Утилиты

Данный раздел содержит основную информацию об утилитах StarVault.

1. Содержание раздела

- Совместимость версий

Рекомендации

Чтобы успешно развернуть этот механизм секретов, необходимо знать ряд важных аспектов, а также предпринять некоторые подготовительные шаги. Перед использованием этого механизма секретов или созданием ЦС для использования с этим механизмом секретов вам следует ознакомиться со всем этим.

1. Содержание раздела

- Корневые CA
- Один сертификат CA, один механизм секретов
  - Настройка эмитента по умолчанию
- Типы ключей имеют значение
  - Производительность кластера и типы ключей
- Использование иерархии CA
  - Перекрестно подписанные промежуточные продукты
- URL-адреса кластеров — это важно
- Автоматизация ротации с помощью ACME
  - Хранение сертификатов ACME
  - Ограничения роли ACME требуют EAB
  - ACME и интернет
  - Ошибки ACME в логах сервера
  - Соображения безопасности ACME
  - ACME и подсчет клиентов
- Поддержание коротких сроков жизни сертификатов ради CRL
  - Поведение NotAfter для leaf сертификатов
  - Производительность кластера и количество leaf сертификатов
- Настройка информации о выдаче /CRL/OCSP
- Распространение CRL и OCSP
- Автоматизируйте создание и очистку CRL
- Спектр поддержки отзыва
- Субъекты-эмитенты и CRL
- Автоматизация обновления сертификата leaf
- Безопасные минимумы
- Время жизни и отзыв токенов
- Безопасное использование ролей
- Телеметрия
- Аудит
- Доступ на основе ролей
- Реплицированные наборы данных
- Масштабируемость кластера
- Поддержка PSS

- API

2. Корневые CA

Хранилище безопасно, но не настолько, как лист бумаги в банковском хранилище. В конце концов, это сетевое программное обеспечение. Если ваш корневой CA размещен за пределами StarVault, не помещайте его в StarVault; вместо этого выпустите сертификат промежуточного CA с более коротким сроком действия и поместите его в StarVault. Это соответствует лучшим отраслевым практикам.

Механизм секретов поддерживает генерацию самоподписанных корневых CA, а также создание и подписание CSR для промежуточных CA. В каждом случае, по соображениям безопасности, закрытый ключ можно экспортировать только во время генерации, и возможность сделать это является частью командного пути (поэтому ее можно внести в политики ACL).

Если вы планируете использовать промежуточные CA с StarVault, рекомендуется разрешить StarVault создавать CSR и не экспортировать закрытый ключ, а затем подписывать их с помощью корневого CA (который может быть вторым монтированием механизма `pki` secrets).

3. Один сертификат CA, один механизм секретов

Механизм секретов PKI поддерживает несколько эмитентов в одном монтировании. Однако для упрощения конфигурации настоятельно рекомендуется, чтобы операторы ограничивали монтирование одним сертификатом. Если вы хотите выпускать сертификаты от нескольких разных CA, смонтируйте механизм секретов PKI в нескольких точках монтирования с отдельными сертификатами CA в каждой.

Разделение монтирования необходимо для упрощения управления разрешениями: очень немногим людям нужен доступ для выполнения операций с корневым сертификатом, но многим нужен доступ для создания leaves. Операции с корневым сертификатом обычно должны ограничиваться выпуском и отзывом промежуточных CA, что является высокопривилегированной операцией; гораздо проще проводить аудит этих операций, когда они находятся в отдельном монтировании, чем если они смешаны с ежедневным выпуском leaves.

Общим шаблоном является использование одного монтируемого центра в качестве корневого CA и использование этого CA только для подписи промежуточных CA CSR от других секретных механизмов PKI.

Чтобы сохранить старые CA активными, существует два подхода к обеспечению ротации:

1. Используйте несколько механизмов секретов. Это позволяет начать все с чистого листа, сохранив старого эмитента и CRL. Политику StarVault ACL можно обновить, чтобы запретить новую выдачу в старой точке монтирования, а роли можно заново оценить перед импортом в новую точку монтирования.
2. Использование нескольких эмитентов в одной точке монтирования. Использование старого эмитента может быть ограничено подписанием CRL, а существующие роли и политика ACL могут быть сохранены как есть. Это позволит осуществлять перекрестное подписание в рамках одной точки монтирования, а потребителям монтирования не придется обновлять свою конфигурацию. После завершения переходного периода и истечения срока действия всех прошлых сертификатов рекомендуется полностью удалить старого эмитента и всех ненужных эмитентов с перекрестными подписями из точки монтирования.

Еще один вариант использования нескольких эмитентов в одном монтировании — разделение выпуска по времени жизни TTL. Для сертификатов с коротким сроком действия промежуточный ключ, хранящийся в StarVault, часто превосходит промежуточный ключ, хранящийся в HSM. Однако для сертификатов с большим сроком действия часто важно, чтобы промежуточный ключевой материал был защищен в течение всего срока действия сертификата конечного субъекта. Это означает, что два промежуточных устройства в одном и том же хранилище - одно с поддержкой HSM и одно с поддержкой StarVault - могут удовлетворять обоим вариантам использования. Операторы могут задавать роли, устанавливающие максимальные TTL для каждого эмитента, потребители монтирования могут решать, что использовать.

3.1. Настройка эмитента по умолчанию

В целях обратной совместимости эмитент по умолчанию используется для обслуживания конечных точек PKI без явного эмитента (либо через выбор пути, либо через выбор на основе роли). Когда сертификаты отзываются и их эмитент больше не является частью этой PKI, StarVault помещает их в CRL эмитента по умолчанию. Это означает, что сохранение эмитента по умолчанию важно как для обратной совместимости при выпуске сертификатов, так и для обеспечения попадания отозванных сертификатов в CRL.

4. Типы ключей имеют значение

Некоторые типы ключей влияют на производительность. Подписание сертификатов с помощью ключа RSA будет медленнее, чем при выпуске с помощью ключа ECDSA или Ed25519. Генерация ключей (с помощью конечных точек `/issue/:role`) с использованием ключей RSA также будет медленной: Генерация ключей RSA включает в себя поиск подходящих случайных простых чисел, в то время как ключи Ed25519 могут быть случайными данными. С увеличением количества бит (RSA 2048 → 4096 или ECDSA P-256 → P-521) время подписи также увеличивается.

Это имеет значение в обоих направлениях: дороже не только выдача, но и проверка соответствующей подписи (скажем, в TLS). Тщательное рассмотрение типов ключей как эмитента, так и выданных сертификатов может оказать значительное влияние на производительность не только StarVault, но и систем, использующих эти сертификаты.

4.1. Производительность кластера и типы ключей

Проект benchmark-vault можно использовать для измерения производительности экземпляра StarVault PKI. В целом, следует обратить внимание на некоторые моменты:

- Генерация ключей RSA происходит гораздо медленнее и с высокой степенью вариативности, чем генерация ключей EC. Если производительность и пропускная способность важны, рассмотрите возможность использования EC-ключей (включая NIST P-curves и Ed25519) вместо RSA.
- Запросы на подписание ключей через `/pki/sign` будут быстрее, чем `/pki/issue`, особенно для ключей RSA: это устраняет необходимость для StarVault генерировать ключевой материал и позволяет подписать ключевой материал, предоставленный клиентом. Этот шаг подписания является общим для обеих конечных точек, таким образом, генерация ключей - это чисто служебные расходы, если у клиента есть достаточно безопасный источник энтропии.
- Тип ключа CA также имеет значение: использование RSA CA приведет к RSA-подписи и займет больше времени, чем ECDSA или Ed25519 CA.
- Хранение - важный фактор: при BYOC Revocation использование `no_store=true` все равно дает возможность отзывать сертификаты, а журналы аудита можно использовать для отслеживания выдачи. Кластеры, использующие удаленное хранилище в медленной сети и использующие `no_store=false`, приведут к дополнительным задержкам при выпуске. Добавление лизинга для каждого выпущенного сертификата усугубляет проблему.
    - Хранение слишком большого количества сертификатов приводит к увеличению времени работы `LIST /pki/certs`, включая время на очистку экземпляра. Поэтому при больших масштабах развертывания (>= 250 тыс. активных сертификатов) рекомендуется использовать журналы аудита для отслеживания сертификатов за пределами StarVault.

В качестве общего сравнения на неопределенном оборудовании, используя `benchmark-vault` в течение 30 секунд на локальном экземпляре хранилища с одним узлом, поддерживаемом raft:

- StarVault может выпустить 300 тысяч сертификатов, используя EC P-256 для ключей CA & leaf и без хранения.
    - Переход на хранение leaves снижает это число до 65 тысяч, а с арендой - только до 20 тысяч.
- Используя большие и дорогие ключи RSA-4096 бит, StarVault может выдать только 160 leaves, независимо от того, использовались ли хранилища или аренда. Время генерации 95% ключей превышает 10 с.

- Для сравнения, используя ключи P-521, StarVault может выдать около 30 тысяч leaves без аренды и 18 тысяч с арендой.

Эти цифры приведены только для примера, чтобы показать, как различные типы ключей могут повлиять на производительность кластера PKI.

Использование ACME вносит дополнительную задержку в эти цифры, поскольку необходимо хранить сертификаты и выполнять проверку вызовов.

5. Использование иерархии CA

Обычно рекомендуется использовать иерархическую структуру CA, с корневым сертификатом, который выдает один или несколько промежуточных (в зависимости от использования), которые, в свою очередь, выдают leaves сертификаты.

Это позволяет усилить гарантии хранения или политики защиты корневого CA, в то же время позволяя StarVault управлять промежуточными CA и выдачей leaves. Разные промежуточные центры могут выдаваться для разных целей, например для подписи VPN, подписи электронной почты или тестирования в сравнении с производственными службами TLS. Это помогает ограничить CRL конкретными целями: например, VPN-службам не важен отозванный набор сертификатов подписи электронной почты, если они используют отдельные сертификаты и разных посредников, и поэтому им не нужно содержимое обоих CRL. Кроме того, это позволяет посредникам с повышенным риском (например, выпускающим сертификаты подписи электронной почты с большим сроком действия) иметь HSM-поддержку, не влияя на производительность более простых в обращении посредников и сертификатов (например, TLS-посредников).

StarVault поддерживает использование параметра `allowed_domains` для ролей и параметра `permitted_dns_domains` для установки расширения Name Constraints для корневого и промежуточного поколений. Это позволяет на нескольких уровнях разделить проблемы между службами на основе TLS.

5.1. Перекрестно подписанные промежуточные продукты

При перекрестном подписании промежуточных цепочек от двух разных корня в монтировании StarVault PKI будут существовать два отдельных промежуточных эмитента. Чтобы корректно обслуживать перекрестно подписанную цепочку в запросах на выдачу, необходимо переопределить `manual_chain` на одном или обоих промежуточных эмитентах. Его можно построить в следующем порядке:

- текущий эмитент (`self`)
- текущий корень
- другая копия текущего промежуточного
- другой корень

Все запросы на подпись к текущему эмитенту теперь будут представлять полную кросс-подписанную цепочку.

6. URL-адреса кластеров — это важно

В StarVault есть поддержка шаблонизированных URL-адресов AIA. При конфигурации URL-адреса для каждого кластера, указывающего на этот кластер Performance Replication, информация AIA будет автоматически указывать на кластер, выпустивший этот сертификат.

В StarVault, с поддержкой ACME, такая же конфигурация используется для того, чтобы клиенты ACME могли обнаружить URL этого кластера.

Важно убедиться, что эта конфигурация актуальна и поддерживается правильно, всегда указывая на адрес кластера PR узла (который может быть адресом Load Balanced или DNS Round-Robbin). Если эта конфигурация не установлена на каждом кластере Performance Replication, выпуск сертификатов (через REST и/или через ACME) будет неудачным.

7. Автоматизация ротации с помощью ACME

В движок PKI Engine добавлена поддержка протокола Automatic Certificate Management Environment (ACME). Это стандартизированный способ обработки проверки, выдачи, ротации и отзыва сертификатов сервера.

Многие экосистемы, от веб-серверов, таких как Caddy, Nginx и Apache, до сред оркестровки, таких как Kubernetes (через cert-manager), поддерживают выдачу сертификатов по протоколу ACME. Для развертываний без встроенной поддержки существуют отдельные инструменты, такие как certbot, поддерживающие получение и обновление сертификатов от имени потребителей. StarVault's PKI Engine включает только серверную поддержку ACME; клиентская функциональность не предусмотрена.

Сервер StarVault PKI ACME ограничивает срок действия сертификата максимум 90 днями, независимо от роли и/или глобальных ограничений. Более короткие сроки действия можно установить, ограничив TTL роли до 90 дней. В соответствии с рекомендациями Let's Encrypt мы не поддерживаем необязательные параметры запроса порядка `NotBefore` и `NotAfter`.

7.1. Хранение сертификатов ACME

Поскольку для работы ACME требуются сохраненные сертификаты, приведенные ниже заметки об автоматизации очистки особенно важны для долгосрочной успешной работы кластера PKI. ACME также представляет дополнительные типы ресурсов (учетные записи, заказы, авторизации и вызовы), которые должны быть очищены с помощью опции `tidy_acme=true`. Заказы, авторизации и вызовы очищаются на основе параметра `safety_buffer`, но учетные записи могут жить дольше, чем их последний выпущенный сертификат, управляя параметром `acme_account_safety_buffer`.

Как следствие вышесказанного, а также обсуждения в разделе "Масштабируемость кластера", поскольку для этих ролей установлен параметр `no_store=false`, ACME может выдавать сертификаты только на активных узлах кластеров PR; резервные узлы, если с ними связаться, будут прозрачно пересылать все запросы на активный узел.

7.2. Ограничения роли ACME требуют EAB

Поскольку ACME по умолчанию не имеет внешнего механизма авторизации и неаутентифицирован с точки зрения StarVault, использование ролей с ACME в конфигурации по умолчанию имеет ограниченную ценность, поскольку любой клиент ACME может запросить сертификаты под любой ролью, подтвердив владение идентификаторами запрошенных сертификатов.

Для решения этой проблемы существует два возможных подхода:

1. Используйте ограничительные настройки `allowed_roles`, `allowed_issuers` и `default_directory_policy` ACME, чтобы разрешить использование только одной роли и эмитента. Это предотвращает выбор пользователей, позволяет наложить некоторые глобальные ограничения на выпуск и не требует от клиентов ACME иметь (при первоначальной настройке) доступ к токену StarVault и другим механизмам для приобретения токена StarVault EAB ACME.
2. Используйте более разрешительную конфигурацию с `eab_policy=always-required`, чтобы разрешить больше ролей и пользователей для выбора ролей, но привязать ACME-клиентов к токену StarVault, который может быть соответствующим образом подключен с помощью ACL к определенным наборам одобренных ACME-каталогов.

Выбор подхода зависит от политики организации, желающей использовать ACME.

Еще одним следствием неаутентифицированных запросов ACME является то, что шаблонизация ролей, основанная на информации о сущностях, не может быть использована, поскольку нет токена.Следовательно, нет сущности, связанной с запросом, даже если используется привязка EAB.

7.3. ACME и интернет

Использование ACME возможно через интернет; публичные CA, такие как Let's Encrypt, предоставляют такую услугу. Аналогичным образом, организации, имеющие внутреннюю инфраструктуру PKI, могут захотеть выдать сертификаты

серверов частям инфраструктуры, находящимся за пределами их внутренних границ сети, из общедоступного экземпляра StarVault. По умолчанию, без применения ограничительной политики `eab_policy`, это приводит к сложной модели угроз: любой внешний клиент, который может подтвердить владение доменом, может выпустить сертификат под этим CA, который может считаться более доверенным для данной организации.

В связи с этим мы настоятельно рекомендуем публичным экземплярам StarVault следить за тем, чтобы операторы монтирования PKI требовали ограничительной конфигурации `eab_policy=always-required`. Системные администраторы экземпляров StarVault могут обеспечить это, установив переменную окружения `STARVAULT_DISABLE_PUBLIC_ACME=true`.

7.4. Ошибки ACME в логах сервера

Поскольку клиенту ACME не всегда можно доверять (поскольку регистрация учетной записи может быть не связана с действительным токеном StarVault, если не используется EAB), многие сообщения об ошибках попадают в журналы сервера StarVault из соображений безопасности. При устранении проблем с клиентами, запрашивающими сертификаты, сначала проверьте журналы клиента, если таковые имеются (например, certbot укажет местоположение журнала при ошибках), а затем соотнесите с журналами сервера StarVault, чтобы определить причину сбоя.

7.5. Соображения безопасности ACME

ACME позволяет любому клиенту использовать StarVault для выполнения какого-либо внешнего вызова; хотя дизайн ACME пытается минимизировать эту область и запрещает выдачу при обращении к неправильным серверам, он не может учесть все возможные реализации удаленных серверов. Сервер StarVault ACME выполняет три типа запросов:

1. DNS-запросы для `_acme-challenge.<domain>`, что должно быть наименее инвазивным и наиболее безопасным.
2. Запросы TLS ALPN для протокола `acme-tls/1`, которые должны быть безопасно обработаны TLS до того, как будет вызван какой-либо код приложения.
3. HTTP-запросы к http://<domain>/.well-known/acme-challenge/<token>;, что может быть проблематично, исходя из дизайна сервера; если все запросы, независимо от пути, будут рассматриваться одинаково и считаться доверенными, это может привести к тому, что StarVault будет использоваться для выполнения (недействительных) запросов. В идеале, любые реализации таких серверов должны быть обновлены, чтобы игнорировать такие запросы на проверку ACME или блокировать доступ из StarVault к этой службе.

Во всех случаях клиенту ACME не возвращается никакой информации об ответе, представленном удаленным сервером.

При работе StarVault в нескольких сетях следует учитывать, что сервер StarVault ACME не накладывает ограничений на запрос путей проверки идентификаторов клиента/назначения; клиент может использовать HTTP-запрос, чтобы заставить StarVault связаться с сервером в сети, к которой иначе он не мог бы получить доступ.

7.6. ACME и подсчет клиентов

В StarVault ACME вносит иной вклад в метрики использования, чем другие взаимодействия с PKI Secrets Engine. Из-за использования неаутентифицированных запросов (которые не генерируют токены StarVault), он не будет учитываться в традиционных API журналах активности. Вместо этого сертификаты, выпущенные через ACME, будут учитываться по их уникальным идентификаторам сертификатов (комбинация CN, DNS SAN и IP SAN). Они создадут стабильный идентификатор, который будет согласован с обновлениями, другими клиентами ACME, монтированиями и пространствами имен, внося свой вклад в журнал активности в настоящее время как токен, не являющийся сущностью, приписанный первому монтированию, которое создало этот запрос.

8. Поддержание коротких сроков жизни сертификатов ради CRL

Этот механизм секретов соответствует философии StarVault о недолговечности секретов. Поэтому не ожидается, что CRL будут разрастаться; единственный способ возврата закрытого ключа - это передача его запрашивающему клиенту (этот механизм секретов не хранит сгенерированные закрытые ключи, за исключением сертификатов CA). В

большинстве случаев, если ключ утерян, сертификат можно просто проигнорировать, так как срок его действия истечет в ближайшее время.

Если сертификат действительно должен быть отозван, можно использовать обычную функцию отзыва StarVault, и любое действие по отзыву приведет к перегенерации CRL. Когда CRL перегенерируется, все сертификаты с истекшим сроком действия удаляются из CRL (а все отозванные сертификаты с истекшим сроком действия удаляются из хранилища механизма секретов). Это дорогостоящая операция! Из-за структуры стандарта CRL StarVault должен считать все отозванные сертификаты в память, чтобы перестроить CRL, а клиенты должны получить обновленный CRL.

Этот механизм секретов не поддерживает несколько конечных точек CRL со скользящими окнами дат; часто такие механизмы имеют точку перехода с разницей в несколько дней, но это попадает в ожидаемую область фактических сроков действия сертификатов, выдаваемых этим механизмом секретов.

Хорошим эмпирическим правилом для этого механизма секретов будет просто не выдавать сертификаты со сроком действия, превышающим максимальное комфортное время жизни CRL. В качестве альтернативы вы можете управлять поведением кэширования CRL на клиенте, чтобы проверки происходили чаще.

Часто используется несколько конечных точек в случае, если одна конечная точка CRL не работает, чтобы клиентам не приходилось решать, что делать в случае отсутствия ответа. Запустите StarVault в режиме HA, и конечная точка CRL должна быть доступна, даже если определенный узел не работает.

При наличии нескольких эмитентов в одной точке монтирования, разные эмитенты могут иметь разные CRL (в зависимости от темы и материала ключа). Это означает, что StarVault может потребоваться перегенерировать несколько CRL. Это еще раз подтверждает необходимость поддерживать короткие TTL и по возможности избегать отзыва.

Поддерживаются два дополнительных механизма отзыва: Delta CRLs, который позволяет перестраивать небольшие, инкрементные дополнения к последнему полному CRL, и OCSP, который позволяет отвечать на запросы статуса отзыва для отдельных сертификатов. В сочетании с новой функцией автоматического перестроения CRL это означает, что шаг отзыва не так затратен (поскольку CRL не всегда перестраивается при каждом отзыве), помимо соображений хранения. Однако, хотя операция перестройки по-прежнему может быть дорогостоящей при большом количестве сертификатов, она будет выполняться по расписанию, а не по требованию.

8.1. Поведение NotAfter для leaf сертификатов

В PKI Secrets Engine существует параметр `leaf_not_after_behavior` для эмитентов. Это позволяет изменять поведение при выдаче: должен ли StarVault выдавать `err`, предотвращая выдачу leaf сертификата с большим сроком действия, чем у эмитента, без уведомления усекать до значения `NotAfter` эмитента или разрешать (`permit`) более длительное истечение срока действия.

Настоятельно рекомендуется использовать `err` или `truncate` для промежуточных сертификатов; `permit` полезен только для корневых сертификатов, так как промежуточные `NotAfter` проверяются при проверке представленных цепочек.

В сочетании с каскадным истечением срока действия с более долгоживущими корневыми (возможно, в диапазоне 2-10 лет), более короткоживущими промежуточными (возможно, в диапазоне от 6 месяцев до 2 лет) и короткоживущими leaf сертификатами (в диапазоне от 30 до 90 дней) и стратегиями ротации, обсуждаемыми в других разделах, это позволит сохранить CRL достаточно маленькими.

8.2. Производительность кластера и количество leaf сертификатов

Как уже говорилось выше, поддержание коротких TTL (или использование `no_store=true`) и отказ от аренды важны для здорового кластера. Однако важно отметить, что это проблема масштаба: 10-1000 долгоживущих, хранящихся сертификатов, вероятно, в порядке, но 50k-100k становятся проблемой, а 500k+ хранящихся, непросроченных сертификатов могут негативно повлиять даже на большие кластеры StarVault - даже с короткими TTL!

Однако после истечения срока действия этих сертификатов операция очистки очистит CRL и хранилище кластера StarVault.

Обратите внимание, что оценка риска компрометации сертификата может означать, что определенные типы сертификатов должны всегда выпускаться с `no_store=false`; даже недолговечные сертификаты с широким подстановочным знаком (например, `*.example.com`) могут быть достаточно важны, чтобы иметь точный контроль над отзывом. Однако внутренняя служба с хорошо скопированным сертификатом (скажем, `service.example.com`) может представлять достаточно низкий риск, чтобы выпустить 90-дневный TTL с `no_store=true`, предотвращая необходимость отзыва в маловероятном случае компрометации.

Наличие более короткого TTL снижает необходимость отзыва сертификата (но не может предотвратить его полностью) и уменьшает последствия любой такой компрометации.

Функция Bring-Your-Own-Cert (BYOC) в PKI Secret Engine позволяет отзывать сертификаты, которые ранее не хранились (например, выданные через роль с параметром `no_store=true`). Это означает, что установка `no_store=true` теперь безопасна для глобального использования, независимо от важности выпущенных сертификатов (и вероятности их отзыва).

9. Настройка информации о выпуске/CRL/OCSP

Этот механизм секретов обслуживает CRL из предсказуемого места, но механизм секретов не может знать, где он запущен. Поэтому необходимо настроить нужные URL-адреса для сертификата-эмитента, точек распространения CRL и серверов OCSP вручную с помощью конечной точки `config/urls`. Поддерживается возможность использования более одного из них путем передачи нескольких URL в качестве строкового параметра, разделенного запятой.

При использовании нескольких эмитентов в одном монтировании рекомендуется использовать поля AIA для каждого эмитента, а не глобальный вариант (`/config/urls`). Это необходимо для корректности: эти поля используются для построения цепочек и автоматического обнаружения CRL в некоторых приложениях. Если они указывают на неверную информацию об эмитенте, эти приложения могут выйти из строя.

10. Распространение CRL и OCSP

И CRL, и OCSP позволяют проверить статус отзыва сертификатов. Оба эти метода включают в себя внутреннюю безопасность и аутентичность (ответы и CRL, и OCSP подписываются выдавшим их CA внутри StarVault). Это означает, что оба метода можно распространять по небезопасным и неаутентифицированным каналам, таким как HTTP.

Реализация OCSP для GET-запросов может приводить к периодическим ошибкам 400, когда закодированный OCSP-запрос содержит последовательные символы `'/'`. Пока эта проблема не будет решена, рекомендуется использовать OCSP-запросы на основе POST.

11. Автоматизируйте создание и очистку CRL

PKI Secrets Engine поддерживает автоматическое создание CRL (включая опциональные Delta CRL, которые могут создаваться чаще, чем полные CRL) через конечную точку `/config/crl`. Кроме того, через конечную точку `/config/auto-tidy` можно автоматически настраивать очистку отозванных и просроченных сертификатов. Обе эти функции должны быть включены, чтобы обеспечить совместимость с более широкой экосистемой PKIX и производительность кластера.

12. Спектр поддержки отзыва

Механизм секретов PKI способен поддерживать различные размеры кластеров и количество аннулированных сертификатов.

Пользователям с небольшим количеством отзывов и тем, кто хочет получить единое представление и имеет пропускную способность межкластерного канала связи, мы рекомендуем включить автоматическое восстановление CRL, кросс-

кластерных очередей отзывов и кросс-кластерных CRL. Это позволит всем потребителям CRL иметь наиболее точное представление об отзывах, независимо от того, с каким кластером они общаются.

Если единый CRL становится слишком большим для базового механизма хранения или для отдельного узла, мы рекомендуем использовать OCSP вместо CRL. Они содержат гораздо меньше записей, а флаг `disabled` CRL не зависит от `unified_crls`, что позволяет сохранить унифицированный OCSP.

Однако, если кросс-кластерный трафик становится слишком большим (или если CRL все еще необходимы в дополнение к OCSP), мы рекомендуем разделить CRL между разными кластерами. Это было стандартным поведением StarVault, но с введением шаблонизированной информации AIA для каждого кластера, информация Authority Information Access (AIA) сертификата листа будет указывать непосредственно на кластер, который его выпустил, что позволит приложению определить правильный CRL для этого сертификата. Это более точно имитирует поведение разделения CRL в Let's Encrypt.

Такое разделение можно использовать и для OCSP, если кросс-кластерный трафик для записей об отзыве становится слишком большим.

Для пользователей, которые хотят управлять отзывом вручную, использование журналов аудита для отслеживания выпуска сертификатов позволит внешней системе определить, какие сертификаты были выпущены. Их можно вручную отследить на предмет отзыва, а также создать собственный CRL, используя отслеженные извне отзывы. Это позволит использовать роли, установленные на `no_store=true`, так что StarVault будет использоваться только как орган выдачи и не будет хранить никаких сертификатов, выпущенных или отозванных. Для самых больших объемов отзыва это может быть лучшим вариантом.

Последний подход может быть использован как для создания внешних унифицированных, так и разделенных CRL. Если размер одного внешнего унифицированного CRL становится неоправданно большим, сертификаты каждого кластера могут иметь информацию AIA, указывающую на хранящийся и поддерживаемый извне разделенный CRL. Однако на данный момент StarVault не имеет механизма подписи запросов OCSP.

13. Субъекты-эмитенты и CRL

Библиотека Go x509 имеет свой механизм разбора и структурирования субъектов сертификата. С эмитентами, созданными в StarVault, все в порядке, но при использовании сертификатов CA, созданных извне, они могут быть разобраны неправильно во всех частях PKI. В частности, CRL содержат (измененную) копию имени эмитента. Этого можно избежать, используя OCSP для отслеживания отзыва, но учтите, что характеристики производительности у OCSP и CRL разные.

Начиная с Go 1.20, Go корректно форматирует имя издателя CRL, и это уведомление не применяется.

14. Автоматизация обновления сертификата leaf

Чтобы управлять сертификатами для служб в масштабе, лучше всего автоматизировать обновление сертификатов, насколько это возможно. StarVault Agent поддерживает автоматическое обновление запрошенных сертификатов на основе поля `validTo`. Другие решения могут включать использование cert-manager в Kubernetes или OpenShift, подкрепленного CA StarVault.

15. Безопасные минимумы

С момента своего создания этот механизм секретов использует SHA256 для хэшей подписей, а не SHA1. Для ключей RSA также используется минимум 2048 бит. Программное обеспечение, способное работать с подписями SHA256, должно также уметь работать с 2048-битными ключами, а 1024-битные ключи считаются небезопасными и запрещены в Интернет PKI.

16. Время жизни и отзыв токенов

Когда срок действия токена истекает, он аннулирует все связанные с ним аренды. Это означает, что долгоживущие CA-церты нуждаются в соответствующих долгоживущих токенах, о чем легко забыть. Корневые и промежуточные сертификаты CA не имеют связанных с ними аренд, чтобы предотвратить непреднамеренный отзыв, если не используется токен с достаточно большим сроком действия. Чтобы отозвать эти сертификаты, используйте конечную точку `pki/revoke`.

17. Безопасное использование ролей

Механизм секретов StarVault PKI поддерживает множество опций для ограничения выдачи через роли. Необходимо тщательно продумать конструкцию, чтобы убедиться, что не дается больше разрешений, чем необходимо. Кроме того, роли, как правило, должны выполнять одну задачу; несколько ролей предпочтительнее, чем слишком свободные роли, позволяющие произвольную выдачу (например, `allow_any_name` следует использовать редко, если вообще использовать).

- `allow_any_name` обычно должно быть установлено значение `false`; это значение используется по умолчанию.
- `allow_localhost` обычно должно быть установлено в `false` для производственных сервисов, если только не предполагается прослушивание `localhost`
- Если нет необходимости, параметр `allow_wildcard_certificates` обычно должен быть установлен в `false`. Это значение не используется по умолчанию из-за соображений обратной совместимости
  - Это особенно необходимо, если включены функции `allow_subdomains` или `allow_glob_domains`
- `enforce_hostnames` обычно должен быть включен для служб TLS; это значение используется по умолчанию
- `allow_ip_sans`, как правило, должно быть установлено значение `false` (но по умолчанию установлено значение `true`), если только сертификаты IP-адресов не требуются в явном виде.
- при использовании коротких TTL (< 30 дней) или при большом количестве выпусков обычно рекомендуется установить значение `no_store` в `true` (по умолчанию `false`). Это предотвращает отзыв, но обеспечивает более высокую пропускную способность, поскольку StarVault больше не нужно хранить каждый выпущенный сертификат. Подробнее об этом говорится в разделе "Репликация наборов данных".
- Не используйте роли с `root` сертификатами (`issuer_ref`). Корневые сертификаты, как правило, должны выдавать только промежуточные сертификаты (см. раздел об иерархии CA выше), которые не зависят от ролей.
- ограничьте `key_usage` и `ext_key_usage`; не пытайтесь разрешить все варианты использования для всех целей. Как правило, значения по умолчанию полезны для аутентификации клиента и сервера TLS.

18. Телеметрия

Помимо стандартной телеметрии StarVault по обработке запросов, PKI предоставляет метрики количества и длительности для вызовов issue, sign, sign-verbatim и revoke. Ключи метрик имеют вид `mount-path`, `operation`, `[failure]` с метками для имени роли.

Обратите внимание, что эти метрики рассчитаны на один узел, поэтому их необходимо агрегировать по узлам и кластерам.

19. Аудит

Поскольку по умолчанию StarVault HMAC проверяет строковые ключи, необходимо настроить монтирование секретов PKI, чтобы получить точное представление о выдаче, происходящей под этим монтированием.

В зависимости от использования StarVault, CRL (и, в редких случаях, цепочки CA) могут стать довольно большими. По этой причине мы не рекомендуем отменять HMAC для поля `crl`, но обратите внимание, что в рекомендациях ниже предлагается отменять HMAC для ответного параметра `certificate`, в котором CRL может быть предоставлен через конечную точку API `/pki/cert/crl`. Кроме того, `http_raw_body` может использоваться для возврата CRL как в PEM, так и в сыром бинарном DER-формате, поэтому рекомендуется не удалять это поле, чтобы не испортить формат журнала.

Если это делается только с помощью устройства аудита syslog, StarVault может отклонять запросы (с сообщением `500 Internal Error`) после выполнения действия на сервере, потому что он не смог записать сообщение в журнал.

В качестве обходного пути предлагается оставить поля `certificate` и `crl` ответа HMACed и/или включить тип журнала аудита file.

Ниже приведены некоторые предлагаемые ключи для отмены HMAC для запросов:

- `csr` - запрашиваемый CSR для подписи,
- `certificate` - запрашиваемый самоподписанный сертификат для повторной подписи или при импорте эмитентов
- Различные переопределяющие параметры, связанные с выпуском, такие как:
  - `issuer_ref` - эмитент, запрошенный для подписания этого сертификата
  - `common_name` - запрошенное общее имя,
  - `alt_names` - альтернативные запрошенные SAN типа DNS для этого сертификата
  - `other_sans` - другие (не-DNS, не-Email, не-IP, не-URI) запрошенные SAN для этого сертификата
  - `ip_sans` - запрошенные SAN типа IP для этого сертификата
  - `uri_sans` - запрашиваемые SAN типа URI для данного сертификата
  - `ttl` - запрашиваемая дата истечения срока действия данного сертификата
  - `not_after` - запрашиваемая дата истечения срока действия данного сертификата
  - `serial_number` - запрашиваемый серийный номер субъекта
  - `key_type` - запрашиваемый тип ключа
  - `private_key_format` - запрашиваемый формат ключа, который также используется для формата открытого сертификата
- Различные параметры генерации, связанные с ролью или эмитентом, такие как:
  - `managed_key_name` - при создании эмитента запрашиваемое имя управляемого ключа,
  - `managed_key_id` - при создании эмитента запрашиваемый идентификатор управляемого ключа,
  - `ou` - организационная единица субъекта,
  - `organization` - организация субъекта
  - `country` - код страны субъекта
  - `locality` - населенный пункт субъекта
  - `province` - провинция субъекта
  - `street_address` - уличный адрес субъекта
  - `postal_code` - почтовый индекс субъекта
  - `permitted_dns_domains` - разрешенные DNS-домены
  - `policy_identifiers` - запрашиваемые идентификаторы политики при создании роли
  - `ext_key_usage_oids` - расширенные OID использования ключей для запрашиваемого сертификата

Ниже приведены некоторые предлагаемые ключи для отмены HMAC для ответов:

- `certificate` - выданный сертификат
- `issuing_ca` - сертификат центра сертификации, выдавшего запрошенный сертификат
- `serial_number` - серийный номер выданного сертификата

- `error` - для отображения ошибок, связанных с запросом
- `ca_chain` - необязательно из-за шума; полная цепочка центра сертификации издателя запрошенного сертификата

Этот список параметров для un-HMAC приведен в качестве примера и не может быть исчерпывающим.

Следующие ключи рекомендуется не использовать в un-HMAC из-за их чувствительности:

- `private_key` - этот параметр ответа содержит закрытые ключи, сгенерированные StarVault во время эмиссии
- `pem_bundle` этот параметр запроса используется только на пути эмитент-импорт и может содержать конфиденциальный материал закрытого ключа.

20. Доступ на основе ролей

StarVault поддерживает ACL-политики на основе путей для ограничения доступа к различным путям в StarVault.

Ниже приведен сжатый пример ACL для механизма PKI Secrets Engine. Это всего лишь предложение; могут быть использованы и другие персоны и подходы к политике.

Предлагаются следующие роли:

- Оператор (`Operator`); привилегированный пользователь, который управляет состоянием подсистемы PKI; управляет эмитентами и ключевым материалом.
- Агент (`Agent`); полупривилегированный пользователь, который управляет ролями и занимается отзывом от имени оператора; может также заниматься делегированной выдачей. Его также можно назвать администратором или менеджером ролей.
- Продвинутый (`Advanced`); потенциальный пользователь или служба, имеющая доступ к дополнительным API выдачи.
- Запрашивающий пользователь (`Requester`); низкоуровневый пользователь или служба, которая просто запрашивает сертификаты.
- Неавторизованный (`Unauthed`); любой произвольный пользователь или служба, не имеющая токена StarVault.

Для этих ролей предлагаются следующие ACL в сжатой табличной форме:

Путь	Операции	Оператор	Агент	Продвинутый	Запрашивающий пользователь	Неавторизованный
`/ca(/pem)?`	чтение	да	да	да	да	да
`/ca_chain`	чтение	да	да	да	да	да
`/crl(/pem)?`	чтение	да	да	да	да	да
`/crl/delta(/pem)?`	чтение	да	да	да	да	да
`/cert/:serial(/raw(/pem)?)?`	чтение	да	да	да	да	да
`/issuers`	список	да	да	да	да	да
`/issuer/:issuer_ref/(json¦der¦pem)`	чтение	да	да	да	да	да
`/issuer/:issuer_ref/crl(/der¦/pem)?`	чтение	да	да	да	да	да
`/issuer/:issuer_ref/crl/delta(/der¦/pem)?`	чтение	да	да	да	да	да
`/ocsp/<request>`	чтение	да	да	да	да	да
`/ocsp`	запись	да	да	да	да	да
`/certs`	список	да	да	да	да	
`/revoke-with-key`	запись	да	да	да	да	

/roles	список	да	да	да	да	
/roles/:role	чтение	да	да	да	да	
/(issue¦sign)/:role	запись	да	да	да	да	
/issuer/:issuer_ref/(issue¦sign)/:role	запись	да	да	да		
/config/auto-tidy	чтение	да	да			
/config/ca	чтение	да	да			
/config/crl	чтение	да	да			
/config/issuers	чтение	да	да			
/crl/rotate	чтение	да	да			
/crl/rotate-delta	чтение	да	да			
/roles/:role	запись	да	да			
/issuer/:issuer_ref	чтение	да	да			
/sign-verbatim(/:role)?	запись	да	да			
/issuer/:issuer_ref/sign-verbatim(/:role)?	запись	да	да			
/revoke	запись	да	да			
/tidy	запись	да	да			
/tidy-cancel	запись	да	да			
/tidy-status	чтение	да	да			
/config/auto-tidy	запись	да				
/config/ca	запись	да				
/config/crl	запись	да				
/config/issuers	запись	да				
/config/keys	запись, чтение	да				
/config/urls	запись, чтение	да				
/issuer/:issuer_ref	запись	да				
/issuer/:issuer_ref/revoke	запись	да				
/issuer/:issuer_ref/sign-intermediate	запись	да				
/issuer/issuer_ref/sign-self-issued	запись	да				
/issuers/generate//	запись	да				
/issuers/import/+	запись	да				
/intermediate/generate/+	запись	да				
/intermediate/cross-sign	запись	да				
/intermediate/set-signed	запись	да				
/keys	список	да				

`/key/:key_ref`	запись, чтение	да				
`/keys/generate/+`	запись	да				
`/keys/import`	запись	да				
`/root/generate/+`	запись	да				
`/root/sign-intermediate`	запись	да				
`/root/sign-self-issued`	запись	да				
`/root/rotate/+`	запись	да				
`/root/replace`	запись	да				

При использовании управляемых ключей (`managed keys`) операторам может потребоваться доступ для чтения настраиваемых данных точки монтирования (`/sys/mounts`), а также доступ для использования или управления управляемыми ключами.

21. Реплицированные наборы данных

При работе с кластерами Performance Secondary некоторые наборы данных поддерживаются во всех кластерах, а другие по соображениям производительности и масштабируемости хранятся в пределах одного кластера.

В следующей таблице по типам данных показано, какие наборы данных будут пересекать границы кластеров. Для типов данных, которые не пересекают границы кластеров, запросы на чтение этих данных должны быть направлены на соответствующий кластер, на котором эти данные были сгенерированы.

Данные	**Репликация между кластерами**
Эмитенты и ключи	да
Роли	да
CRL конфиг	да
URL конфиг	да
Конфиг эмитентов	да
Конфиг ключа	да
CRL	нет
Отозванные сертификаты	нет
Сертификаты leaf/эмитентов	нет

Основной эффект заключается в том, что в механизме секретов PKI листовые сертификаты, выпущенные с параметром `no_store` равным `false`, хранятся локально в кластере, который их выпустил. Это позволяет активному узлу как первичного, так и вторичного кластеров Performance Secondary выдавать сертификаты для большей масштабируемости. В результате эти сертификаты и любые их аннулирования видны только на кластере, выдавшем их. Это дополнительно означает, что каждый кластер имеет свой собственный набор CRL, отличный от других кластеров. Эти CRL должны быть либо объединены в один CRL для распространения с одного URI, либо операторы сервера должны знать, как получить все CRL со всех кластеров.

22. Масштабируемость кластера

Большинство операций, не связанных с интроспекцией, в механизме секретов PKI требуют записи в хранилище, поэтому для выполнения они передаются на активный узел кластера. В этой таблице показано, какие операции могут выполняться на резервных узлах и, таким образом, горизонтально масштабироваться по всем узлам кластера.

Путь	Операции
`ca[/pem]`	чтение
`cert/serial-number`	чтение
`cert/ca_chain`	чтение
`config/crl`	чтение
`certs`	список
`ca_chain`	чтение
`crl[/pem]`	чтение
`issue`	обновление*
`revoke/serial-number`	чтение
`sign`	обновление*
`sign-verbatim`	обновление*

*Только если для соответствующей роли установлено значение `no_store` - `true` и `generate_lease` - `false`. Если значение `generate_lease` равно `true`, создание аренды будет передано активному узлу; если значение `no_store` равно `false`, весь запрос будет передан активному узлу.

23. Поддержка PSS

В Go отсутствует поддержка PSS-сертификатов, ключей и CSR, использующих идентификатор `rsaPSS` OID (`1.2.840.113549.1.1.10`). Он требует, чтобы все RSA-сертификаты, ключи и CSR использовали альтернативный идентификатор `rsaEncryption` OID (`1.2.840.113549.1.1.1`).

При использовании OpenSSL для генерации CA или CSR из PKCS8-кодированных ключей PSS, полученные CA и CSR будут иметь идентификатор `rsaPSS`. Go и StarVault отвергнут их. Вместо этого используйте OpenSSL для генерации или преобразования в файл закрытого ключа PKCS#1v1.5 и используйте его для генерации CSR. В зависимости от роли и механизма подписания StarVault все равно будет использовать подпись PSS, несмотря на OID `rsaEncryption` в запросе, поскольку поля SubjectPublicKeyInfo и SignatureAlgorithm ортогональны. При создании внешнего CA и импорте его в StarVault убедитесь, что идентификатор `rsaEncryption` OID присутствует в поле SubjectPublicKeyInfo, даже если алгоритм подписи основан на PSS.

Эти сертификаты, сгенерированные Go (с OID `rsaEncryption`, но с подписями на основе PSS), в остальном совместимы с сертификатами, полностью основанными на PSS. OpenSSL и NSS поддерживают разбор и проверку цепочек с использованием этого типа сертификатов. Обратите внимание, что некоторые реализации TLS могут не поддерживать сертификаты этого типа, если они не поддерживают схемы подписи `rsa_pss_rsae_*`. Кроме того, некоторые реализации позволяют сертификатам `rsaPSS` OID содержать ограничения на параметры подписи, разрешенные этим сертификатом, но Go и StarVault не поддерживают добавление таких ограничений.

На данный момент в Go отсутствует поддержка подписи CSR с помощью алгоритма подписи PSS. При использовании управляемого ключа, требующего алгоритма RSA PSS (например, PKCS#11 HSM) в качестве основы для ключа промежуточного ЦС, попытка сгенерировать CSR (через `pki/intermediate/generate/kms`) не пройдет проверку подписи. В этом случае CSR нужно будет сгенерировать вне StarVault, а подписанный итоговый сертификат можно импортировать в монтировку.

В Go также отсутствует поддержка создания ответов OCSP с алгоритмом подписи PSS. StarVault автоматически понижает рейтинг эмитентов с алгоритмами подписи отзыва на основе PSS до PKCS#1v1.5, но следует учитывать, что

некоторые устройства KMS (например, HSM) могут не поддерживать этот алгоритм с тем же ключом. В результате ответчик OCSP может не подписывать ответы, возвращая внутреннюю ошибку.

24. API

Движок секретов PKI имеет полный HTTP API. Более подробную информацию см. в разделе API движка PKI secrets.

Настройка промежуточного центра сертификации

В первом руководстве по быстрому запуску сертификаты выдавались напрямую из корневого центра сертификации. Однако, как показано в предыдущем примере, такой подход не рекомендуется. В этом руководстве на основе корневого центра сертификации, созданного в предыдущем руководстве, создается промежуточный центр. Он использует корневой центр для подписи сертификата промежуточного центра.

1. Монтаж бэкэнда

Чтобы добавить еще один центр сертификации в экземпляр StarVault, необходимо смонтировать его по другому пути.

BASH

```
$ starvault secrets enable -path=pki_int pki
Successfully mounted 'pki' at 'pki_int'!
```

2. Настройка промежуточного CA

BASH

```
$ starvault secrets tune -max-lease-ttl=43800h pki_int
Successfully tuned mount 'pki_int'!
```

Это устанавливает максимальное значение TTL для секретов, выданных с монтирования, равным 5 годам. Это значение должно быть меньше или равно значению корневого центра сертификации.

Далее необходимо сгенерировать запрос на подписание промежуточного сертификата:

BASH

```
$ starvault write pki_int/intermediate/generate/internal
common_name="myvault.com Intermediate Authority" ttl=43800h
Key Value
csr -----BEGIN CERTIFICATE REQUEST-----
MIICsjCCAZoCAQAwLTErMCkGA1UEAxMibXl2YXVsdC5jb20gSW50ZXJtZWRpYXRl
IEF1dGhvcml0eTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJU1Qh8l
BW16WHAu34Fy92FnSy4219WVlKw1xwpKxjd95xH6WcxXozOs6oHFQ9c592bz51F8
KK3FFJYraUrGONI5Cz9qHbzC1mFCmjnXVXCoeNKIzEBG0Y+ehH7MQ1SvDCyvaJPX
ItFXaGf6zENiGsApw3Y3lFr0MjPzZDBH1p4Nq3aA6L2BaxvO5vczdQl5tE2ud/zs
GIdCWnl1ThDEeiX1Ppduos/dx3gaZa9ly3iCuDMKIL9yK5XTBTgKB6ALPApekLQB
kcUFbOuMzjrDSBe9ytu65yICYp26iAPPA8aKTj5cUgscgzEvQS66rSAVG/unrWxb
```

```
wbl8b7eQztCmp60CAwEAAaBAMD4GCSqGSIb3DQEJDjExMC8wLQYDVR0RBCYwJIIi
bXl2YXVsdC5jb20gSW50ZXJtZWRpYXRlIEF1dGhvcml0eTANBgkqhkiG9w0BAQsF
AAOCAQEAZA9A1QvTdAd45+Ay55FmKNWnis1zLjbmWNJURUoDei6i6SCJg0YGX1cZ
WkD0ibxPYihSsKRaIUwC2bE8cxZM57OSs7ISUmyPQAT2IHTHvuGK72qlFRBlFOzg
SHEG7gfyKdrALphyF8wM3u4gXhcnY3CdltjabL3YakZqd3Ey4870/0XXeo5c4k7w
/+n9M4xED4TnXYCGfLAlu5WWKSeCvu9mHXnJcLo1MiYjX7KGey/xYYbfxHSPm4ul
tI6Vf59zDRscfNmq37fERD3TiKP0QZNGTSRvnrxrx2RUQGXFywM8l4doG8nS5BxU
2jP20cdv0lJFvHr9663/8B/+F5L6Yw==
-----END CERTIFICATE REQUEST-----
```

Возьмите запрос на подписание от промежуточного центра и подпишите его с помощью другого центра сертификации, в данном случае корневого центра сертификации, созданного в первом примере.

BASH

```
$ starvault write pki/root/sign-intermediate csr=@pki_int.csr format=pem_bundle
ttl=43800h
Key             Value
certificate     -----BEGIN CERTIFICATE-----
MIIDZTCCAk2gAwIBAgIUENxQD7KIJi1zE/jEiYqAG1VC4NwwDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAxMLbXl2YXVsdC5jb20wHhcNMTcxMTI4MTcwNzIzWhcNMjIx
MTI3MTcwNzUzWjAtMSswKQYDVQQDEyJteXZhdWx0LmNvbSBJbnRlcm1lZGlhdGUg
QXV0aG9yaXR5MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA5seNV4Yd
uCMX0POUUuSzCBiR3Cyf9b9tGsCX7UfvZmjPs+Fl/X+Ovq6UtHM9RuTGlyfFrCWy
pflO7mc0H8PBzlvhv1WQet5aRyUOXkG6iYmooG9iobIY8z/TZCaCF605pgygfOaS
DIlwOdJkfiXxGpQ00pfIwe/Y2OK2I5e36u0E2EA6kXvcfexLjQGFPbod+H0R29Ro
/GwOJ6MpSHqB77mF025x1y08EtqT1z1kFCiDzFSkzNZEZYWljhDS6ZRY9ctzKufm
5CkUwmvCVRI2CivDJvmfhXyv0DRoq4IhYdJHo179RSObq3BY9f9LQ0balNLiM0Ft
O8f0urTqUAbySwIDAQABo4GTMIGQMA4GA1UdDwEB/wQEAwIBBjAPBgNVHRMBAf8E
BTADAQH/MB0GA1UdDgQWBBSQgTfcMrKYzyckP6t/0iVQkl0ZBDAfBgNVHSMEGDAW
gBRccsCARqs3wQDjW7JMNXS6pWlFSDAtBgNVHREEJjAkgiJteXZhdWx0LmNvbSBJ
bnRlcm1lZGlhdGUgQXV0aG9yaXR5MA0GCSqGSIb3DQEBCwUAA4IBAQABNg2HxccY
DwRpsJ+sxA0BgDyF+tYtOlXViVNv6Z+nOU0nNhQSCjfzjYWmBg25nfKaFhQSC3b7
fIW+e7it/FLVrCgaqdysoxljqhR0gXMAy8S/ubmskPWjJiKauJB5bfB59Uf2GP6j
zimZDu6WjWvvgkKcJqJEbOOS9DWBvCTdmmml1NMXZtcytpod2Y7mxninqNRx3qpx
Pst4vgAbyM/3zLSzkyUD+MXIyRXwxktFlyEYBHvMd9OoHzLO6WLxk22FyQQ+w4by
NfXJY4r5pj6a4lJ6pPuqyfBhidYMTdY3AI7w/QRGk4qQv1iDmnZspk2AxdbR5Lwe
YmChIML/f++S
-----END CERTIFICATE-----
expiration      1669568873
issuing_ca      -----BEGIN CERTIFICATE-----
MIIDNTCCAh2gAwIBAgIUdR44qhhyh3CZjnCtflGKQlTI8NswDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAxMLbXl2YXVsdC5jb20wHhcNMTcxMTI4MTYxODA2WhcNMjcx
MTI2MTYxODM1WjAWMRQwEgYDVQQDEwtteXZhdWx0LmNvbTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBANTPnQ2CUkuLrYT4V6/IIK/gWFZXFG4lWTmgM5Zh
PDquMhLEikZCbZKbupouBI8MOr5i8tycENaTnSs9dBwVEOWAHbLkliVgvCKgLi0F
PfPM87FnBoKVctO2ip8AdmYcAt/wc096dWBG6eKLVP5xsAe7NcYDtF/inHgEZ22q
ZjGVEyC6WntIASgULoHGgHakPp1AHLhGm8nL5YbusWY7RgZIlNeGWLVoneG0pxdV
7W1SPO67dsQyq58mTxMIGVUj5YE1q7/C6OhCTnAHc+sRm0oUehPfO8kY4NHpCJGv
nDRdJi6k6ewk94c0KK2tUUM/TN6ZSRfx6ccgfPH8zNcVPVcCAwEAAaN7MHkwDgYD
VR0PAQH/BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFFxywIBGqzfB
```

```
AONbskw1dLqlaUVIMB8GA1UdIwQYMBaAFFxywIBGqzfBAONbskw1dLqlaUVIMBYG
A1UdEQQPMA2CC215dmF1bHQuY29tMA0GCSqGSIb3DQEBCwUAA4IBAQBgvsgpBuVR
iKVdXXpFyoQLImuoaHZgaj5tuUDqnMoxOA1XWW6SVlZmGfDQ7+u5NBkp2cGSDRGm
ARHJTeURvdZIwdFdGkNqfAZjutRjjQOnXgS65ujZd7AnlZq1v0ZOZqVVk9YEOhOe
Rh2MjnHGNuiLBib1YNQHNuRef1mPwIE2Gm/Tz/z3JPHtkKNIKbn60zHrIIM/OT2Z
HYjcMUcqXtKGYfNjVspJm3lSDUoyJdaq80Afmy2Ez1Vt9crGG3Dj8mgs59lEhEyo
MDVhOP116M5HJfQlRPVd29qS8pFrjBvXKjJSnJNG1UFdrWBJRJ3QrBxUQALKrJlR
g5lvTeymHjS/
-----END CERTIFICATE-----
serial_number   10:dc:50:0f:b2:88:26:2d:73:13:f8:c4:89:8a:80:1b:55:42:e0:dc
```

Теперь установите сертификат подписи промежуточных центров сертификации на сертификат с корневой подписью.

BASH

```
$ starvault write pki_int/intermediate/set-signed
certificate=@signed_certificate.pem
Success! Data written to: pki_int/intermediate/set-signed
```

Теперь промежуточный центр сертификации настроен и готов к выдаче сертификатов.

3. Установка конфигурации URL

В сгенерированных сертификатах может быть закодировано местоположение CRL и местоположение выдавшего сертификат. Данные значения должны быть заданы вручную, но могут быть изменены в любое время.

BASH

```
$ starvault write pki_int/config/urls
issuing_certificates="http://127.0.0.1:8200/v1/pki_int/ca"
crl_distribution_points="http://127.0.0.1:8200/v1/pki_int/crl"
Success! Data written to: pki_int/ca/urls
```

4. Настройка роли

В сгенерированных сертификатах может быть закодировано местоположение CRL и местоположение выдавшего сертификат. Данные значения должны быть заданы вручную, но могут быть изменены в любое время.

BASH

```
$ starvault write pki_int/roles/example-dot-com \
    allowed_domains=example.com \
    allow_subdomains=true max_ttl=72h
Success! Data written to: pki_int/roles/example-dot-com
```

5. Выдача сертификатов

Записывая путь `roles/example-dot-com`, мы определяем роль `example-dot-com`. Чтобы сгенерировать новый сертификат, мы просто пишем в конечную точку `issue` с этим именем роли: теперь StarVault настроен для создания и управления сертификатами.

BASH

```
$ starvault write pki_int/issue/example-dot-com \
    common_name=blah.example.com
Key                 Value
---                 -----
certificate         -----BEGIN CERTIFICATE-----
MIIDbDCCAlSgAwIBAgIUPiAyxq+nIE6xlWf7hrzLkPQxtvMwDQYJKoZIhvcNAQEL
BQAwMzExMC8GA1UEAxMoVmF1bHQgVGVzdGluZyBJbnRlcm1lZGlhdGUgU3ViIEF1
dGhvcml0eTAeFw0xNjA5MjcwMDA5MTNaFw0xNjA5MjcwMTA5NDNaMBsxGTAXBgNV
BAMTEGJsYWguZXhhbXBsZS5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQDJAYB04IVdmSC/TimaA6BbXlvgBTZHL5wBUTmO4iHhenL0eDEXVe2Fd7Yq
75LiBJmcC96hKbqh5rwS8KwN9ElZI52/mSMC+IvoNlYHAf7shwfsjrVx3q7/bTFg
lz6wECn1ugysxynmMvgQD/pliRkxTQ7RMh4Qlh75YG3R9BHy9ZddklZp0aNaitts
0uufHnN1UER/wxBCZdWTUu34KDL9I6yE7Br0slKKHPdEsGlFcMkbZhvjslZ7DGvO
974S0qtOdKiawJZbpNPg0foGZ3AxesDUlkHmmgzUNes/sjknDYTHEfeXM6Uap0j6
XvyhCxqdeahb/Vtibg0z9I0IusJbAgMBAAGjgY8wgYwwDgYDVR0PAQH/BAQDAgOo
MB0GA1UdJQQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjAdBgNVHQ4EFgQU/5oy0rL7
TT0wX7KZK7qcXqgayNwwHwYDVR0jBBgwFoAUgM37P8oXmA972ztLfw+b1eIY5now
GwYDVR0RBBQwEoIQYmxhaC5leGFtcGxlLmNvbTANBgkqhkiG9w0BAQsFAAOCAQEA
CT2vI6/taeLTw6ZulUhLXEXYXWZu1gF8n2COjZzbZXmHxQAoZ3GtnSNwacPHAyIj
f3cA9Moo552y39LUtWk+wgFtQokWGK7LXglLaveNUBowOHq/xk0waiIinJcgTG53
Z/qnbJnTjAOG7JwVJplWUIiS1avCksrHt7heE2EGRGJALqyLZ119+PW6ogtCLUv1
X8RCTw/UkIF/LT+sLF0bXWy4Hn38Gjwj1MVv1l76cEGOVSHyrYkN+6AMnAP58L5+
IWE9tN3oac4x7jhbuNpfxazIJ8Q6l/Up5U5Evfbh6N1DI0/gFCP20fMBkHwkuLfZ
2ekZoSeCgFRDlHGkr7Vv9w==
-----END CERTIFICATE-----
issuing_ca          -----BEGIN CERTIFICATE-----
MIIDijCCAnKgAwIBAgIUB28DoGwgGFKL7fbOu9S4FalHLn0wDQYJKoZIhvcNAQEL
BQAwLzEtMCsGA1UEAxMkVmF1bHQgVGVzdGluZyBJbnRlcm1lZGlhdGUgQXV0aG9y
aXR5MB4XDTE2MDkyNzAwMDgyMVoXDTI2MDkxNjE2MDg1MVowMzExMC8GA1UEAxMo
VmF1bHQgVGVzdGluZyBJbnRlcm1lZGlhdGUgU3ViIEF1dGhvcml0eTCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAOSCiSij4wy1wiMwvZt+rtU3IaO6ZTn9
LfIPuGsR5/QSJk37pCZQco1LgoE/rTl+/xu3bDovyHDmgObghC6rzVOX2Tpi7kD+
DOZpqxOsaS8ebYgxB/XJTSxyEJuSAcpSNLqqAiZivuQXdaD0N7H3Or0awwmKE9mD
I0g8CF4fPDmuuOG0ASn9fMqXVVt5tXtEqZ9yJYfNOXx3FOPjRVOZf+kvSc31wCKe
i/KmR0AQOmToKMzq988nLqFPTi9KZB8sEU20cGFeTQFol+m3FTcIru94EPD+nLUn
xtlLELVspYb/PP3VpvRj9b+DY8FGJ5nfSJl7Rkje+CD4VxJpSadin3kCAwEAAaOB
mTCBljAOBgNVHQ8BAf8EBAMCAQYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQU
gM37P8oXmA972ztLfw+b1eIY5nowHwYDVR0jBBgwFoAUj4YAIxRwrBy0QMRKLnD0
kVidIuYwMwYDVR0RBCwwKoIoVmF1bHQgVGVzdGluZyBJbnRlcm1lZGlhdGUgU3Vi
IEF1dGhvcml0eTANBgkqhkiG9w0BAQsFAAOCAQEAA4buJuPNJvA1kiATLw1dVU2J
HPubk2Kp26Mg+GwLn7Vz45Ub133JCYfF3/zXLFZZ5Yub9gWTtjScrvNfQTAbNGdQ
BdnUlMmIRmfB7bfckhryR2R9byumeHATgNKZF7h8liNHI7X8tTzZGs6wPdXOLlzR
TlM3m1RNK8pbSPOkfPb06w9cBRlD8OAbNtJmuypXA6tYyiiMYBhP0QLAO3i4m1ns
```

```
aAjAgEjtkB1rQxW5DxoTArZ0asiIdmIcIGmsVxfDQIjFlRxAkafMs74v+5U5gbBX
wsOledU0fLl8KLq8W3OXqJwhGLK65fscrP0/omPAcFgzXf+L4VUADM4XhW6Xyg==
-----END CERTIFICATE-----
ca_chain            [-----BEGIN CERTIFICATE-----
MIIDijCCAnKgAwIBAgIUB28DoGwgGFKL7fbOu9S4FalHLn0wDQYJKoZIhvcNAQEL
BQAwLzEtMCsGA1UEAxMkVmF1bHQgVGVzdGluZyBJbnRlcm1lZGlhdGUgQXV0aG9y
aXR5MB4XDTE2MDkyNzAwMDgyMVoXDTI2MDkxNjE2MDg1MVowMzExMC8GA1UEAxMo
VmF1bHQgVGVzdGluZyBJbnRlcm1lZGlhdGUgU3ViIEF1dGhvcml0eTCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAOSCiSij4wy1wiMwvZt+rtU3IaO6ZTn9
LfIPuGsR5/QSJk37pCZQco1LgoE/rTl+/xu3bDovyHDmgObghC6rzVOX2Tpi7kD+
DOZpqxOsaS8ebYgxB/XJTSxyEJuSAcpSNLqqAiZivuQXdaD0N7H3Or0awwmKE9mD
I0g8CF4fPDmuuOG0ASn9fMqXVVt5tXtEqZ9yJYfNOXx3FOPjRVOZf+kvSc31wCKe
i/KmR0AQOmToKMzq988nLqFPTi9KZB8sEU20cGFeTQFol+m3FTcIru94EPD+nLUn
xtlLELVspYb/PP3VpvRj9b+DY8FGJ5nfSJl7Rkje+CD4VxJpSadin3kCAwEAAaOB
mTCBljAOBgNVHQ8BAf8EBAMCAQYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQU
gM37P8oXmA972ztLfw+b1eIY5nowHwYDVR0jBBgwFoAUj4YAIxRwrBy0QMRKLnD0
kVidIuYwMwYDVR0RBCwwKoIoVmF1bHQgVGVzdGluZyBJbnRlcm1lZGlhdGUgU3Vi
IEF1dGhvcml0eTANBgkqhkiG9w0BAQsFAAOCAQEAA4buJuPNJvA1kiATLw1dVU2J
HPubk2Kp26Mg+GwLn7Vz45Ub133JCYfF3/zXLFZZ5Yub9gWTtjScrvNfQTAbNGdQ
BdnUlMmIRmfB7bfckhryR2R9byumeHATgNKZF7h8liNHI7X8tTzZGs6wPdXOLlzR
TlM3m1RNK8pbSPOkfPb06w9cBRlD8OAbNtJmuypXA6tYyiiMYBhP0QLAO3i4m1ns
aAjAgEjtkB1rQxW5DxoTArZ0asiIdmIcIGmsVxfDQIjFlRxAkafMs74v+5U5gbBX
wsOledU0fLl8KLq8W3OXqJwhGLK65fscrP0/omPAcFgzXf+L4VUADM4XhW6Xyg==
-----END CERTIFICATE-----]
private_key         -----BEGIN RSA PRIVATE KEY-----
MIIEpgIBAAKCAQEAyQGAdOCFXZkgv04pmgOgW15b4AU2Ry+cAVE5juIh4Xpy9Hgx
F1XthXe2Ku+S4gSZnAveoSm6oea8EvCsDfRJWSOdv5kjAviL6DZWBwH+7IcH7I61
cd6u/20xYJc+sBAp9boMrMcp5jL4EA/6ZYkZMU0O0TIeEJYe+WBt0fQR8vWXXZJW
adGjWorbbNLrnx5zdVBEf8MQQmXVk1Lt+Cgy/SOshOwa9LJSihz3RLBpRXDJG2Yb
47JWewxrzve+EtKrTnSomsCWW6TT4NH6BmdwMXrA1JZB5poM1DXrP7I5Jw2ExxH3
lzOlGqdI+l78oQsanXmoW/1bYm4NM/SNCLrCWwIDAQABAoIBAQCCbHMJY1Wl8eIJ
v5HG2WuHXaaHqVoavo2fXTDXwWryfx1v+zz/Q0YnQBH3shPAi/OQCTOfpw/uVWTb
dUZul3+wUyfcVmUdXGCLgBY53dWna8Z8e+zHwhISsqtDXV/TpelUBDCNO324XIIR
Cg0TLO4nyzQ+ESLo6D+Y2DTp8lBjMEkmKTd8CLXR2ycEoVykN98qPZm8keiLGO91
I8K7aRd8uOyQ6HUfJRlzFHSuwaLReErxGTEPI4t/wVqh2nP2gGBsn3apiJ0ul6Jz
NlYO5PqiwpeDk4ibhQBpicnm1jnEcynH/WtGuKgMNB0M4SBRBsEguO7WoKx3o+qZ
iVIaPWDhAoGBAO05UBvyJpAcz/ZNQlaF0EAOhoxNQ3h6+6ZYUE52PgZ/DHftyJPI
Y+JJNclY91wn91Yk3ROrDi8gqhzA+2Lelxo1kuZDu+m+bpzhVUdJia7tZDNzRIhI
24eP2GdochooOZ0qjvrik4kuX43amBhQ4RHsBjmX5CnUlL5ZULs8v2xnAoGBANjq
VLAwiIIqJZEC6BuBvVYKaRWkBCAXvQ3j/OqxHRYu3P68PZ58Q7HrhrCuyQHTph2v
fzfmEMPbSCrFIrrMRmjUG8wopL7GjZjFl8HOBHFwzFiz+CT5DEC+IJIRkp4HM8F/
PAzjB2wCdRdSjLTD5ph0/xQIg5xfln7D+wqU0QHtAoGBAKkLF0/ivaIiNftw0J3x
WxXag/yErlizYpIGCqvuzII6lLr9YdoViT/eJYrmb9Zm0HS9biCu2zuwDijRSBIL
RieyF40opUaKoi3+0JMtDwTtO2MCd8qaCH3QfkgqAG0tTuj1Q8/6F2JA/myKYamq
MMhhpYny9+7rAlemM8ZJIqtvAoGBAKOI3zpKDNCdd98A4v7B7H2usZUIJ7gOTZDo
XqiNyRENWb2PK6GNq/e6SrxvuclvyKA+zFnXULJoYtsj7tAH69lieGaOCc5uoRgZ
eBU7/euMj/McE6vEO3GgJawaJYCQi3uJMjvA+bp7i81+hehOfU5ZfmmbFaZSBoMh
u+U5Vu3tAoGBANnBIbHfD3E7rqnqdpH1oRRHLA1VdghzEKgyUTPHNDzPJG87RY3c
rRqeXepblud3qFjD60xS9BzcBijOvZ4+KHk6VIMpkyqoeNVFCJbBVCw+JGMp88+v
e9t+2iwryh5+rnq+pg6anmgwHldptJc1XEFZA2UUQ89RP7kOGQF6IkIS
-----END RSA PRIVATE KEY-----
```

```
private_key_type    rsa
serial_number       3e:20:32:c6:af:a7:20:4e:b1:95:67:fb:86:bc:cb:90:f4:31:b6:f3
```

Теперь StarVault сгенерировал новый набор учетных данных, используя конфигурацию роли `example-dot-com`. Здесь мы видим закрытый ключ и сертификат, сгенерированные динамически. Также возвращаются сертификат CA и цепочка доверия CA. Цепочка CA Chain возвращает все промежуточные центры в цепочке доверия. Корневой центр не включен, поскольку он считается доверенным на уровне ОС.

6. API

Движок секретов PKI имеет полноценный HTTP API. Более подробную информацию можно найти в разделе API движка PKI secrets.