заголовок будет опущен независимо от того, включена ли опция. По умолчанию false;
- log_level — необязательный строковый параметр. Определяет уровень детализации журнала безопасности. Поддерживаются следующие пять значений в порядке убывания детализации: trace, debug, info, warn и error. Это значение также можно задать с помощью переменной окружения VAULT_LOG_LEVEL. По умолчанию info.

Если указано корректное значение параметра log_level, при SIGHUP (sudo kill -s HUP pid Stronghold), ПО «Deckhouse Stronghold» обновит существующий уровень журнала. При этом будут

отменены как флаг CLI, так и переменная окружения. Динамически изменять таким образом уровень журнала могут не все части журнала ПО «Deckhouse Stronghold». Например, не обновляются динамически плагины secrets/auth.

- log_format — необязательный строковый параметр. Формат журнала. Поддерживаются два значения: standard и json. По умолчанию standard;
- log_file — необязательный строковый параметр. Абсолютный путь, по которому ПО «Deckhouse Stronghold» должно сохранять сообщения журнала в дополнение к другим существующим выводам, таким как journald/stdout. Пути, которые заканчиваются разделителем пути, используют имя файла по умолчанию (vault.log). Пути, которые не заканчиваются расширением файла, используют расширение по умолчанию (.log). Если файл журнала перезаписывается, в момент перезаписи ПО добавляет к имени файла текущую временную метку;
- log_rotate_duration — необязательный строковый параметр. Указывает максимальную продолжительность записи в файл журнала до его перезаписи. Значение указывается с использованием суффикса времени, например, 40s. По умолчанию 24h;
- log_rotate_bytes — необязательный целочисленный параметр. Указывает количество байт, которое можно записать в файл журнала до его перезаписи. Если значение не указано, то количество байт, которое можно записать в файл журнала, не ограничено;
- log_rotate_max_files — необязательный целочисленный параметр. Указывает максимальное количество старых файлов журнала, которые нужно сохранять. По умолчанию установлено значение 0, то есть файлы никогда не удаляются. Чтобы удалять старые файлы журналы при создании нового установите значение -1;
- imprecise_lease_role_tracking — необязательный логический параметр. Позволяет пропустить подсчет аренды по ролям, если квоты на основе ролей не включены. Когда этот параметр установлен в значение true и включена новая квота на основе ролей, последующий подсчет аренд начнется с нуля. Параметр влияет на квоты подсчета аренды на основе ролей, но уменьшает задержки, если квоты на основе роли не используются;
- experiments — необязательный массив значений. Список экспериментальных функций, которые следует активировать для узла. Не используйте экспериментальные функции в production-среде. Связанные с ними API могут претерпевать несовместимые с предыдущими версиями изменения между релизами. Дополнительные

экспериментальные функции также можно указать через переменную окружения VAULT_EXPERIMENTS в виде списка значений, разделенных запятыми.

4.1.2. Параметры высокой доступности

Для бэкендов, которые поддерживают режим высокой доступности (HA), используются следующие параметры:

- api_addr — необязательный строковый параметр. Указывает адрес, который будет анонсироваться другим серверам ПО в кластере для перенаправления клиентов. Также это значение используется для бэкендов плагинов. Параметр api_addr также можно задать через переменную окружения VAULT_API_ADDR. В общем случае это должен быть полный URL, который указывает на значение адреса слушателя (listener). Адрес может быть динамически определен с помощью шаблона go-sockaddr, который разрешается во время выполнения;
- cluster_addr — необязательный строковый параметр. Указывает адрес для анонсирования другим серверам ПО в кластере для перенаправления запросов. Этот параметр также можно задать через переменную окружения VAULT_CLUSTER_ADDR. Подобно api_addr это полный URL, но ПО «Deckhouse Stronghold» будет игнорировать схему (все участники кластера всегда используют TLS с приватным ключом/сертификатом). Адрес может быть динамически определен с помощью шаблона go-sockaddr, который разрешается во время выполнения;
- disable_clustering — необязательный логический параметр. Указывает, включены ли функции кластеризации, например, переадресация запросов. Если установить значение true для одного узла хранилища, функции кластеризации будут отключены только в случае, если этот узел является активным. Параметр нельзя установить в true, если типом хранилища является raft. По умолчанию false.

4.2. Проверка настроек конфигурации

После выполнения настроек конфигурации необходимо выполнить их корректность. Для проверки корректности настроек конфигурации необходимо выполнить следующие проверки:

- Убедитесь, что параметр introspection_endpoint=false (по умолчанию) или не указан;
- Убедитесь, что параметр raw_storage_endpoint=false (по умолчанию) или не указан;
- Убедитесь, что для публичных listeners параметр tls_disable=0 (по умолчанию) или не указан;

- Убедитесь, что параметр unauthenticated_pprof_access=false (по умолчанию) или не указан;
- Убедитесь, что параметр unauthenticated_in_flight_requests_access=false (по умолчанию) или не указан;
- Убедитесь, что параметр disable_lockout=false (по умолчанию) или не указан;
- Убедитесь, что параметр experiments не задан и переменная VAULT_EXPERIMENTS не установлена;
- Убедитесь, что в случае включенения unauthenticated_metrics_access для telemetry адрес для доступа недоступен публично;
- Убедитесь, что параметр disable_cache=false (по умолчанию) или не задан;
- Убедитесь, что если в вашей системе используется swap, то параметр disable_mlock=false (по умолчанию) или не задан.

4.3. Секция listener

Секция listener настраивает адреса и порты, на которых ПО будет отвечать на запросы. Существует два типа слушателей:

1. TCP.
2. Unix Domain Socket.

4.3.1. TCP

TCP-слушатель настраивает ПО «Deckhouse Stronghold» на прослушивание TCP-адреса/порта.

```
listener "tcp" {
  address = "127.0.0.1:8200"
}
```

Можно указать секцию listener больше одного раза, чтобы ПО «Deckhouse Stronghold» прослушивало несколько интерфейсов. При настройке нескольких слушателей обязательно указывайте параметры api_addr и cluster_addr, чтобы ПО «Deckhouse Stronghold» анонсировало другим узлам корректный адрес.

4.3.2. Сокрытие конфиденциальных данных для неаутентифицированных эндпойнтов

Неаутентифицированные API-эндпойнты могут возвращать следующую конфиденциальную информацию:

- номер версии ПО;

- дату сборки бинарного файла ПО;
- имя кластера ПО;
- IP-адрес узлов в кластере.

При этом, ПО «Deckhouse Stronghold» позволяет настроить каждую секцию tcp listener так, чтобы при необходимости удалить эти данные из ответов API. Удаление чувствительной информации на основе конфигурации секции listener поддерживают три эндпойнта API:

- /sys/health;
- /sys/leader;
- /sys/seal-status.

Удаленную информацию ПО «Deckhouse Stronghold» заменяет пустой строкой — "". Также некоторые API ПО «Deckhouse Stronghold» опускают ключи в ответе, если соответствующее значение пустое ("").

4.3.3. Пользовательские заголовки ответов

ПО «Deckhouse Stronghold» позволяет задавать пользовательские HTTP-заголовки ответов для корневого пути (/) и для эндпойнтов API (/v1/). Эти заголовки определяются на основе возвращаемого кода состояния. Например, администратор может определить один список пользовательских заголовков ответов для кода состояния 200 и другой список для кода состояния 307.

Эндпойнт API /sys/config/ui позволяет пользователям устанавливать специфические для UI пользовательские заголовки. Однако если заголовок настроен в конфигурационном файле, перенастроить его через данный эндпойнт нельзя. Чтобы удалить пользовательский заголовок или изменить его значение, нужно изменить файл конфигурации ПО «Deckhouse Stronghold» и отправить сигнал SIGHUP в процесс ПО.

Если в конфигурационном файле задан пользовательский заголовок и такой же заголовок используют внутренние процессы ПО, настроенный заголовок не будет принят. Например, не будет принят пользовательский заголовок с префиксом X-Vault-. При запуске в журналах ПО «Deckhouse Stronghold» будет зарегистрировано соответствующее сообщение.

4.3.4. Порядок приоритетов

Если один и тот же заголовок указан в конфигурационном файле и через эндпойнт API /sys/config/ui, приоритет отдается заголовку из конфигурационного файла. Например, заголовок Content-Security-Policy по умолчанию определен в эндпойнте API /sys/config/ui. Однако если этот

же заголовок прописан в конфигурационном файле, ПО «Deckhouse Stronghold» использует именно его, подставляя в ответ вместо значения по умолчанию в /sys/config/ui.

4.3.5. Параметры слушателя TCP

- address — строковый параметр в формате IP-адрес:порт. Указывает назначенные для прослушивания адрес и порт. Параметр может быть динамически определен с помощью шаблона go-sockaddr, который разрешается во время выполнения;
- cluster_address — строковый параметр в формате IP-адрес:порт. Указывает адрес и порт, назначенные для прослушивания запросов типа сервер-сервер в кластере. По умолчанию значение параметра на один порт выше значения address. В большинстве случаев задавать cluster_address не нужно, однако параметр может быть полезен, если изолированным серверам ПО «Deckhouse Stronghold» нужно обходить TCP-балансировщик нагрузки или использовать какую-то другую схему для связи. Параметр может быть динамически определен с помощью шаблона go-sockaddr, который разрешается во время выполнения;
- http_idle_timeout — строковый параметр. Указывает максимальное время ожидания следующего запроса при включении режима keep-alives. Значение задается с помощью суффикса времени, например, "40s" (40 секунд) или "1h" (1 час). Если значение параметра равно нулю, будет использоваться значение http_read_timeout. Если же оба этих значения равны нулю, будет использоваться значение http_read_header_timeout;
- http_read_header_timeout — строковый параметр. Указывает время, отведенное на чтение заголовков запроса. Значение задается с помощью суффикса времени, например, "40s" (40 секунд) или "1h" (1 час);
- http_read_timeout — строковый параметр. Указывает максимальную продолжительность чтения всего запроса, включая заголовки и тело. Значение задается с помощью суффикса времени, например, "40s" (40 секунд) или "1h" (1 час);
- http_write_timeout — строковый параметр. Указывает максимальную продолжительность записи ответа. Сбрасывается каждый раз, когда считывается заголовок нового запроса. Значение по умолчанию (0) означает отсутствие ограничения по времени. Задается с помощью суффикса времени, например, "40s" (40 секунд) или "1h" (1 час);

- max_request_size — целочисленный параметр. Указывает максимально допустимый размер запроса в байтах. По умолчанию равен 32 МБ, если значение не задано или задан 0. Значение меньше 0 отключает ограничение;
- max_request_duration — строковый параметр. Указывает максимальное время обработки запроса, после которого ПО «Deckhouse Stronghold» отклонит запрос. Этот параметр отменяет значение default_max_request_duration для данного слушателя. Задается с помощью суффикса времени, например, "90s" (90 секунд);
- proxy_protocol_behavior — строковый параметр. Включает поведение протокола PROXY версии 1 для слушателя. Принимаются следующие значения:
    - use_always — всегда будет использоваться IP-адрес клиента;
    - allow_authorized — будет использоваться IP-адрес клиента, если исходный IP-адрес находится в списке;
    - proxy_protocol_authorized_addrs — будет использоваться исходный IP-адрес, если исходного IP в списке нет;
    - deny_unauthorized — если исходный IP-адрес не в списке proxy_protocol_authorized_addrs, трафик будет отклонен;
    - proxy_protocol_authorized_addrs — строковый параметр или массив строк. Определяет список разрешенных IP-адресов источников для использования с протоколом PROXY.

Не требуется, если для параметра proxy_protocol_behavior установлено значение use_always. При указании в формате строки значения нужно разделить запятыми. Значение proxy_protocol_authorized_addrs не может быть пустым: нужно указать как минимум один IP-адрес источника.

- redact_addresses — логический параметр. Скрывает значения leader_address и cluster_leader_address в соответствующих ответах API, если установлен в значение true;
- redact_cluster_name — логический параметр. Скрывает значение cluster_name в соответствующих ответах API, если установлен в значение true;
- redact_version — логический параметр. Скрывает значения version и build_date в соответствующих ответах API, если установлен в значение true;
- tls_disable — логический параметр. Указывает, будет ли отключен TLS. ПО «Deckhouse Stronghold» использует TLS по умолчанию, поэтому необходимо явно отключить TLS, чтобы отказаться от небезопасной связи. Отключение TLS может привести к недоступности некоторых функций пользовательского интерфейса;

- tls_cert_file — строковый параметр. Указывает путь к файлу сертификата для TLS. Файл должен быть в кодировке PEM. Для настройки слушателя на использование сертификата ЦС нужно объединить основной сертификат и сертификат ЦС. При этом в начале объединенного файла должен находиться основной сертификат. Указанный в параметре tls_cert_file путь используется при запуске ПО. Изменение значения во время работы ПО «Deckhouse Stronghold» не приводит к изменениям в работе системы;
- tls_key_file — строковый параметр. Указывает путь к закрытому ключу для сертификата. Файл должен быть в кодировке PEM. Если файл ключа зашифрован, при запуске сервера потребуется ввести парольную фразу. При перезагрузке конфигурации с помощью SIGHUP парольная фраза между файлами ключей должна оставаться неизменной. Указанный в параметре tls_key_file путь используется при запуске ПО. Изменение значения во время работы ПО не приводит к изменениям в работе системы;
- tls_min_version — строковый параметр. Указывает минимальную версию TLS, поддерживаемую слушателем. Допустимые значения: "tls10", "tls11", "tls12" или "tls13". TLS 1.1 и ниже, то есть значения tls10 и tls11, указанные в параметре tls_min_version, считаются небезопасными и не рекомендуются;
- tls_max_version — строковый параметр. Указывает максимальную версию TLS, поддерживаемую слушателем. Допустимые значения: "tls10", "tls11", "tls12" или "tls13". TLS 1.1 и ниже, то есть значения tls10 и tls11, указанные в параметре tls_max_version, считаются небезопасными и не рекомендуются;
- tls_cipher_suites — строковый параметр. Указывает список поддерживаемых наборов шифров в виде списка значений, перечисленных через запятую. Список всех доступных наборов шифров можно найти в документации TLS для Go.

Для эффективности tls_cipher_suites параметр tls_max_version нужно установить в "tls12", чтобы предотвратить согласование TLSv1.3.

- tls_require_and_verify_client_cert — логический параметр. Включает аутентификацию клиента для данного слушателя. При этом слушателю понадобится клиентский сертификат, успешно проверенный на системных ЦС;
- tls_client_ca_file — строковый параметр. Файл сертификата ЦС в кодировке PEM, который используется для проверки подлинности клиента;
- tls_disable_client_certs — логический параметр. Отключает аутентификацию клиента для данного слушателя. По умолчанию — при установленном значении false —

ПО «Deckhouse Stronghold» запрашивает сертификаты аутентификации клиента, когда они доступны;

- x_forwarded_for_authorized_addrs — строковый параметр. Может быть указан в формате строки со списком разделенных запятыми значений или в формате JSON-массива. Указывает список CIDR исходных IP-адресов, для которых заголовок X-Forwarded-For будет доверенным. Включает поддержку X-Forwarded-For.

Например, ПО «Deckhouse Stronghold» получает соединение от IP-адреса балансировщика нагрузки 1.2.3.4. Добавление 1.2.3.4 в параметр x_forwarded_for_authorized_addrs приведет к тому, что в поле remote_address журнала аудита будет отображен IP-адрес клиента, инициировавшего подключение, например 3.4.5.6. Важно, чтобы балансировщик нагрузки отправлял IP-адрес подключающегося клиента в заголовке X-Forwarded-For.

- x_forwarded_for_hop_skips — строковый параметр. Указывает количество адресов, которые нужно пропустить с конца набора переходов. Например, если в заголовке X-Forwarded-For указаны адреса 1.2.3.4, 2.3.4.5, 3.4.5.6, 4.5.6.7, и параметр x_forwarded_for_hop_skips установлен в значение "1", то в качестве IP-адреса исходящего клиента будет использован 3.4.5.6;
- x_forwarded_for_reject_not_authorized — логический параметр. При установке значения false позволяет игнорировать заголовок X-Forwarded-For, если он поступает в соединении с неавторизованного адреса. Клиентское соединение в этом случае будет использоваться как есть, а не будет отклонено;
- x_forwarded_for_reject_not_present — логический параметр. При установке значения false позволяет использовать адрес клиента как есть, если заголовок X-Forwarded-For отсутствует или пуст, вместо того чтобы отклонить соединение с клиентом;
- disable_replication_status_endpoints — логический параметр. При установке значения true отключит эндпойнты статуса репликации для данного слушателя.

Параметры telemetry:

- unauthenticated_metrics_access — логический параметр. При установке значения true разрешает неаутентифицированный доступ к эндпойнту /v1/sys/metrics.

Параметры profiling:

- unauthenticated_pprof_access — логический параметр. При установке значения true разрешает неаутентифицированный доступ к эндпойнту /v1/sys/pprof.

Параметры inflight_requests_logging:

- unauthenticated_in_flight_requests_access — логический параметр. При установке значения true разрешает неаутентифицированный доступ к эндпойнту /v1/sys/in-flight-req.

Параметры custom_response_headers default:

Список сопоставлений типа:

```
{
 "ключ1" = ["значение1", "значение 2", ...],
 "ключ2" = ["значение1", "значение 2", ...],
}
```

позволяет сопоставить имена заголовков по умолчанию с массивом значений. Заголовки по умолчанию устанавливаются на всех эндпойнтах независимо от значения кода состояния.

Список сопоставлений типа:

```
{
 "ключ1" = ["значение1", "значение 2", ...],
 "ключ2" = ["значение1", "значение 2", ...],
}
```

позволяет сопоставить имена заголовков с массивом значений. Заголовки, указанные в этой секции, устанавливаются для указанных кодов состояния.

Список сопоставлений типа:

```
{
 "ключ1" = ["значение1", "значение 2", ...],
 "ключ2" = ["значение1", "значение 2", ...],
}
```

позволяет сопоставить имена заголовков с массивом значений. Заголовки, указанные в этой секции, устанавливаются для кодов состояния, попадающих в указанные группы кодов состояния.

4.3.6. Примеры конфигурации секции listener tcp

Пример 1. Указание необходимых параметров для TLS.

Здесь демонстрируется указание сертификата и ключа для TLS.

```
listener "tcp" {
 tls_cert_file = "/etc/certs/tls.crt"
 tls_key_file  = "/etc/certs/tls.key"
}
```

Пример 2. Прослушивание на нескольких интерфейсах.

Здесь демонстрируется настройка прослушивания ПО «Deckhouse Stronghold» на частном интерфейсе, а также localhost.

```
listener "tcp" {
  address = "127.0.0.1:8200"
}

listener "tcp" {
  address = "10.0.0.5:8200"
}

# Advertise the non-loopback interface
api_addr = "https://10.0.0.5:8200"
cluster_addr = "https://10.0.0.5:8201"
```

Пример 3. Разрешение неаутентифицированного доступа к метрикам.

```
listener "tcp" {
  telemetry {
    unauthenticated_metrics_access = true
  }
}
```

Пример 4. Разрешение неаутентифицированного доступа к профилированию.

```
listener "tcp" {
  profiling {
    unauthenticated_pprof_access = true
    unauthenticated_in_flight_request_access = true
  }
}
```

Пример 5. Настройка пользовательских HTTP-заголовков ответов.

Администраторы могут настроить подсекцию custom_response_headers в секции listener для добавления пользовательских HTTP-заголовков, соответствующих их приложениям.

```
listener "tcp" {
  custom_response_headers {
    "default" = {
      "Strict-Transport-Security" = ["max-age=31536000","includeSubDomains"],
```

```
    "Content-Security-Policy" = ["connect-src https://clusterA.vault.external/"],
    "X-Custom-Header" = ["Custom Header Default Value"],
   },
   "2xx" = {
    "Content-Security-Policy" = ["connect-src https://clusterB.vault.external/"],
    "X-Custom-Header" = ["Custom Header Value 1", "Custom Header Value 2"],
   },
   "301" = {
    "Strict-Transport-Security" = ["max-age=31536000"],
    "Content-Security-Policy" = ["connect-src https://clusterC.vault.external/"],
   },
  }
}
```

Примеры пользовательских HTTP-заголовков — Strict-Transport-Security и Content-Security-Policy. Их можно настроить для усиления безопасности приложения, которое взаимодействует с эндпойнтами ПО «Deckhouse Stronghold». Сканеры уязвимостей часто исследуют такие связанные с безопасностью HTTP-заголовки. Также можно настроить и специфические для приложения пользовательские заголовки, как настроен X-Custom-Header в примере выше.

Если заголовок определен в нескольких подразделах кодов состояния, будет возвращен заголовок, соответствующий наиболее конкретному коду ответа. Из примера конфигурации ниже ответ 306 вернет значение заголовка Custom для 3xx, а 307 вернет значение заголовка Custom для 307.

```
listener "tcp" {
 custom_response_headers {
  "default" = {
    "X-Custom-Header" = ["default Custom header value"]
  },
  "3xx" = {
    "X-Custom-Header" = ["3xx Custom header value"]
  },
  "307" = {
    "X-Custom-Header" = ["307 Custom header value"]
  }
 }
```

```
}
```

Пример 6. Прослушивание на всех IPv4- и IPv6-интерфейсах.

В этом примере ПО «Deckhouse Stronghold» прослушивает все интерфейсы IPv4 и IPv6, включая localhost.

```
listener "tcp" {
  address       = "[::]:8200"
  cluster_address = "[::]:8201"
}
```

Пример 7. Прослушивание на определенных IPv6-адресах.

В примере настроено использование только IPv6 с привязкой к интерфейсу с IP-адресом 2001:1c04:90d:1c00:a00:27ff:fefa:58ec.

```
listener "tcp" {
  address       = "[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8200"
  cluster_address = "[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8201"
}
```

Объявление внешнего (не loopback-) интерфейса:

```
api_addr = "https://[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8200"
cluster_addr = "https://[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8201"
```

4.3.7. Примеры скрытия информации

Пример 1. Конфигурация с использованием параметров redact_addresses, redact_cluster_name и redact_version для скрытия информации в ответах.

```
ui            = true
cluster_addr  = "https://127.0.0.1:8201"
api_addr      = "https://127.0.0.1:8200"
disable_mlock = true


storage "raft" {
  path = "/path/to/raft/data"
  node_id = "raft_node_1"
}


listener "tcp" {
  address           = "127.0.0.1:8200",
```

```
 tls_cert_file = "/path/to/full-chain.pem"
 tls_key_file  = "/path/to/private-key.pem"
 redact_addresses    = "true"
 redact_cluster_name = "true"
 redact_version      = "true"
}

telemetry {
 statsite_address = "127.0.0.1:8125"
 disable_hostname = true
}
```

Пример 2. Результаты применения параметров скрытия API: /sys/health.

Пример вызова /sys/health/, где скрыты cluster_name и version. Поле cluster_name полностью исключено из ответа, а version представлено пустой строкой ("").

```
$ curl -s https://127.0.0.1:8200/v1/sys/health | jq

{
 "initialized": true,
 "sealed": false,
 "standby": false,
 "performance_standby": false,
 "replication_performance_mode": "disabled",
 "replication_dr_mode": "disabled",
 "server_time_utc": 1715935559,
 "version": "",
 "cluster_id": "be574716-e7e9-a950-ee34-d62d56cd6d4a"
}
```

Пример 3. Результаты применения параметров скрытия API: sys/leader.

Пример вызова /sys/leader/, где скрыты leader_address и leader_cluster_address. Они установлены в пустую строку ("").

```
curl -s https://127.0.0.1:8200/v1/sys/leader | jq

{
 "ha_enabled": true,
 "is_self": true,
```

```
 "active_time": "2024-05-13T07:54:20.471072843Z",
 "leader_address": "",
 "leader_cluster_address": "",
 "performance_standby": false,
 "performance_standby_last_remote_wal": 0,
 "raft_committed_index": 78,
 "raft_applied_index": 78
}
```


Пример 4. Результаты применения параметров скрытия API: sys/seal-status.

Пример вызова /sys/seal-status/, где скрыты поля cluster_name, build_date и version. Поле cluster_name полностью исключено из ответа, а build_date и version представлены пустыми строками ("").


```
curl -s https://127.0.0.1:8200/v1/sys/seal-status | jq


{
 "type": "shamir",
 "initialized": true,
 "sealed": false,
 "t": 3,
 "n": 6,
 "progress": 0,
 "nonce": "",
 "version": "",
 "build_date": "",
 "migration": false,
 "cluster_id": "be574716-e7e9-a950-ee34-d62d56cd6d4a",
 "recovery_seal": false,
 "storage_type": "raft"
}
```


Пример 5. CLI: stronghold status

Команда CLI stronghold status использует эндпойнты, которые поддерживают скрытие данных, поэтому в выводе скрываются Version, Build Date, HA Cluster и Active Node Address.

Version, Build Date и HA Cluster показывают n/a, потому что соответствующий эндпойнт вернул пустую строку. В свою очередь Active Node Address показывается как <none>, потому что адрес был опущен в ответе API.

```
stronghold status


Key                 Value
---                 -----
Seal Type           shamir
Initialized         true
Sealed              false
Total Shares        6
Threshold           3
Version             n/a
Build Date          n/a
Storage Type        raft
HA Enabled          true
HA Cluster          n/a
HA Mode             active
Active Since        2024-05-13T07:54:20.471072843Z
Active Node Address    <none>
Raft Committed Index   78
Raft Applied Index     78
```

4.3.8. Unix

Конфигурация слушателя Unix настраивает ПО «Deckhouse Stronghold» на прослушивание на указанном доменном сокете Unix:

```
listener "unix" {
  address = "/run/vault.sock"
}
```

Можно указать секцию listener больше одного раза, чтобы ПО «Deckhouse Stronghold» прослушивало несколько сокетов.

4.3.8.1. Параметры слушателя Unix

- address — обязательный строковый параметр. Указывает адрес для привязки Unix-сокета.
- socket_mode — необязательный строковый параметр. Изменяет права доступа и специальные флаги Unix-сокета.
- socket_user — необязательный строковый параметр. Изменяет пользователя-владельца Unix-сокета.
- socket_group — необязательный строковый параметр. Изменяет группу-владельца Unix-сокета.

4.3.8.2. Примеры конфигурации секции listener unix

Пример 1. Прослушивание на нескольких сокетах.

В примере ПО «Deckhouse Stronghold» настроено на прослушивание на указанном сокете и сокете по умолчанию.

```
listener "unix" {}


listener "unix" {
 address = "/var/run/vault.sock"
}
```

Пример 2. Прослушивание на нескольких интерфейсах.

В этом примере ПО «Deckhouse Stronghold» настроено на прослушивание на указанном Unix-сокете, а также на loopback-интерфейсе.

```
listener "unix" {
 address = "/var/run/vault.sock"
}


listener "tcp" {
 address = "127.0.0.1:8200"
}
```

Пример 3. Настройка прав.

В примере показана конфигурация прав и владельца — пользователя и группы.

```
listener "unix" {
 address = "/var/run/vault.sock"
 socket_mode = "644"
```

```
 socket_user = "1000"
 socket_group = "1000"
}
```

4.4. Секция storage

Секция storage настраивает бэкенд хранения, который используется для долговременного хранения данных ПО «Deckhouse Stronghold».

У каждого бэкенда есть свои преимущества и недостатки. Так, одни бэкенды предоставляют более надежный процесс резервного копирования и восстановления, а другие поддерживают высокую доступность.

4.4.1. Конфигурация

Бэкенд хранилища настраивается в конфигурационном файле ПО через секцию storage.

```
storage [NAME] {
 [PARAMETERS...]
}
```

Например:

```
storage "file" {
 path = "/mnt/vault/data"
}
```

Для параметров конфигурации, которые также считывают переменную окружения, приоритет будет отдан переменной окружения, а не значениям в файле конфигурации.

4.4.2. Бэкенд хранения filesystem

Бэкенд хранения типа filesystem сохраняет данные ПО «Deckhouse Stronghold» в файловой системе, используя стандартную структуру каталогов. Его можно использовать в ситуациях с одним долговечным сервером или для локальной разработки, где долговечность не является критической.

Бэкенд хранения типа filesystem не поддерживает высокую доступность.

Пример конфигурации секции storage с таким бэкендом хранения:

```
storage "file" {
 path = "/mnt/vault/data"
}
```

4.4.2.1. Параметры

- path — обязательный строковый параметр. Абсолютный путь к каталогу, где будут храниться данные. Если каталога не существует, ПО его создаст.

4.4.3. Бэкенд хранения Raft

Бэкенд интегрированного хранилища (Raft) используется для хранения данных ПО «Deckhouse Stronghold». В отличие от других бэкендов хранения, он не работает с одним источником данных: вместо этого все узлы кластера ПО будут иметь реплицированную копию данных. Данные реплицируются между всеми узлами с помощью алгоритма консенсуса Raft.

Бэкенд хранения Raft поддерживает высокую доступность.

Пример конфигурации секции storage с бэкендом хранения типа Raft:

```
storage "raft" {
  path = "/path/to/raft/data"
  node_id = "raft_node_1"
}
cluster_addr = "http://127.0.0.1:8201"
```

При использовании бэкенда интегрированного хранилища нужно указать параметр cluster_addr, то есть адрес и порт, которые будут использоваться для связи между узлами в кластере Raft.

Также при использовании бэкенда интегрированного хранилища нельзя объявлять отдельный бэкенд ha_storage. Настоятельно рекомендуется установить disable_mlock в значение true и отключить своппинг в системе.

4.4.3.1. Описание параметров

- path — строковый параметр. Указывает путь к каталогу в файловой системе, где хранятся данные ПО «Deckhouse Stronghold». Значение может быть переопределено установкой переменной окружения VAULT_RAFT_PATH.
- node_id — строковый параметр. Идентификатор узла в кластере Raft. Значение может быть переопределено установкой переменной окружения VAULT_RAFT_NODE_ID.
- performance_multiplier — целочисленный параметр. Множитель, который используют серверы для масштабирования ключевых параметров времени Raft.

Настройка параметра performance_multiplier влияет на время, необходимое ПО «Deckhouse Stronghold» для обнаружения сбоев лидера и выбора лидера. Это достигается за счет требования большего количества сетевых и вычислительных ресурсов для улучшения производительности. Если значение не задано или установлено в 0, будет использоваться

стандартное время, описанное ниже. Более низкие значения используются для ужесточения времени и повышения чувствительности, а более высокие ослабляют время и снижают чувствительность.

По умолчанию ПО «Deckhouse Stronghold» использует временные параметры с более низкой производительностью, которые подходят для серверов ПО «Deckhouse Stronghold», соответствующих минимальным требованиям. Это эквивалентно установке значения в 5, но стандарт может измениться в будущих версиях ПО «Deckhouse Stronghold», если поменяется целевой минимальный профиль сервера. Установка значения performance_multiplier в 1 настроит Raft на режим наивысшей производительности и рекомендуется для серверов ПО «Deckhouse Stronghold» в production-среде. Максимально допустимое значение — 10.

- trailing_logs — целочисленный параметр. Контролирует количество записей журнала, которые остаются в хранилище журналов на диске после создания снимка. Этот параметр стоит регулировать только если последователи не могут догнать лидера из-за очень большого размера снимка и высокой пропускной способности записи, вызывающей усечение журнала до того, как снимок может быть полностью установлен.

При необходимости восстановления кластера возможно снижение пропускной способности записи или объема данных, которые хранятся в ПО «Deckhouse Stronghold». Значение по умолчанию — 10000, и оно подходит для всех нормальных рабочих нагрузок. Метрика trailing_logs не тождественна параметру max_trailing_logs.

- snapshot_threshold — целочисленный параметр. Контролирует минимальное количество записей фиксации Raft между снимками, которые сохраняются на диск. Обычно этот низкоуровневый параметр не требует изменений. В сильно загруженных кластерах, которые испытывают чрезмерный дисковый ввод-вывод, можно увеличить значение, чтобы снизить нагрузку на диск и минимизировать вероятность одновременного создания снимков на всех серверах. Увеличение параметра snapshot_threshold — это компромисс между дисковым вводом-выводом и дисковым пространством, так как журнал Raft будет значительно расти, и пространство в файле в raft.db нельзя будет очистить до следующего снимка. Кроме того, при восстановлении после сбоев или переключении на резервный узел серверам понадобится больше времени, ведь потребуется воспроизвести больше журналов. Значение по умолчанию — 8192;
- snapshot_interval — целочисленный параметр. Интервал между снимками в секундах, который контролирует, как часто Raft проверяет необходимость сделать снимок. Чтобы предотвратить ситуацию, когда весь кластер создает снимок одновременно, Raft

случайным образом выбирает время между указанным интервалом и его удвоенным значением. По умолчанию 120 секунд;

- retry_join — список параметров. Позволяет настроить набор параметров подключения для других узлов кластера. Этот набор используется для помощи узлам в поиске лидера для присоединения к кластеру. Можно задать несколько секций retry_join. Если параметры подключения для всех узлов в кластере известны заранее, можно включить эти секции. В таком случае, как только один из узлов будет инициализирован как лидер, оставшиеся будут использовать свою конфигурацию retry_join для поиска лидера и присоединения к кластеру. При использовании схемы разделения секрета Шамира присоединенные узлы нужно будет разблокировать вручную;
- retry_join_as_non_voter — логический параметр. Если установить значение true, то любая конфигурация retry_join присоединяется к кластеру Raft как не участвующая в голосовании. Узел не будет участвовать в кворуме Raft, но будет получать поток репликации данных. Это позволяет добавить кластеру масштабируемость чтения; Опция имеет тот же эффект, что и флаг -non-voter для команды stronghold operator raft join, но влияет только на статус голосования при присоединении через конфигурацию retry_join. Параметр можно переопределить на true, установив переменную окружения VAULT_RAFT_RETRY_JOIN_AS_NON_VOTER в непустое значение. Действует только при наличии хотя бы одной секции retry_join;
- max_entry_size — целочисленный параметр. Настройка максимального количество байт для записи Raft. Применяется как к операциям put, так и к транзакциям. Любая операция put или транзакция, превышающая это значение конфигурации, приведет к сбою соответствующей операции. У Raft есть рекомендуемый максимальный размер данных в записи журнала. Он основан на текущей архитектуре, стандартном времени и т. д. Интегрированное хранилище также использует размер блока — порог, применяемый для разделения большого значения на блоки. По умолчанию размер блока равен максимальному размеру записи журнала Raft. Значение по умолчанию 1048576;
- autopilot_reconcile_interval — строковый параметр. Задает временной интервал, после которого автопилот обнаружит любые изменения состояния. Изменение состояния может означать множество вещей, например: 1. узел, изначально добавленный как не голосующий в кластер Raft, успешно завершил период стабилизации, что квалифицирует его для продвижения в голосующие; 2. узел должен получить статус unhealthy в API состояния; 3. узел был помечен как dead и должен быть исключен из

конфигурации Raft. Значение указывается с использованием суффикса времени, например, "40s" (40 секунд) или "1h" (1 час);

- autopilot_update_interval — строковый параметр. Временной интервал, после которого автопилот будет опрашивать ПО на предмет обновления интересующей информации. Включает такие данные, как конфигурация автопилота и его текущее состояние, конфигурация Raft, известные серверы, последний индекс Raft и статистика для всех известных серверов. Полученная информация будет использоваться для расчета следующего состояния автопилота. Значение указывается с использованием суффикса времени, например, "40s" (40 секунд) или "1h" (1 час).

4.4.3.2. Параметры секции retry_join

- leader_api_addr — строковый параметр. IP-адрес возможного узла-лидера;
- leader_tls_servername — строковый параметр. Имя сервера TLS, которое используется при подключении по HTTPS. Должно соответствовать одному из имен в DNS SANs (Subject Alternative Names) TLS-сертификата лидера. Узел использует leader_tls_servername для верификации сертификата лидера, когда пытается соединиться с лидером кластера. Это помогает обеспечить безопасность соединения и убедиться, что соединение происходит с правильным сервером;
- leader_ca_cert_file — строковый параметр. Путь к файлу с сертификатом ЦС возможного узла-лидера;
- leader_client_cert_file — строковый параметр. Путь к файлу сертификата клиента, который используется для аутентификации при соединении с лидером кластера по протоколу TLS. Узел Raft предъявляет этот сертификат при установлении защищенного соединения с лидером, чтобы подтвердить свою подлинность;
- leader_client_key_file — строковый параметр. Путь к файлу приватного ключа, который используется вместе с клиентским сертификатом для аутентификации при соединении с лидером кластера по протоколу TLS;
- leader_ca_cert — строковый параметр. Значение сертификата ЦС возможного узла-лидера;
- leader_client_cert — строковый параметр. Значение сертификата клиента, который используется для аутентификации при соединении с лидером кластера по протоколу TLS. Узел Raft предъявляет этот сертификат при установлении защищенного соединения с лидером, чтобы подтвердить свою подлинность;

- leader_client_key — строковый параметр. Значение приватного ключа, который используется вместе с клиентским сертификатом для аутентификации при соединении с лидером кластера по протоколу TLS.

Каждый блок retry_join может предоставлять TLS-сертификаты через пути к файлам или как значение сертификата, но не комбинацию обоих. Если передается значение сертификата, оно должно быть указано в одной строке с использованием \n для указания необходимых переносов на новую строку.

4.5. Секция UI

У ПО «Deckhouse Stronghold» есть пользовательский интерфейс (веб-интерфейс, UI) для администраторов. С его помощью можно создавать, читать, обновлять и удалять секреты, проходить аутентификацию, распечатывать хранилище и выполнять другие операции.

4.5.1. Активация UI

По умолчанию пользовательский интерфейс ПО «Deckhouse Stronghold» не активирован. Для его активации установите параметр ui в конфигурации сервера ПО в значение true.

```
ui = true


listener "tcp" {
 # ...
}
```

4.5.2. Доступ к Stronghold UI

Пользовательский интерфейс работает на том же порту, что и слушатель ПО «Deckhouse Stronghold». Для доступа к UI необходимо настроить хотя бы одну секцию listener.

Пользовательский интерфейс доступен по URL-адресу указанному в секции активации с любой машины в подсети — при условии отсутствия межсетевых экранов или соответствующей их настройке. Также UI доступен по любой записи DNS, которая разрешается на этот IP-адрес.

4.6. Секция user_lockout

Секция user_lockout содержит настройки блокировки пользователей при неудачных попытках входа в ПО «Deckhouse Stronghold». Настройки можно применить как глобально — для всех методов аутентификации (userpass, ldap и approle) с использованием общего имени секции

user_lockout "all", так и индивидуально для конкретного метода, указав его имя в секции. Поддерживаются следующие значения: all, userpass, ldap и approle.

Конфигурация конкретного метода аутентификации имеет более высокий приоритет по сравнению с настройками для всех методов аутентификации с использованием общего имени секции user_lockout "all". Если заданы оба варианта конфигурации, будут применены параметры, которые относятся к конкретному методу.

4.6.1. Параметры секции user_lockout

- lockout_threshold — строковый параметр. Указывает количество неудачных попыток входа в систему, после которых пользователь будет заблокирован.
- lockout_duration — строковый параметр. Указывает длительность блокировки пользователя. Значение указывается с использованием суффикса времени, например, "40s" (40 секунд) или "1h" (1 час).
- lockout_counter_reset — строковый параметр. Определяет временной интервал, после которого счетчик блокировки сбрасывается, если нет неудачных попыток входа в систему. Значение указывается с использованием суффикса времени, например, "40s" (40 секунд) или "1h" (1 час).
- disable_lockout — логический параметр. Отключает функцию блокировки пользователя, если установить значение true.

4.7. Настройка политик безопасности

Политики реализуют декларативный способ предоставления или запрета доступа к определенным путям и операциям в ПО «Deckhouse Stronghold». Изначально в ПО отсутствуют политики доступа (доступ к объектам для пользователя запрещен), поэтому пустая политика не предоставляет никаких разрешений в системе.

Политики составляются в формате HCL или JSON и описывают, к каким путям в ПО «Deckhouse Stronghold» пользователю или машине разрешен доступ.

Пример базовой политики, предоставляющей возможность чтения пути KVv1 "secret/foo":

```
path "secret/foo" {
 capabilities = ["read"]
}
```

Когда эта политика назначена токену, токен может читать из "secret/foo". Однако токен не может обновить или удалить "secret/foo", поскольку такие возможности не предоставлены. Поскольку политики по умолчанию запрещены, токен не будет иметь другого доступа в ПО.

Подробное описание политик представлено в (каталог «Электронные приложения» - «Deckhouse Stronghold. Руководство администратора» - «Руководство.pdf).

5. Описание действий по работе в средстве

5.1. Токены сущностей

Каждый метод аутентификации в ПО имеет собственный набор API-путей. Методы аутентификации могут быть активированы по определенному пути, но для упрощения документация будет предполагать использование путей по умолчанию. Если вы активируете методы аутентификации по другому пути, вам следует скорректировать API-запросы соответственно. Информация об идентификации используется в ПО «Deckhouse Stronghold», но ее также можно экспортировать для для использования другими приложениями. Авторизованный пользователь или приложение может запросить токен, который содержит информацию об идентификации для связанной с ним сущности. Эти токены представляют собой подписанные JWT (JSON Web Token), соответствующие структуре OIDC (OpenID Connect) ID token. Открытые ключи, используемые для аутентификации токенов, публикуются ПО «Deckhouse Stronghold» на неаутентифицированном эндпойнте в соответствии с соглашениями OIDC discovery и JWKS (JSON Web Key Set), которые которые должны быть непосредственно использованы библиотеками JWT/OIDC. Эндпойнт интроспекции также предоставляется ПО для проверки токена.

5.1.1. Роли и ключи

OIDC-совместимые ID-токены генерируются на основе роли, которая позволяет настроить требования к токену с помощью системы шаблонов, указать срок его действия (TTL) и задать «ключ», который будет использоваться для подписи токена. Шаблон роли является необязательным параметром для настройки содержимого токена и описывается в следующем разделе. TTL токена контролирует время действия токена, по истечении которого библиотеки проверки будут считать токен недействительным. Все роли имеют связанный с ними client_id, который будет добавляется к параметру aud токена. Библиотеки JWT/OIDC обычно требуют это значение. Параметр может быть установлен Администратором в выбранное значение, либо, если значение не задано, будет использоваться значение, сгенерированное ПО.

Параметр key роли связывает роль с существующим именованным ключом (несколько ролей могут ссылаться на один и тот же ключ). Невозможно сгенерировать неподписанный ID-токен.

Именованный ключ — это пара открытый/закрытый ключ, сгенерированная ПО. Закрытый ключ используется для подписи идентификационных токенов, а открытый ключ используется клиентами для проверки подписи. Ключи регулярно ротируются, при этом генерируется новая пара ключей, а предыдущий публичный ключ сохраняется в течение ограниченного времени для проверки.

В конфигурации именованного ключа указывается период ротации, время проверки, алгоритм подписи и разрешенные идентификаторы клиентов. Период ротации определяет частоту, с которой генерируется новый ключ подписи и удаляется закрытая часть предыдущего ключа подписи. Verification l - это время, в течение которого открытый ключ сохраняется для проверки после ротации. По умолчанию ключи ротируются каждые 24 часа и остаются доступными для проверки в течение 24 часов после их ротации.

Список разрешенных идентификаторов клиентов ключа ограничивает, какие роли могут ссылаться на этот ключ. Параметр может быть установлен в * , чтобы разрешить все роли. Оценка валидности производится при запросе токена, а не во время конфигурации.

5.1.2. Содержание и шаблоны токенов

Токены идентификации всегда будут содержать, как минимум, требования OIDC:

- iss — URL-адрес эмитента;
- sub — идентификатор организации-заявителя;
- aud — client_id роли;
- iat — время выдачи;
- exp — время истечения срока действия токена.

Администратор может настраивать шаблоны для каждой роли, которые позволяют добавлять дополнительную информацию о сущности, которая может быть добавлена к токену. В ПО «Deckhouse Stronghold» реализована свободная ролевая модель. Каждому пользователю присваивается ролевая политика (роль) на доступ к какому-либо объекту в соответствии с шаблоном. Пользователь может обладать несколькими ролевыми политиками (ролями). Шаблоны структурированы в формате JSON с заменяемыми параметрами. Синтаксис параметров аналогичен ACL Path Templa ng.

Пример:

```
{
"color": {{identity.entity.metadata.color}},
"userinfo": {
"username": {{identity.entity.aliases.usermap_123.metadata.username}},
"groups": {{identity.entity.groups.names}}
},
"nbf": {{time.now}}
}
```

Когда запрашивается токен, результирующий шаблон может быть заполнен

следующим образом:

```
{
"color": "green",
"userinfo": {
"username": "bob",
"groups": ["web", "engr", "default"]
},
"nbf": 1561411915
}
```

Эти параметры объединяются с базовыми утверждениями OIDC в окончательный токен:

```
{
"iss": "https://10.1.1.45:8200/v1/identity/oidc",
"sub": "a2cd63d3-5364-406f-980e-8d71bb0692f5",
"aud": "SxSouteCYPBoaTFy94hFghmekos",
"iat": 1561411915,
"exp": 1561412215,
"color": "green",
"userinfo": {
"username": "bob",
"groups": ["web", "engr", "default"]
},
"nbf": 1561411915
```

Слияние шаблонов происходит следующим образом: ключи шаблонов верхнего уровня становятся ключами токенов верхнего уровня ключи токенов. По этой причине шаблоны не могут содержать ключи верхнего уровня, которые перезаписывают стандартные формулы OIDC.

Параметры шаблона, которые отсутствуют для сущности (например, метаданные или несуществующий аксессуар псевдонима), интерпретируются как пустые строки или объекты — в зависимости от типа данных.

Шаблоны настраиваются на роль и могут быть дополнительно закодированы в Base64.

Полный список параметров шаблона приведен в Таблице 2 :

Таблица 2. Список параметров шаблона

Имя	Описание
identity.entity.id	Идентификатор сущности
identity.entity.name	Имя сущности
identity.entity.groups.ids	Идентификаторы групп, членом которых является сущность
identity.entity.groups.names	Имена групп, в которых состоит сущность
identity.entity.metadata	Метаданные, связанные с сущностью
identity.entity.metadata.<metadata key>	Метаданные, связанные с сущностью для данного ключа
identity.entity.aliases.<mount accessor>.id	ID псевдонима сущности для данного монтирования
identity.entity.aliases.<mount accessor>.name	Имя псевдонима сущности для данного монтирования
identity.entity.aliases.<mount accessor>.metadata	Метаданные, связанные с псевдонимом для данного монтирования
identity.entity.aliases.<mount accessor>.metadata.<metadata key>	Метаданные, связанные с псевдонимом для данного монтирования и ключом метаданных
identity.entity.aliases.<mount accessor>.custom_metadata	Пользовательские метаданные, связанные с псевдонимом для данного монтирования
identity.entity.aliases.<mount accessor>.custom_metadata.<custom metadata key>	Пользовательские метаданные, связанные с псевдонимом для данного монтирования и ключом пользовательских метаданных
time.now	Текущее время в интегральных секундах с начала «эпохи Unix» (1 января 1970 года)
time.now.plus.<duration>	Текущее время плюс длительность
time.now.minus.<duration>	Текущее время минус длительность

5.1.3. Генерация токенов

Аутентифицированный клиент может запросить токен, используя генерацию токена эндпойнта. Токен будет сгенерирован в соответствии со спецификациями запрашиваемой роли для запрашивающей сущности. Невозможно сгенерировать токен для произвольной сущности.

5.1.4. Проверка подлинности идентификационных токенов, сгенерированных ПО «Deckhouse Stronghold»

Идентификационный токен может быть проверен клиентской стороной с помощью открытых ключей, опубликованных ПО «Deckhouse Stronghold», или через эндпойнт интроспекции, предоставленный ПО.

ПО «Deckhouse Stronghold» будет обслуживать стандартные «.well-known»-эндпойнты, которые позволяют интегрироваться с библиотеками проверки OIDC. Настройка библиотек обычно включает предоставление URL-адреса эмитента и идентификатора клиента. Библиотека будет обрабатывать запросы ключей и проверять подписи и требования к утверждениям на токенах. Преимущество этого подхода заключается в том, что требуется только доступ к ПО «Deckhouse Stronghold», а не авторизация, поскольку «.well-known»-эндпойнты не проходят аутентификацию.

В качестве альтернативы токен может быть отправлен в ПО для проверки через эндпойнт интроспекции. В ответе будет указано, является ли токен «активным» или нет, а также любые ошибки, возникшие при проверке. Помимо того, что клиент может делегировать проверку ПО «Deckhouse Stronghold», использование этого эндпойнта подразумевает дополнительную проверку того, является ли сущность активной или нет, то есть то, что что невозможно определить только по токену. В отличие от «.well-known»-эндпойнта, для доступа к эндпойнту интроспекции требуется действительный токен ПО и достаточный уровень авторизации.

5.1.5. Эмитенты

Система идентификационных токенов имеет один настраиваемый параметр: эмитент. Параметр эмитента iss особенно важен для правильной проверки токена клиентами. Потребители токена будут запрашивать открытые ключи у ПО, используя URL-адрес эмитента, поэтому он должен быть доступен по сети. Более того, возвращаемый набор ключей будет включать URL-адрес эмитента, который должен соответствовать запросу.

По умолчанию ПО «Deckhouse Stronghold» подставляет URL-адрес эмитента в адрес экземпляра ПО api_addr. Это означает, что токены, выпущенные в кластере, должны быть подтверждены в пределах этого кластера. В качестве альтернативы, параметр issuer может быть задан явно. В этом адресе должен быть указан путь к идентификатору/OIDC для экземпляра ПО (например, https://stronghold.example.com:8200/v1/identity/oidc) и должен быть доступным для любого клиента при попытке подтвердить идентификационные токены.

5.2. Методы аутентификации

Каждый метод аутентификации в ПО имеет собственный набор API-путей. Методы аутентификации могут быть активированы по определенному пути, но для упрощения документация будет предполагать использование путей по умолчанию. Если вы активируете методы аутентификации по другому пути, вам следует скорректировать API-запросы соответственно.

Методы аутентификации могут быть включены и отключены с помощью UI, CLI или API.

Включение с помощью CLI:

```
stronghold auth enable approle
```

При включении методы аутентификации работают аналогично механизмам секретов: они монтируются в таблицу монтирования ПО «Deckhouse Stronghold» и становятся доступны для настройки и использования через стандартный API для чтения и записи. Все методы аутентификации по умолчанию монтируются в поддиректории auth/ и отображаются как auth/, например, auth/oidc/.

Администраторы ПО «Deckhouse Stronghold» могут монтировать один и тот же метод аутентификации несколько раз, используя CLI для задания пути, отличного от стандартного. Процесс настройки метода с помощью UI представлен на Рисунках 1–5.

Включение метода AppRole с помощью UI представлено на рисунке 1.

Рисунок 1. Включение метода AppRole с помощью UI

Выбор метода аутентификации представлен на Рисунке 2.

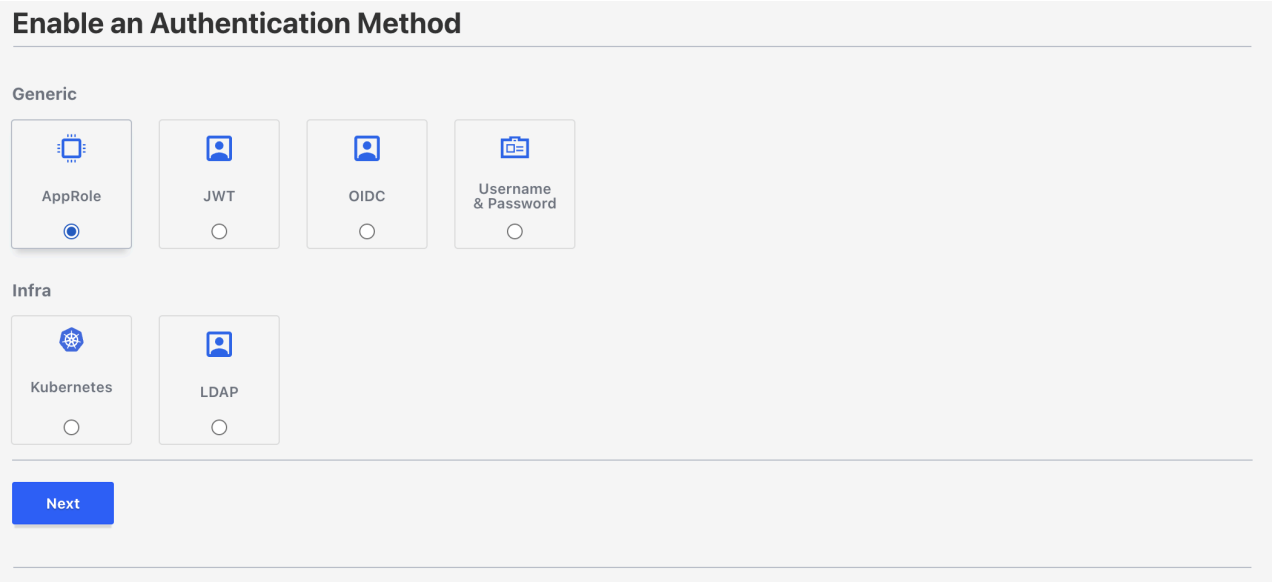

Рисунок 2. Выбор метода аутентификации

Далее необходимо настроить и подтвердить включение метода аутентификации, как показано на Рисунке 3.

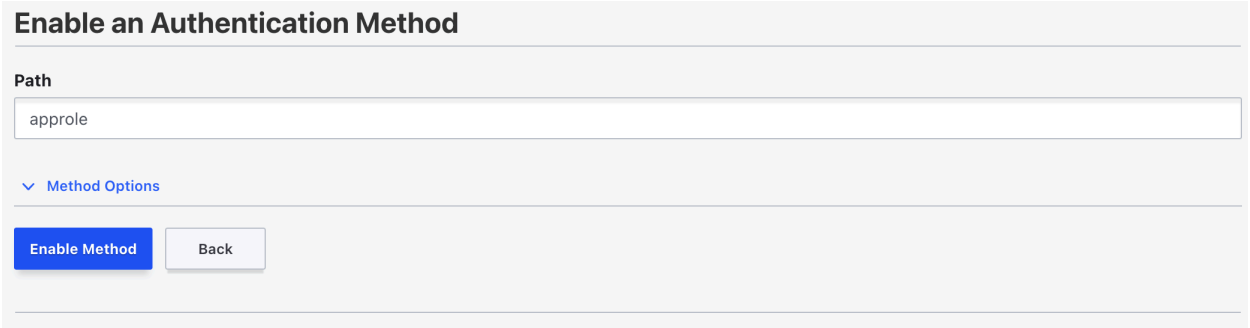

Рисунок 3. Подтверждение включения метода аутентификации

Для отключения метода аутентификации нажмите кнопку с тремя точками в строке с названием метода и выберите пункт Disable, как показано на Рисунке 4.

Рисунок 4. Отключение метода аутентификации

Для удаления метода аутентификации необходимо подтвердить удаление, как показано на Рисунке 5.

Рисунок 5. Подтверждение удаления метода аутентификации

5.2.1. AppRole

В этом разделе рассматривается метод аутентификации по пути /auth/approle.

5.2.1.1. Вывод AppRole

Этот путь возвращает список существующих AppRoles в методе.

Метод	Путь
LIST	/auth/approle/role

Пример запроса:

```
$ curl \
    --header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
    --request POST \
    ${STRONGHOLD_ADDR}/v1/auth/approle/tidy/secret-id
```

Пример ответа API:

```
{
  "auth": null,
  "warnings": null,
  "wrap_info": null,
  "data": {
    "keys": ["dev", "prod", "test"]
  },
  "lease_duration": 0,
  "renewable": false,
  "lease_id": ""
```

```
}
```

5.2.1.2. Изменение AppRole

Создает новую AppRole или обновляет существующую. Этот путь поддерживает как создание, так и обновление роли. К роли можно применить одно или несколько ограничений. Необходимо, чтобы хотя бы одно из них было включено при создании или обновлении роли.

Метод	Путь
POST	/auth/approle/role/:role_name

Описание параметров:

- role_name (строка:) — имя AppRole. Должно быть короче 4096 байт, допустимые символы включают a-Z, 0-9, пробел, тире, подчеркивания и точки;
- bind_secret_id (булевый: true) — требуется, чтобы secret_id был представлен при аутентификации с использованием указанной AppRole;
- secret_id_bound_cidrs (массив: []) — строка, разделенная запятыми, или список блоков CIDR; в значении указываются блоки IP-адресов, которые могут выполнять аутентификацию;
- secret_id_num_uses (целое число: 0) — количество раз, которое SecretID может быть использован для получения токена через данную AppRole. При превышении этого количества SecretID становится недействительным. Значение 0 подразумевает неограниченное количество использований. Параметр может быть переопределен полем 'num_uses' в запросе при создании SecretID;
- secret_id_ttl (строка: "") — срок действия SecretID в виде количества секунд (3600) или временного интервала (60m). Значение 0 означает, что у SecretID не будет срока действия. Параметр может быть переопределен полем 'ttl' в запросе при создании SecretID;
- local_secret_ids (булевый: false) — если установлен в true, создаваемые для данной роли SecretID будут локальными для текущего кластера. Этот параметр можно задать только при создании роли и изменить его впоследствии невозможно;
- token_ttl (целое число) — срок действия созданных токенов. Значение параметра учитывается при продлении, рекомендуется установить TTL равный 86400 (значение указано в секундах);

- token_max_ttl (целое число) — максимальный срок действия созданных токенов. Значение параметра учитывается при продлении. Рекомендуемое значение token_max_ttl 604800 (значение указано в секундах);
- token_policies (массив: [] или строка с разделением запятыми: "") — список политик, которые будут применены к создаваемым токенам. В зависимости от метода аутентификации список может быть дополнен пользовательскими, групповыми или другими политиками;
- token_bound_cidrs (массив: [] или строка с разделением запятыми: "") — список блоков CIDR; в значении указываются блоки IP-адресов, с которых возможна успешная аутентификация. Токен будет привязан к указанным блокам;
- token_explicit_max_ttl (целое число: 0 или строка: "") — задает жесткое ограничение максимального срока действия токена. Это значение имеет приоритет над параметрами token_ttl и token_max_ttl, даже если они допускают продление токена;
- token_no_default_policy (булевый: false) — если установлен в true, политика по умолчанию не будет автоматически применяться к создаваемым токенам; в противном случае она будет добавлена к политикам, установленным в параметре token_policies;
- token_num_uses (целое число: 0) — максимальное количество использований для создаваемого токена (в пределах его срока действия); значение 0 означает неограниченное количество использований. Для создания дочерних токенов установите значение 0;
- token_period (целое число: 0 или строка: "") — максимально допустимый срок действия периодических токенов, запрашиваемых через данную роль;
- token_type (строка: "") — тип создаваемого токена. Допустимые значения: service, batch или default. Default означает использование типа по умолчанию, которым является service, если определено иначе. Для ролей хранилища токенов доступны дополнительные значения: default-service и default-batch, которые указывают тип возвращаемого токена, если клиент явно не запросит другой тип при создании. Для случаев аутентификации, основанной на машинном взаимодействии, используйте токены типа batch.

Пример данных:

```
{
"token_type": "batch",
"token_ttl": "10m",
```

```
"token_max_ttl": "15m",
"token_policies": ["default"],
"period": 0,
"bind_secret_id": true
}
```

Пример запроса:

```
curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request POST \
--data @payload.json \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application
```

Пример ответа API:

```
{
"auth": null,
"warnings": null,
"wrap_info": null,
"data": {
"keys": ["dev", "prod", "test"]
},
"lease_duration": 0,
"renewable": false,
"lease_id": ""
}
```

5.2.1.3. Чтение списка ролей

Выводит параметры существующей AppRole.

Метод	Путь
GET	/auth/approle/role/:role_name

Параметр: role_name (строка: <обязательный_параметр>) — имя AppRole. Должно быть короче 4096 байт.

Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
```

```
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1
```

Пример ответа API:

```
{
"auth": null,
"warnings": null,
"wrap_info": null,
"data": {
"token_ttl": 1200,
"token_max_ttl": 1800,
"secret_id_ttl": 600,
"secret_id_num_uses": 40,
"token_policies": ["default"],
"period": 0,
"bind_secret_id": true,
"secret_id_bound_cidrs": []
},
"lease_duration": 0,
"renewable": false,
"lease_id": ""
}
```

5.2.1.4. Удаление AppRole

Удаляет существующую AppRole из метода.

Метод	Путь
DELETE	/auth/approle/role/:role_name

Параметр: role_name (строка: <обязательный_параметр>) — имя AppRole. Должно быть короче 4096 байт.

Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request DELETE \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1
```

5.2.1.5. Чтение RoleID AppRole

Выводит RoleID существующей AppRole.

Метод	Путь
GET	/auth/approle/role/:role_name/role-id

Параметр: role_name (строка: <обязательный_параметр>) — имя AppRole. Должно быть короче 4096 байт.

Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1/role-id
```

Пример ответа API:

```
{
"auth": null,
"warnings": null,
"wrap_info": null,
"data": {
"role_id": "e5a7b66e-5d08-da9c-7075-71984634b882"
},
"lease_duration": 0,
"renewable": false,
"lease_id": ""
}
```

5.2.1.6. Редактирование RoleID AppRole

Позволяет задать новое значение RoleID для существующей роли AppRole.

Метод	Путь
POST	/auth/approle/role/:role_name/role-id

Параметры:

- role_name (строка: <обязательный_параметр>) — имя AppRole. Должно быть короче 4096 байт.
- role_id (строка: <обязательный_параметр>) — значение RoleID.

Пример данных:

```
{
"role_id": "custom-role-id"
}
```

Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request POST \
--data @payload.json \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1/role-id
```

5.2.1.7. Генерация нового SecretID

Генерирует и выводит новый SecretID для существующей AppRole. Как и в случае с токенами, ответ также будет содержать значение secret_id_accessor, которое можно использовать для чтения параметров секрета без раскрытия самого индентификатора, а также для удаления SecretID из AppRole.

Метод	Путь
POST	/auth/approle/role/:role_name/secret-id

Параметры:

- role_name (строка: <обязательный_параметр>) — имя AppRole. Должно быть короче 4096 байт.
- metadata (строка: "") — метаданные, связанные с SecretID. Должны быть представлены в виде строки в формате JSON, содержащей пары ключ-значение. Метаданные будут применены к токенам, выданным с использованием данного SecretID, и записаны в журнал аудита в открытом виде.
- cidr_list (массив: []) — Строка, разделенная запятыми, или список блоков CIDR, ограничивающих использование SecretID определенным набором IP-адресов. Если для роли задан параметр secret_id_bound_cidrs, указанный в cidr_list список блоков CIDR должен быть его подмножеством.
- token_bound_cidrs (массив: []) — строка, разделенная запятыми, или список блоков CIDR, определяющих IP-адреса, с которых могут использоваться аутентификационные токены, созданные с помощью данного SecretID. Значение переопределяет соответствующий параметр роли, но должно быть его подмножеством.

- num_uses (целое число: 0) — количество допустимых использований данного SecretID, по достижении которого SecretID становится недействительным. Значение 0 означает неограниченное количество использований. Переопределяет параметр secret_id_num_uses, заданный на уровне роли, и не может его превышать.
- ttl (строка: "") — срок действия SecretID в секундах (3600) или в виде временного интервала (60m). Значение 0 означает, что срок действия отсутствует. Переопределяет параметр secret_id_ttl, заданный на уровне роли, и не может превышать его.

Пример данных:

```
{
"metadata": "{ \"tag1\": \"production\" }",
"ttl": 600,
"num_uses": 50
}
```

Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request POST \
--data @payload.json \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1/secret-id
```

Пример ответа API:

```
{
"auth": null,
"warnings": null,
"wrap_info": null,
"data": {
"secret_id_accessor": "84896a0c-1347-aa80-a3f6-aca8b7558780",
"secret_id": "841771dc-11c9-bbc7-bcac-6a3945a69cd9",
"secret_id_ttl": 600,
"secret_id_num_uses": 50
},
"lease_duration": 0,
"renewable": false,
```

```
"lease_id": ""
}
```

5.2.1.8. Список идентификаторов доступа SecretID

Выводит идентификаторы доступа всех выданных SecretID для AppRole, включая идентификаторы доступа для "пользовательских" SecretID.

Метод	Путь
LIST	/auth/approle/role/:role_name/secret-id

Параметр: role_name (строка: <обязательный_параметр>) — имя AppRole. Должно быть короче 4096 байт.

Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request LIST \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1/secret-id
```

Пример ответа API:

```
{
"auth": null,
"warnings": null,
"wrap_info": null,
"data": {
"keys": [
"ce202d2a-8253-c437-bf9a-aceed4241491",
"a1c0dee4-b869-e68d-3520-2040c1a0849a",
"be03b7e2-044c-7244-07e1-47560ca1c787",
"84896a0c-1347-aa80-a3f6-aca8b7558780",
"439b1328-6523-15e7-403a-a48038cdc45a"
]
},
"lease_duration": 0,
"renewable": false,
"lease_id": ""
}
```

5.2.1.9. Чтение SecretID AppRole

Выводит параметры SecretID AppRole.

Метод	Путь
POST	/auth/approle/role/:role_name/secret-id/lookup

Параметры:

- role_name (строка: <обязательный_параметр>) — имя AppRole. Должно быть короче 4096 байт.
- secret_id (строка: <обязательный_параметр>) — SecretID, привязанный к роли.

Пример данных:

```
{
"secret_id": "84896a0c-1347-aa80-a3f6-aca8b7558780"
}
```

Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request POST \
--data @payload.json \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1/secret-id/lookup
```

Пример ответа API:

```
{
"request_id": "74752925-f309-6859-3d2d-0fcded95150e",
"lease_id": "",
"renewable": false,
"lease_duration": 0,
"data": {
"cidr_list": [],
"creation_time": "2023-02-10T18:17:27.089757383Z",
"expiration_time": "0001-01-01T00:00:00Z",
"last_updated_time": "2023-02-10T18:17:27.089757383Z",
"metadata": {
```

```
"tag1": "production"
},
"secret_id_accessor": "2be760a4-87bb-2fa9-1637-1b7fa9ba2896",
"secret_id_num_uses": 0,
"secret_id_ttl": 0,
"token_bound_cidrs": []
},
"wrap_info": null,
"warnings": null,
"auth": null
}
```


5.2.1.10. Удаление SecretID AppRole

Удаляет SecretID AppRole.

Метод	Путь
POST	/auth/approle/role/:role_name/secret-id/destroy

Параметры:

- role_name (строка: <обязательный_параметр>) — имя AppRole. Должно быть короче 4096 байт.
- secret_id (строка: <обязательный_параметр>) — SecretID, привязанный к роли.

Пример данных:


```
{
"secret_id": "84896a0c-1347-aa80-a3f6-aca8b7558780"
}
```


Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request POST \
--data @payload.json \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1/secret-id/destroy
```

5.2.1.11. Чтение SecretID AppRole

Выводит свойства SecretID AppRole.

Метод	Путь
POST	/auth/approle/role/:role_name/secret-id-accessor/lookup

Параметры:

- role_name (строка: <обязательный_параметр>) - Имя AppRole. Должно быть короче 4096 байт.
- secret_id_accessor (строка: <обязательный_параметр>) — привязанный к роли идентификатор доступа, позволяющий ссылаться на SecretID без раскрытия его значения.

Пример данных:

```
{
"secret_id_accessor": "84896a0c-1347-aa80-a3f6-aca8b7558780"
}
```

Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request POST \
--data @payload.json \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1/secret-id-accessor/lookup
```

Пример ответа API:

```
{
"request_id": "72836cd1-139c-fe66-1402-8bb5ca4044b8",
"lease_id": "",
"renewable": false,
"lease_duration": 0,
"data": {
"cidr_list": [],
"creation_time": "2023-02-10T18:17:27.089757383Z",
"expiration_time": "0001-01-01T00:00:00Z",
```

```
"last_updated_time": "2023-02-10T18:17:27.089757383Z",
"metadata": {
"tag1": "production"
},
"secret_id_accessor": "2be760a4-87bb-2fa9-1637-1b7fa9ba2896",
"secret_id_num_uses": 0,
"secret_id_ttl": 0,
"token_bound_cidrs": []
},
"wrap_info": null,
"warnings": null,
"auth": null
}
```


5.2.1.12. Уничтожение SecretID AppRole по идентификатору доступа

Уничтожает SecretID AppRole по его идентификатору доступа (secret_id_accessor).

Метод	Путь
POST	/auth/approle/role/:role_name/secret-id-accessor/destroy

Параметры:

- role_name (строка: <обязательный_параметр>) — имя AppRole. Должно быть короче 4096 байт.
- secret_id_accessor (строка: <обязательный_параметр>) — привязанный к роли идентификатор доступа, позволяющий ссылаться на SecretID без раскрытия его значения.

Пример данных:


```
{
"secret_id_accessor": "84896a0c-1347-aa80-a3f6-aca8b7558780"
}
```


Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request POST \
```

```
--data @payload.json \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1/secret-id-accessor/destroy
```

5.2.1.13. Создание пользовательского SecretID AppRole

Назначает "пользовательский" SecretID для существующей роли AppRole. Используется в модели Push-операции.

Метод	Путь
POST	/auth/approle/role/:role_name/custom-secret-id

Параметры:

- role_name (строка: <обязательный_параметр>) — имя AppRole. Должно быть короче 4096 байт.
- secret_id (строка: <обязательный_параметр>) — SecretID, который будет привязан к роли.
- metadata (строка: "") — метаданные, связанные с SecretID. Должны быть представлены в виде строки в формате JSON, содержащей метаданные в виде пары ключ-значение. Метаданные будут применены к токенам, выданным с использованием данного SecretID, и будут записаны в журнал аудита в открытом виде.
- cidr_list (массив: []) — строка, разделенная запятыми, или список блоков CIDR, ограничивающих использование SecretID определенным набором IP-адресов. Если для роли задан параметр secret_id_bound_cidrs, указанный в cidr_list список блоков CIDR должен быть его подмножеством.
- token_bound_cidrs (массив: []) — строка, разделенная запятыми, или список блоков CIDR, определяющих IP-адреса, с которых могут использоваться аутентификационные токены, созданные с помощью данного SecretID. Значение переопределяет соответствующий параметр роли, но должно быть его подмножеством.
- num_uses (целое число: 0) — количество допустимых использований данного SecretID, по достижении которого SecretID становится недействительным. Значение 0 означает неограниченное количество использований. Переопределяет параметр secret_id_num_uses, заданный на уровне роли, и не может его превышать.
- ttl (строка: "") — срок действия SecretID в секундах (3600) или в виде временного интервала (60m). Значение 0 означает, что срок действия отсутствует. Переопределяет параметр secret_id_ttl, заданный на уровне роли, и не может его превышать.

Пример данных:

```
{
"secret_id": "testsecretid",
"ttl": 600,
"num_uses": 50
}
```

Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request POST \
--data @payload.json \
${STRONGHOLD_ADDR}/v1/auth/approle/role/application1/custom-secret-id
```

Пример ответа API:

```
{
"auth": null,
"warnings": null,
"wrap_info": null,
"data": {
"secret_id": "testsecretid",
"secret_id_accessor": "84896a0c-1347-aa80-a3f6-aca8b7558780",
"secret_id_ttl": 600,
"secret_id_num_uses": 50
},
"lease_duration": 0,
"renewable": false,
"lease_id": ""
}
```

5.2.1.14. Вход с использованием AppRole

Выдает токен ПО «Deckhouse Stronghold» на основе представленных учетных данных. role_id требуется всегда; если bind_secret_id включен (по умолчанию) в AppRole, также требуется

secret_id. Также оцениваются любые другие привязанные значения аутентификации в AppRole (например, CIDR клиентского IP).

Метод	Путь
POST	/auth/approle/login

Параметры:

- role_id (строка: <обязательный_параметр>) — RoleID AppRole.
- secret_id (строка: <обязательный_параметр>) — SecretID, привязанный к AppRole.

Пример данных:

```
{
"role_id": "59d6d1ca-47bb-4e7e-a40b-8be3bc5a0ba8",
"secret_id": "84896a0c-1347-aa80-a3f6-aca8b7558780"
}
```

Пример запроса:

```
$ curl \
--request POST \
--data @payload.json \
${STRONGHOLD_ADDR}/v1/auth/approle/login
```

Пример ответа API:

```
{
"auth": {
"renewable": true,
"lease_duration": 1200,
"metadata": null,
"token_policies": ["default"],
"accessor": "fd6c9a00-d2dc-3b11-0be5-af7ae0e1d374",
"client_token": "5b1a0318-679c-9c45-e5c6-d1b9a9035d49"
},
"warnings": null,
"wrap_info": null,
"data": null,
"lease_duration": 0,
"renewable": false,
"lease_id": ""
```

```
}
```

5.2.1.15. Чтение, обновление или удаление параметров AppRole

Вносит изменения в соответствующий параметр существующей AppRole. Параметры AppRole можно изменить, используя прямой доступ к пути /auth/approle/role/:role_name. Пути для каждого поля предоставляются отдельно, чтобы иметь возможность делегировать конкретные пути с использованием системы ACL Stronghold.

Метод	Путь
GET/POST/DELETE	/auth/approle/role/:role_name/policies
GET/POST/DELETE	/auth/approle/role/:role_name/secret-id-num-uses
GET/POST/DELETE	/auth/approle/role/:role_name/secret-id-ttl
GET/POST/DELETE	/auth/approle/role/:role_name/token-ttl
GET/POST/DELETE	/auth/approle/role/:role_name/token-max-ttl
GET/POST/DELETE	/auth/approle/role/:role_name/bind-secret-id
GET/POST/DELETE	/auth/approle/role/:role_name/secret-id-bound-cidrs
GET/POST/DELETE	/auth/approle/role/:role_name/token-bound-cidrs
GET/POST/DELETE	/auth/approle/role/:role_name/period

Ссылка на путь /auth/approle/role/:role_name.

5.2.1.16. Очистка токенов

Выполняет очистку недействительных записей, которые могут оставаться в хранилище токенов. Обычно эта операция не требуется, если только она не указана явно в примечаниях к релизу или рекомендациях службы поддержки. Поскольку операция может привести к большому количеству операций ввода-вывода с хранилищем, ее следует использовать с осторожностью.

Метод	Путь
POST	/auth/approle/tidy/secret-id

Пример запроса:

```
$ curl \
--header "X-Vault-Token: ${STRONGHOLD_TOKEN}" \
--request POST \
${STRONGHOLD_ADDR}/v1/auth/approle/tidy/secret-id
```

Пример ответа API:

```
{
"request_id": "b20b56e3-4699-5b19-cc6b-e74f7b787bbf",
"lease_id": "",
"renewable": false,
"lease_duration": 0,
"data": null,
"wrap_info": null,
"warnings": [
"Tidy operation successfully started. Any information from the operation will be prin
],
"auth": null
}
```

5.3. Резервное копирование

В ПО «Deckhouse Stronghold» реализована процедура резервного копирования. При резервном копировании создается полная копия базы данных, с которой работает ПО «Deckhouse Stronghold». Ниже представлены шаги для настройки и работы с функцией резервного копирования.

5.3.1. Создание/обновление конфигурации автоматического резервного копирования

Метод	Путь
POST	/sys/storage/raft/snapshot-auto/config/:name

5.3.2. Описание параметров резервного копирования

- interval (целое число или строка:) — время между снимками (снэпшотами). Может быть указано в виде целого количества секунд, либо как строка формата Go duration (например, 24h);

- retain (целое число: 3) — количество хранимых снимков. При превышении допустимого количества старые снимки будут удаляться;
- path_prefix (неизменяемая строка:) — для storage_type=local указывает каталог, в который сохраняются снимки. Для облачных типов хранилищ задает префикс бакета, который следует использовать. Начальный символ `/` игнорируется; последующие символы `/` допустимы, но не обязательны. После создания не поддается изменению;
- file_prefix (неизменяемая строка: “stronghold-snapshot”) — строка, с которой начинается имя каждого файла или объекта снимка в пределах каталога или префикса бакета, заданного параметром `path_prefix`. Не поддается изменению;
- storage_type (неизменяемая строка:) — тип используемого хранилища. Поддерживаются значения local и aws-s3. Описанные ниже параметры зависят от назначенного типа хранилища и имеют соответствующий префикс. Не поддается изменению;
- local_max_space (целое число: 0) — для storage_type=local указывает максимальный объем дискового пространства (в байтах), доступный для хранения всех снимков с заданным file_prefix в каталоге path_prefix. При недостатке доступного дискового пространства операции по созданию снимков будут завершаться с ошибкой. Значение 0 (по умолчанию) отключает проверку занимаемого дискового пространства;
- aws_s3_bucket (строка:) — имя S3-бакета для хранения снимков;
- aws_s3_region (строка) — регион, в котором расположен S3-бакета;
- aws_access_key_id (строка) — идентификатор ключа для доступа к S3-бакету;
- aws_secret_access_key (строка) — секретный ключ для доступа к S3-бакету;
- aws_s3_endpoint (строка) — эндпойнт для подключения к S3-бакету;
- aws_s3_disable_tls (булевый) — отключает использование TLS при подключении к S3-эндпойнту. Рекомендуется использовать этот параметр только для тестирования, в сочетании с aws_s3_endpoint;
- aws_s3_ca_certificate (строка) — сертификат центра сертификации (CA) для верификации эндпойнта. Должен быть представлен в формате PEM.

5.3.3. Создание резервной копии

При создании резервной копии необходимо указывать все обязательные поля.

Пример запроса:

```
stronghold write sys/storage/raft/snapshot-auto/config/s3every5min - <<EOF
```

```
{
   "interval":        "5m",
   "path_prefix":      "backups",
   "file_prefix":      "main_stronghold",
   "retain":           "4",
   "storage_type":     "aws-s3",
   "aws_s3_bucket":        "my_bucket",
   "aws_s3_endpoint":      "minio.domain.ru",
   "aws_access_key_id":    "oWdPcQ50zTuMjJI",
   "aws_secret_access_key": "4NzZjboafWyfNTe7aUVgLUdrMurHjty43iUXHFBw"
}
EOF
```

Пример ответа API:

```
Key    Value
---    -----
msg    successfully created config
```

5.3.4. Обновление резервной копии

При обновлении резервной копии допускается частичное указание полей параметров. Значения не измененных полей при сохранении не меняются.

Пример запроса:

```
 stronghold write sys/storage/raft/snapshot-auto/config/s3every5min - <<EOF
{
   "interval":        "3m",
   "retain":           "10",
   "aws_access_key_id":    "vnR9Rfp0toPPgK3",
   "aws_secret_access_key": "FuloGN1RZCtwINCLJtwHXTQ50zCL7s"
}
EOF
```

Пример ответа API:

```
Key    Value
---    -----
msg    successfully updated config
```

5.3.5. Вывод списка существующих конфигураций автоматического резервного копирования

Используется для получения списка названий всех существующих автоматических снэпшотов.

Метод	Путь
LIST	/sys/storage/raft/snapshot-auto/config

Пример запроса:

```
stronghold list sys/storage/raft/snapshot-auto/config
```

Пример ответа API:

```
Keys
----
s3every5min
localEvery3min
```

5.3.6. Получение параметров конфигурации автоматического резервного копирования

Метод	Путь
GET	/sys/storage/raft/snapshot-auto/config/:name

Пример запроса:

```
stronghold read sys/storage/raft/snapshot-auto/config/s3every5min
```

Пример ответа API:

```
Key                 Value
---                 -----
interval            300
path_prefix         backups
file_prefix         main_stronghold
retain              4
storage_type        aws-s3
aws_s3_bucket       my_bucket
aws_s3_disable_tls  false
aws_s3_endpoint     minio.domain.ru
```

```
aws_s3_region          n/a
aws_s3_ca_certificate  n/a
```

5.3.7. Удаление конфигурации автоматического резервного копирования

Метод	Путь
DELETE	/sys/storage/raft/snapshot-auto/config/:name

Пример запроса:

```
stronghold delete sys/storage/raft/snapshot-auto/config/s3every5min
```

Пример ответа API:

```
Key                 Value
---                 -----
consecutive_errors  0
last_snapshot_end   2025-01-31T15:24:14Z
last_snapshot_error n/a
last_snapshot_start 2025-01-31T15:24:12Z
last_snapshot_url   https://minio.domain.ru/my_bucket/backups/main_stronghold_2025-01-31T15:24:12Z
next_snapshot_start 2025-01-31T15:29:12Z
snapshot_start      2025-01-31T15:24:12Z
snapshot_url        https://minio.domain.ru/my_bucket/backups/main_stronghold_2025-01-31T15:24:12Z
```

5.3.8. Получение статуса работы автоматического резервного копирования

Метод	Путь
GET	/sys/storage/raft/snapshot-auto/status/:name

Пример запроса:

```
stronghold read sys/storage/raft/snapshot-auto/status/s3every5min
```

Пример ответа API:

```
Key  Value
---  -----
msg  successfully deleted config
```

5.3.9. Просмотр расписания резервного копирования

Для просмотра информации об автоматическом расписании резервного копирования следует нажать соответствующую кнопку в меню слева.

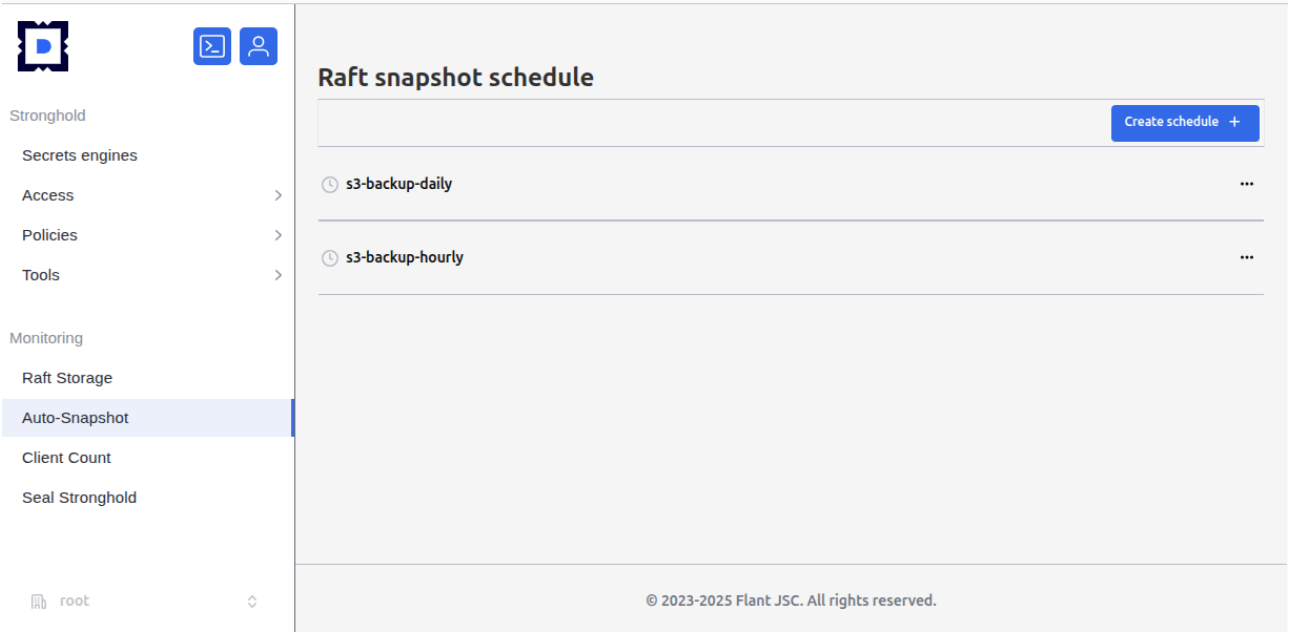

Рисунок 6 - Просмотр расписания резервного копирования

5.3.10. Создание расписания резервного копирования

Для создания нового расписания, необходимо нажать на кнопку, расположенную в правом верхнем углу (Рисунок 6). После этого станет доступно окно создания нового расписания (Рисунок 7), где необходимо указать наименование расписания, интервал создания резервной копии (в секундах), максимальное количество сохраняемых копий.

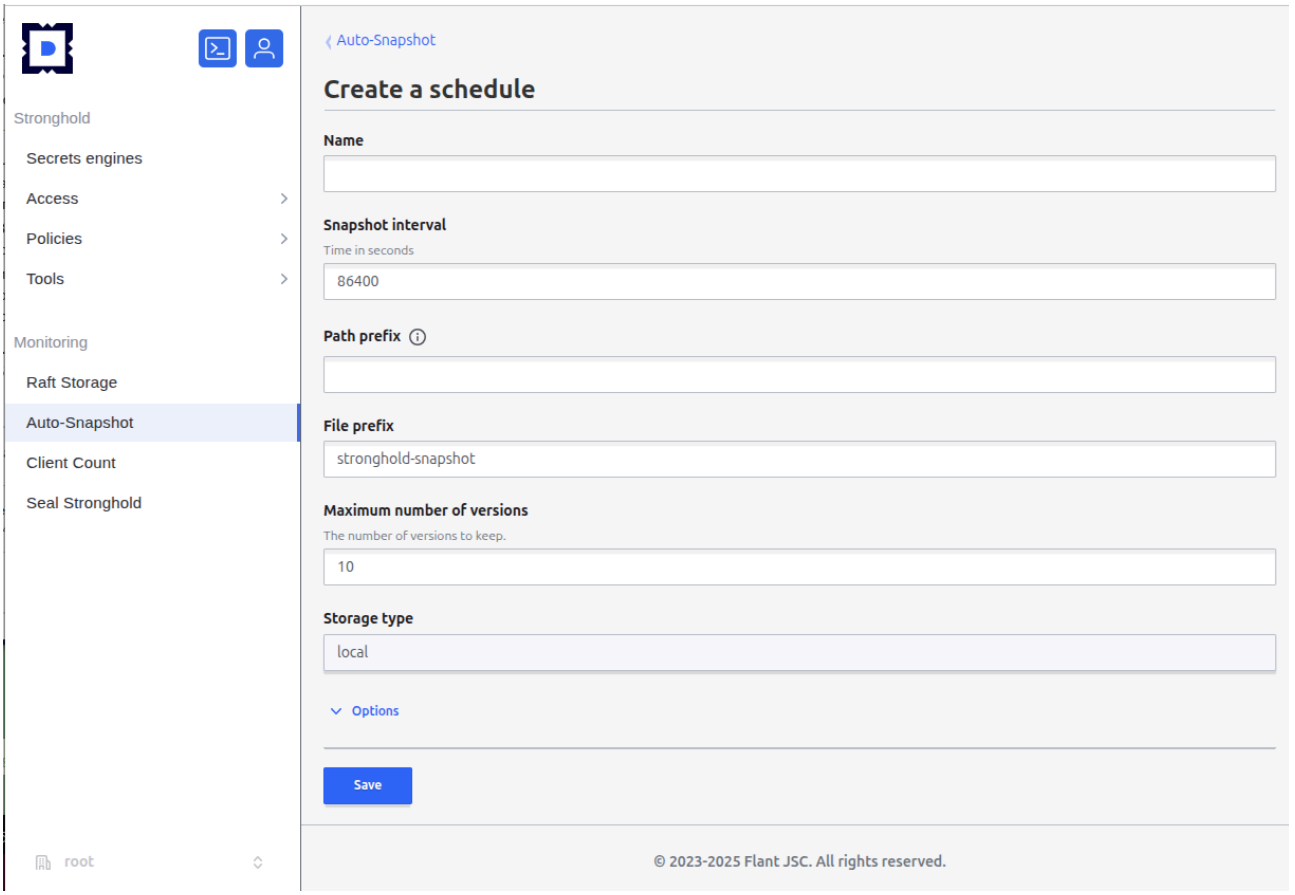

Рисунок 7 - Окно создания нового расписания

5.3.11. Просмотр информации о статусе расписания резервного копирования

Для просмотра информации о расписании резервного копирования, необходимо нажать на его наименование в списке расписаний (Рисунок 6). После нажатия будет доступно окно сведений о выбранном расписании (Рисунок 8).

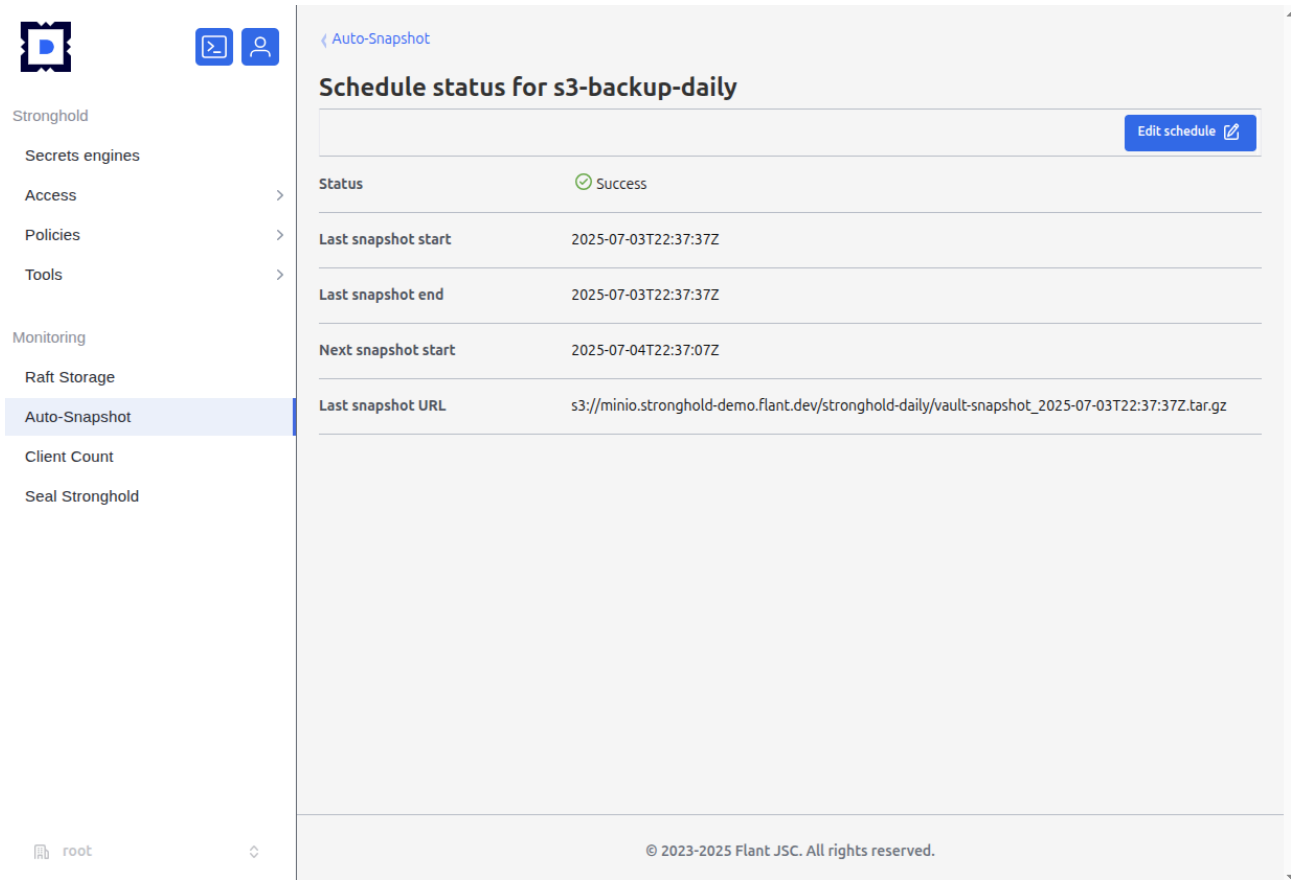

Рисунок 8 - Окно статуса расписания

5.4. Регистрация событий безопасности

В ПО «Deckhouse Stronghold» реализована возможность регистрации событий безопасности в журнал аудита. ПО «Deckhouse Stronghold» логирует записи аудита как отдельные JSON-объекты, разделенные переносами строк. Записи журнала аудита включают атрибуты, общие для всех API-эндпоинтов, а также специфичные для эндпоинта атрибуты запроса и ответа.

Записи журнала аудита представляют API-запросы, полученные ПО, и API-ответы, отправленные ПО «Deckhouse Stronghold». Таким образом, есть возможность сопоставления записей ответов с соответствующими записями запросов по полю ID объекта (.request.id), включенному в обе записи.

Атрибут	Тип	Описание
auth	object	Объект authentication, описывающий аутентифицированного субъекта, выполняющего API-вызов

Атрибут	Тип	Описание
error	string	Строка ошибки, сгенерированная запросом или возвращённая в ответе. Пропускается для успешных транзакций.
forwarded_from	string	Хост и порт узла производительности в режиме ожидания, перенаправляющего запрос. Пропускается для запросов, которые не были перенаправлены.
request	object	Объект request, описывающий детали запроса.
response	object	Объект response, описывающий детали ответа. Пропускается для записей запросов.
time	string	Дата и время API-запроса/ответа в формате ISO 8601.
type	string	Тип записи журнала аудита. Одно из значений: request или response.

Пример запроса:


```
{
  "auth": <authentication object>,
  "error": "error converting input {\"name\":\"John\"} for field \"data\": '' expected a map, got 'string'",
  "forwarded_from": "vault-1.prod.corp.com:443",
  "request": <request object>,
  "time": "2025-06-05T16:10:22.292517Z",
  "type": "request"
}
```


Пример ответа:


```
{
  "auth": <original authentication object>,
  "error": "1 error occurred:\n\t* invalid request\n\n",
  "forwarded_from": "vault-1.prod.corp.com:443",
  "request": <original request object>,
  "response": <response object>,
  "time": "2025-06-05T16:10:22.292639Z",
  "type": "response"
```

```
}
```

5.4.1. Аутентификация

Stronghold пропускает нерелевантные атрибуты объекта authentication в записи аудита. Например, accessor и client_token нерелевантны для неаутентифицированных запросов, а metadata нерелевантен, если токен аутентификации не содержит метаданных.

Атрибут	Тип	Описание
accessor	string	Accessor токена Stronghold, который выполнил запрос. Это значение хешируется по умолчанию; в примере показан нехешированный accessor токена. См. token accessors для получения дополнительной информации.
client_token	string	Токен Stronghold, который выполнил запрос, в хешированном виде.
display_name	string	Имя, связанное с токеном Stronghold, который выполнил запрос. Это нечувствительное значение, которое может помочь идентифицировать токены. Если отображаемое имя не установлено, будет показано "token".
token_type	string	Является ли токен Stronghold токеном типа service, batch или periodic. См. token types.
token_issue_time	string	Дата и время выдачи токена Stronghold в формате ISO 8601.
token_ttl	integer	Срок действия токена Stronghold в секундах на момент первой выдачи токена, относительно token_issue_time. Обратите внимание, что это значение не обновляется, когда срок жизни токена изменяется после первоначальной выдачи.
metadata	object	Метаданные, связанные с токеном Stronghold.
num_uses	integer	Когда API-запрос приводит к выдаче токена с ограниченным количеством использований, количество использований этого токена.
remaining_uses	integer	Если токен Stronghold имеет ограничение на количество использований, количество оставшихся использований до текущего запроса.
entity_created	boolean	Привел ли запрос к созданию сущности, т.е. авторизованный субъект впервые входит в Stronghold. Пропускается, если false.
entity_id	string	Если токен Stronghold связан с сущностью, ID этой сущности.

Атрибут	Тип	Описание
policies	list(string)	Список имён ACL-политик, связанных с токеном Stronghold или соответствующей сущностью.
identity_policies	list(string)	Если токен Stronghold связан с сущностью, список имён ACL-политик, связанных (напрямую или через членство в группе) с этой сущностью.
token_policies	list(string)	Список имён ACL-политик, связанных напрямую с токеном Stronghold.
no_default_policy	boolean	Не связана ли с токеном Stronghold ACL-политика "default" Stronghold. Пропускается, если false.
external_namespace_ policies	object	Если токен Stronghold связан с сущностью, и эта сущность наследует ACL-политики из пространства имён, отличного от пространства имён, в котором существует сущность, JSON-объект, содержащий булев атрибут allowed и granting_policies, список этих унаследованных ACL-политик. См. структуру JSON ниже для получения дополнительной информации о granting_policies.
policy_results	object	JSON-объект, содержащий булев атрибут allowed и granting_policies, список ACL-политик, связанных с токеном Stronghold или соответствующей сущностью, которые привели к разрешению запроса. См. структуру JSON ниже для получения дополнительной информации о granting_policies

Пример запроса:

```
{
  "accessor": "",
  "client_token": "",
  "display_name": "",
  "entity_created": "",
  "entity_id": "",
  "external_namespace_policies": {
    "allowed": true,
    "granting_policies": [
      {
        "name": "",
        "namespace_id": "",
```

```
      "namespace_path": "",
      "type": ""
    }
  ]
 },
 "identity_policies": [""],
 "metadata": {},
 "no_default_policy": false,
 "num_uses": 10,
 "policies": [""],
 "policy_results": {
  "allowed": true,
  "granting_policies": [
   {
    "name": "",
    "namespace_id": "",
    "namespace_path": "",
    "type": ""
   }
  ]
 },
 "remaining_uses": 5,
 "token_policies": [""],
 "token_issue_time": "",
 "token_ttl": 3600,
 "token_type": ""
}
```


5.4.2. Объект запроса

Следующее определение объекта запроса включает примеры данных с простыми типами (string, bool, int) и используется в других JSON-примерах, которые включают объект <request>.

Атрибут	Тип	Описание
id	string	Случайный ID, сгенерированный Stronghold для уникальной идентификации API-запроса.
operation	string	Является ли запрос операцией create, read, update, delete или list..
namespace	object	JSON-объект с уникальным id пространства имён и его path..
path	string	API-путь, который получил запрос..
response	object	Объект response, описывающий детали ответа. Пропускается для записей запросов.
request_uri	string	Исходный путь запроса из HTTP-запроса, если он отличается от path (например, когда пространство имён Stronghold указано как HTTP-заголовок).
mount_accessor	string	Уникальный идентификатор точки монтирования Stronghold (движок секретов или бэкенд аутентификации), который получил API-запрос.
mount_type	string	Тип точки монтирования Stronghold, которая получила API-запрос.
mount_running _version	string	Строка версии для точки монтирования Stronghold, которая получила API-запрос.
mount_running _sha256	string	Если точка монтирования, которая получила API-запрос, является внешним плагином, SHA-256 хеш запущенной версии плагина.
mount_is_exter nal_ plugin	boolean	Является ли точка монтирования, которая получила API-запрос, внешним плагином.
remote_addres s	string	IP-адрес клиента, выполняющего запрос, как видит Stronghold. Обратите внимание, что балансировщики нагрузки, прокси и обратные прокси могут маскировать истинный IP-адрес клиента.
remote_port	integer	Номер порта, используемый клиентом, выполняющим запрос, как видит Stronghold.

Атрибут	Тип	Описание
headers	object	JSON-объект, содержащий заголовки запроса, настроенные для логирования.
policy_override	boolean	Запросил ли клиент переопределение политики Sentinel.
client_id	string	ID клиента Stronghold, выполняющего запрос.
client_token	string	Токен Stronghold, который выполнил запрос, в хешированном виде.
client_token_accessor	string	Accessor токена Stronghold, который выполнил запрос. Это значение хешируется по умолчанию; в примере показан нехешированный accessor токена. См. token accessors для получения дополнительной информации.
client_certificate_serial_number	string	Если соединение клиента с Stronghold является взаимно аутентифицированным, серийный номер TLS-сертификата клиента.
wrap_ttl	integer	Если клиент запросил обёртывание ответа, количество секунд, в течение которых обёрнутый ответ будет доступен.
data	object	JSON-объект, содержащий полезную нагрузку запроса. Это зависит от вызванного API-эндпоинта.
replication_cluster	string	Если запрос был перенаправлен из вторичного кластера Performance Replication, имя вторичного кластера.

Пример запроса:

```
{
  "client_certificate_serial_number": "",
  "client_id": "",
  "client_token": "",
  "client_token_accessor": "",
  "data": {},
  "id": "",
  "headers": {},
  "mount_accessor": "",
  "mount_class": "",
  "mount_point": "",
  "mount_type": "",
```

```
 "mount_running_version": "",
 "mount_running_sha256": "",
 "mount_is_external_plugin": "",
 "namespace": {
  "id": "",
  "path": ""
 },
 "operation": "",
 "path": "",
 "policy_override": true,
 "remote_address": "",
 "remote_port": …,
 "replication_cluster": "",
 "request_uri": "",
 "wrap_ttl": 60
}
```


5.4.3. Объект ответа

Следующее определение объекта ответа включает примеры данных с простыми типами (string, bool, int) и используется в других JSON-примерах, которые включают объект <response>.

Атрибут	Тип	Описание
auth	object	Если запрос приводит к созданию токена, объект authentication с информацией о токене. См. authentication выше.
headers	object	Набор HTTP-заголовков ответа, отправленных плагином, который обработал API-запрос.
redirect	string	Для запросов, отправленных в бэкенды аутентификации, URL, на который бэкенд аутентификации перенаправил пользователя для дальнейшей аутентификации.
warnings	list(string)	Если API вернул одно или несколько предупреждений, список сообщений предупреждений.

Атрибут	Тип	Описание
data	object	JSON-объект, содержащий полезную нагрузку ответа. Это зависит от вызванного API-эндпоинта.
secret	object	Если API вернул арендованный секрет, JSON-объект с одним атрибутом lease_id, который идентифицирует этот арендованный секрет.
wrap_info	object	Если API вернул обёрнутый секрет, JSON-объект, содержащий свойства токена обёртывания. См. ниже атрибуты в JSON-объекте.
mount_class	string	Тип точки монтирования API, auth или secret.
mount_accessor	string	Уникальный идентификатор точки монтирования Stronghold (движок секретов или бэкенд аутентификации), который ответил на API-запрос.
mount_type	string	Тип точки монтирования Stronghold, которая ответила на API-запрос.
mount_running _plugin_version	string	Строка версии для точки монтирования Stronghold, которая ответила на API-запрос.
mount_running _sha256	string	Если точка монтирования, которая получила API-запрос, является внешним плагином, SHA-256 хеш запущенной версии плагина.
mount_is_exter nal_plugin	boolean	Является ли точка монтирования, которая получила API-запрос, внешним плагином.

Пример запроса:

```
{
  "auth": <auth>,
  "data": {},
  "headers": {},
  "mount_accessor": "",
  "mount_class": "",
  "mount_is_external_plugin": false,
  "mount_point": "",
  "mount_running_sha256": "",
  "mount_running_plugin_version": "",
```

```
 "mount_type": "",
 "redirect": "",
 "secret": {
  "lease_id": ""
 },
 "wrap_info": {
  "accessor": "",
  "creation_path": "",
  "creation_time": "",
  "token": "",
  "ttl": 60,
  "wrapped_accessor": ""
 },
 "warnings": [
  ""
 ]
}
```


5.4.4. Записи в журнале событий

Записи в журнале событий могу выглядеть следующим образом:

5.4.4.1.Успешная аутентификация

```
cat audit.log | jq '. | select(.request.path == "auth/userpass/login/user…") | select(.request.operation == "update") | select(.response.data.error| not )'
```

5.4.4.2.Неуспешная аутентификация

```
cat audit.log | jq '. | select(.request.path == "auth/userpass/login/user…") | select(.request.operation == "update") | select(.response.data.error)'
```

5.4.4.3.Успешная аутентификация по токену

```
cat audit.log | jq '. | select(.request.client_token)  | select(.type == "response") | select(.error | not)'
```

5.4.4.4.Неуспешная аутентификация по токену

```
cat audit.log | jq '. | select(.request.client_token)  | select(.type == "response") | select(.error)'
```

5.4.4.5.Успешный доступ к объектам

```
cat audit.log | jq '. | select(.request.client_token)  | select(.type == "response") | select(.error| not)'
```

5.4.4.6.Неуспешный доступ к объектам

```
cat audit.log | jq '. | select(.request.client_token)  | select(.type == "response") | select(.error)'
```

5.4.4.7.Создание пользователя

```
cat audit.log | jq '. | select(.request.path == "auth/userpass/users…") | select(.request.operation == "create")'
```

5.4.4.8.Удаление пользователя

```
cat audit.log | jq '. | select(.request.path == "auth/userpass/users/user…") | select(.request.operation == "delete")
```

5.4.4.9.Создание токена

```
cat audit.log | jq '. | select(.request.path == "auth/token/create") | select(.request.operation == "update")'
```

5.4.4.10.Удаление токена

```
cat audit.log | jq '. | select(.request.path == "auth/token/revoke")'
```

5.4.4.11.Удаление собственного токена

```
cat audit.log | jq '. | select(.request.path == "auth/token/revoke-self")'
```

5.4.4.12.Создание политики

```
cat audit.log | jq '. | select(.request.path == "sys/policies/acl/…") | select(.request.operation == "update")'
```

5.4.4.13.Удаление политики

```
cat audit.log | jq '. | select(.request.path == "sys/policies/acl/…) | select(.request.operation == "delete")'
```

5.4.4.14.Изменение назначения политик

```
cat audit.log | jq '. | select(.request.path == "auth/userpass/users/…") | select(.request.operation == "update") | select(.request.data.policies)'
```

5.4.4.15.Изменение содержания политик

```
cat audit.log | jq '. | select(.request.path == "sys/policies/acl/proverka1") | select(.request.operation == "update")'
```

Запись аудита запроса:

```
{
  "auth": {
    "accessor": "hmac-sha256:...",
    "client_token": "hmac-sha256:...",
    "display_name": "userpass-...",
    "entity_id": "62ff123b-7609-1ed9-5707-ea621da72de7",
```

```
  "metadata": { "username": "..." },
  "policies": ["default"],
  "policy_results": {
   "allowed": true,
   "granting_policies": [
    { "type": "" },
    { "name": "default", "namespace_id": "root", "type": "acl" }
   ]
  },
  "token_policies": ["default"],
  "token_issue_time": "2025-06-04T16:01:31-04:00",
  "token_ttl": 2764800,
  "token_type": "service"
 },
 "request": {
  "client_id": "62ff123b-...-1ed9-5707-...",
  "client_token": "hmac-sha256:...",
  "client_token_accessor": "hmac-sha256:...",
  "headers": { "user-agent": ["Go-http-client/1.1"] },
  "id": "79cc9b26-488f-eabf-...-...",
  "mount_class": "auth",
  "mount_point": "auth/token/",
  "mount_running_version": "v1.19.1+builtin.vault",
  "mount_type": "token",
  "namespace": { "id": "root" },
  "operation": "read",
  "path": "auth/token/lookup-self",
  "remote_address": "127.0.0.0",
  "remote_port": …
 },
 "time": "2025-06-04T20:02:46.117181Z",
 "type": "request"
}
```

Запись аудита ответа:


```
{
 "auth": {
  "accessor": "hmac-sha256:...",
  "client_token": "hmac-sha256:...",
  "display_name": "userpass-...",
  "entity_id": "62ff123b-7609-1ed9-...-...",
  "metadata": { "username": "..." },
  "policies": ["default"],
  "policy_results": {
   "allowed": true,
   "granting_policies": [
    { "type": "" },
    { "name": "default", "namespace_id": "root", "type": "acl" }
   ]
  },
  "token_policies": ["default"],
  "token_issue_time": "2025-06-04T16:01:31-04:00",
  "token_ttl": 2764800,
  "token_type": "service"
 },
 "request": {
  "client_id": "62ff123b-7609-1ed9-5707-...",
  "client_token": "hmac-sha256:...",
  "client_token_accessor": "hmac-sha256:...",
  "headers": { "user-agent": ["Go-http-client/1.1"] },
  "id": "79cc9b26-488f-eabf-...-...",
  "mount_accessor": "auth_token_d43d387d",
  "mount_class": "auth",
  "mount_point": "auth/token/",
  "mount_running_version": "v1.19.1+builtin.vault",
  "mount_type": "token",
  "namespace": { "id": "root" },
  "operation": "read",
```

```
  "path": "auth/token/lookup-self",
  "remote_address": "127.0.0.1",
  "remote_port": …
 },
 "response": {
  "data": {
   "accessor": "hmac-sha256:...",
   "creation_time": 1749067291,
   "creation_ttl": 2764800,
   "display_name": "hmac-sha256:...",
   "entity_id": "hmac-sha256:...",
   "expire_time": "2025-07-06T16:01:31.771304-04:00",
   "explicit_max_ttl": 0,
   "id": "hmac-sha256:...",
   "issue_time": "2025-06-04T16:01:31.771306-04:00",
   "meta": {
    "username": "hmac-sha256:..."
   },
   "num_uses": 0,
   "orphan": true,
   "path": "hmac-sha256:...",
   "policies": [
    "hmac-sha256:..."
   ],
   "renewable": true,
   "ttl": 2764725,
   "type": "hmac-sha256:..."
  },
  "mount_accessor": "auth_token_d43d387d",
  "mount_class": "auth",
  "mount_point": "auth/token/",
  "mount_running_plugin_version": "v1.19.1+builtin.vault",
  "mount_type": "token"
 },
 "time": "2025-06-04T20:02:46.117567Z",
```

```
  "type": "response"
}
```

6. Действия по реализации функций безопасности среды функционирования средства

В зависимости от исполнения средой функционирования ПО «Deckhouse Stronghold» может выступать ОС и ПО «Deckhouse Platform Certified Security Edition».

В случае, когда средой функционирования ПО «Deckhouse Stronghold» является ПО «Deckhouse Platform Certified Security Edition»[1], дополнительная настройка среды функционирования не требуется. ПО «Deckhouse Platform Certified Security Edition» поставляется преднастроенным в соответствии с эксплуатационной документацией на ПО «Deckhouse Platform Certified Security Edition».

В случае, когда средой функционирования ПО «Deckhouse Stronghold» является ОС, допускается функционирование только в ОС, имеющих сертификат соответствия ФСТЭК России. Настройка должна быть произведена в соответствии с методическим документом ФСТЭК России от 25 декабря 2022 г. «Рекомендации по безопасной настройке операционных систем Linux» (https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-25-dekabrya-2022-g).

6.1. Дополнительные действия по реализации функций безопасности среды функционирования средства

В среде функционирования ПО «Deckhouse Stronghold» дополнительно должны быть реализованы следующие функции безопасности:

- физическая защита;
- доверенная загрузка ПО «Deckhouse Stronghold»;
- обеспечение условий безопасного функционирования ПО «Deckhouse Stronghold»;
- обеспечение доверенного маршрута;
- обеспечение доверенного канала.
- необходимо регулярное обновление всех сред функционирования ПО «Deckhouse Stronghold» до актуальных версий с применением всех необходимых патчей безопасности с официальных сайтов разработчиков сред функционирования;
- установка, конфигурирование и управление ПО «Deckhouse Stronghold» должно осуществляться в соответствии с эксплуатационной документацией;

[1] Имеет сертификат соответствия ФСТЭК России №4860 от 04.10.2024, срок действия сертификата 04.10.2029

- ПО «Deckhouse Stronghold» должно использоваться только на совместимых с ним аппаратных мощностях и средствах;
- предоставление пользователям прав доступа к объектам доступа информационной системы обеспечивается, основываясь на задачах, решаемых пользователями в ПО «Deckhouse Stronghold» и взаимодействующими с ней информационными системами;
- каналы передачи данных ПО «Deckhouse Stronghold» должны быть либо расположены в пределах контролируемой зоны и защищены с использованием организационно-технических мер, либо, в случае их выхода за пределы контролируемой зоны, должны быть защищены путем применения средств криптографической защиты информации, сертифицированных в системе сертификации ФСБ России.

7. Модули и параметры, отвечающие за реализацию функций безопасности

Сопоставление интерфейсов ПО «Deckhouse Stronghold» и описание параметров, отвечающих за реализацию функций безопасности, приведено в электронном приложении (каталог «Электронные приложения» - «Deckhouse Stronghold. Руководство администратора» - Deckhouse Stronghold. Описание интерфейсов.xlsx).

Лист регистрации изменений

Номер изменения	Номера страниц				Всего страниц в документе	Наименоd вание и номер документа, вводящего изменения	Подпись, Ф.И.О. внесшего изменения в данный экземпляр	Дата внесения изменения в данный экземпляр	Дата введения изменения
	измененных	замененных	новых	аннулиро-ванных					
1	2	3	4	5	6	7	8	9	10