Настройка корневого центра сертификации

В этом документе приводится краткий обзор настройки StarVault PKI Secrets Engine с сертификатом корневого центра сертификации.

1. Монтаж бэкэнда

Первым шагом к использованию бэкэнда PKI является его монтирование. В отличие от бэкенда `kv`, бэкенд `pki` не монтируется по умолчанию.

```bash
$ starvault secrets enable pki
Successfully mounted 'pki' at 'pki'!
```

2. Настройка сертификата CA

Далее в StarVault необходимо настроить сертификат CA и связанный с ним закрытый ключ. Мы воспользуемся поддержкой генерации самоподписанных корневых сертификатов, но StarVault также поддерживает генерацию промежуточного CA (с CSR для подписи) или установку связки сертификата и закрытого ключа в PEM-кодировке непосредственно в бэкенд.

Обычно корневой сертификат используется только для подписи промежуточных сертификатов CA, но в данном примере мы будем действовать так, как будто вы будете выпускать сертификаты непосредственно из корневого сертификата. Так как это корневой сертификат, мы хотим установить длительное время жизни сертификата; поскольку он соответствует максимальному TTL монтирования, сначала мы настроим его:

```bash
$ starvault secrets tune -max-lease-ttl=87600h pki
Successfully tuned mount 'pki'!
```

Это устанавливает максимальный TTL для секретов, выданных с монтирования, равным 10 годам. (Обратите внимание, что роли могут дополнительно ограничивать максимальный TTL.)

Теперь мы создадим наш корневой сертификат:

```
BASH
$ starvault write pki/root/generate/internal common_name=myvault.com ttl=87600h
Key             Value
---             -----
certificate     -----BEGIN CERTIFICATE-----
MIIDNTCCAh2gAwIBAgIUJqrw/9EDZbp4DExaLjh0vSAHyBgwDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAxMLbXl2YXVsdC5jb20wHhcNMTcxMjA4MTkyMzIwWhcNMjcx
MjA2MTkyMzQ5WjAWMRQwEgYDVQQDEwtteXZhdWx0LmNvbTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAKY/vJ6sRFym+yFYUneoVtDmOCaDKAQiGzQw0IXL
wgMBBb82iKpYj5aQjXZGIl+VkVnCi+M2AQ/iYXWZf1kTAdle4A6OC4+VefSIa2b4
eB7R8aiGTce62jB95+s5/YgrfIqk6igfpCSXYLE8ubNDA2/+cqvjhku1UzlvKBX2
hIlgWkKlrsnybHN+B/3Usw9Km/87rzoDR3OMxLV55YPHiq6+olIfSSwKAPjH8LZm
uM1ITLG3WQUl8ARF17Dj+wOKqbUG38PduVwKL5+qPksrvNwlmCP7Kmjncc6xnYp6
5lfr7V4DC/UezrJYCIb0g/SvtxoN1OuqmmvSTKiEE7hVOAcCAwEAAaN7MHkwDgYD
VR0PAQH/BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFECKdYM4gDbM
kxRZA2wR4f/yNhQUMB8GA1UdIwQYMBaAFECKdYM4gDbMkxRZA2wR4f/yNhQUMBYG
A1UdEQQPMA2CC215dmF1bHQuY29tMA0GCSqGSIb3DQEBCwUAA4IBAQCCJKZPcjjn
7mvD2+sr6lx4DW/vJwVSW8eTuLtOLNu6/aFhcgTY/OOB8q4n6iHuLrEt8/RV7RJI
obRx74SfK9BcOLt4+DHGnFXqu2FNVnhDMOKarj41yGyXlJaQRUPYf6WJJLF+ZphN
nNsZqHJHBfZtpJpE5Vywx3pah08B5yZHk1ItRPEz7EY3uwBI/CJoBb+P5Ahk6krc
LZ62kFwstkVuFp43o3K7cRNexCIsZGx2tsyZ0nyqDUFsBr66xwUfn3C+/1CDc9YL
zjq+8nI2ooIrj4ZKZCOm2fKd1KeGN/CZD7Ob6uNhXrd0Tjwv00a7nffvYQkl/1V5
BT55jevSPVVu
-----END CERTIFICATE-----
expiration      1828121029
issuing_ca      -----BEGIN CERTIFICATE-----
MIIDNTCCAh2gAwIBAgIUJqrw/9EDZbp4DExaLjh0vSAHyBgwDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAxMLbXl2YXVsdC5jb20wHhcNMTcxMjA4MTkyMzIwWhcNMjcx
MjA2MTkyMzQ5WjAWMRQwEgYDVQQDEwtteXZhdWx0LmNvbTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAKY/vJ6sRFym+yFYUneoVtDmOCaDKAQiGzQw0IXL
wgMBBb82iKpYj5aQjXZGIl+VkVnCi+M2AQ/iYXWZf1kTAdle4A6OC4+VefSIa2b4
eB7R8aiGTce62jB95+s5/YgrfIqk6igfpCSXYLE8ubNDA2/+cqvjhku1UzlvKBX2
hIlgWkKlrsnybHN+B/3Usw9Km/87rzoDR3OMxLV55YPHiq6+olIfSSwKAPjH8LZm
uM1ITLG3WQUl8ARF17Dj+wOKqbUG38PduVwKL5+qPksrvNwlmCP7Kmjncc6xnYp6
5lfr7V4DC/UezrJYCIb0g/SvtxoN1OuqmmvSTKiEE7hVOAcCAwEAAaN7MHkwDgYD
VR0PAQH/BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFECKdYM4gDbM
kxRZA2wR4f/yNhQUMB8GA1UdIwQYMBaAFECKdYM4gDbMkxRZA2wR4f/yNhQUMBYG
A1UdEQQPMA2CC215dmF1bHQuY29tMA0GCSqGSIb3DQEBCwUAA4IBAQCCJKZPcjjn
7mvD2+sr6lx4DW/vJwVSW8eTuLtOLNu6/aFhcgTY/OOB8q4n6iHuLrEt8/RV7RJI
obRx74SfK9BcOLt4+DHGnFXqu2FNVnhDMOKarj41yGyXlJaQRUPYf6WJJLF+ZphN
nNsZqHJHBfZtpJpE5Vywx3pah08B5yZHk1ItRPEz7EY3uwBI/CJoBb+P5Ahk6krc
LZ62kFwstkVuFp43o3K7cRNexCIsZGx2tsyZ0nyqDUFsBr66xwUfn3C+/1CDc9YL
zjq+8nI2ooIrj4ZKZCOm2fKd1KeGN/CZD7Ob6uNhXrd0Tjwv00a7nffvYQkl/1V5
BT55jevSPVVu
-----END CERTIFICATE-----
serial_number   26:aa:f0:ff:d1:03:65:ba:78:0c:4c:5a:2e:38:74:bd:20:07:c8:18
```

Возвращаемый сертификат носит чисто информационный характер; он и его закрытый ключ надежно хранятся во внутреннем хранилище.

3. Установка конфигурации URL

В сгенерированных сертификатах может быть закодировано местоположение CRL и местоположение выдавшего сертификат. Эти значения должны быть заданы вручную и обычно соответствуют FQDN, связанному с сервером StarVault, но могут быть изменены в любое время.

BASH

```
$ starvault write pki/config/urls
issuing_certificates="http://starvault.example.com:8200/v1/pki/ca"
crl_distribution_points="http://starvault.example.com:8200/v1/pki/crl"
Success! Data written to: pki/ca/urls
```

4. Настройка роли

Следующим шагом будет настройка роли. Роль - это логическое имя, которое сопоставляется с политикой, используемой для генерации учетных данных. Например, давайте создадим роль "example-dot-com":

BASH

```
$ starvault write pki/roles/example-dot-com \
    allowed_domains=example.com \
    allow_subdomains=true max_ttl=72h
Success! Data written to: pki/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 на создание и управление сертификатами!

BASH

```
$ starvault write pki/issue/example-dot-com \
    common_name=blah.example.com
Key                 Value
---                 -----
certificate         -----BEGIN CERTIFICATE-----
MIIDvzCCAqegAwIBAgIUWQuvpMpA2ym36EoiYyf3Os5UeIowDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAxMLbXl2YXVsdC5jb20wHhcNMTcxMjA4MTkyNDA1WhcNMTcx
MjExMTkyNDM1WjAbMRkwFwYDVQQDExBibGFoLmV4YW1wbGUuY29tMIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1CU93lVgcLXGPxRGTRT3GM5wqytCo7Z6
gjfoHyKoPCAqjRdjsYgp1FMvumNQKjUat5KTtr2fypbOnAURDCh4bN/omcj7eAqt
ldJ8mf8CtKUaaJ1kp3R6RRFY/u96BnmKUG8G7oDeEDsKlXuEuRcNbGlGF8DaM/O1
HFa57cM/8yFB26Nj5wBoG5Om6ee5+W+14Qee8AB6OJbsf883Z+zvhJTaB0QM4ZUq
uAMoMVEutWhdI5EFm5OjtMeMu2U+iJl2XqqgQ/JmLRjRdMn1qd9TzTaVSnjoZ97s
jHK444Px1m45einLqKUJ+Ia2ljXYkkItJj9Ut6ZSAP9fHlAtX84W3QIDAQABo4H/
```

```
MIH8MA4GA1UdDwEB/wQEAwIDqDAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUH
AwIwHQYDVR0OBBYEFH/YdObW6T94U0zuU5hBfTfU5pt1MB8GA1UdIwQYMBaAFECK
dYM4gDbMkxRZA2wR4f/yNhQUMDsGCCsGAQUFBwEBBC8wLTArBggrBgEFBQcwAoYf
aHR0cDovLzEyNy4wLjAuMTo4MjAwL3YxL3BraS9jYTAbBgNVHREEFDASghBibGFo
LmV4YW1wbGUuY29tMDEGA1UdHwQqMCgwJqAkoCKGIGh0dHA6Ly8xMjcuMC4wLjE6
ODIwMC92MS9wa2kvY3JsMA0GCSqGSIb3DQEBCwUAA4IBAQCDXbHV68VayweB2tkb
KDdCaveaTULjCeJUnm9UT/6C0YqC/RxTAjdKFrilK49elOA3rAtEL6dmsDP2yH25
ptqi2iU+y99HhZgu0zkS/p8elYN3+l+0O7pOxayYXBkFf5t0TlEWSTb7cW+Etz/c
MvSqx6vVvspSjB0PsA3eBq0caZnUJv2u/TEiUe7PPY0UmrZxp/R/P/kE54yI3nWN
4Cwto6yUwScOPbVR1d3hE2KU2toiVkEoOk17UyXWTokbG8rG0KLj99zu7my+Fyre
sjV5nWGDSMZODEsGxHOC+JgNAC1z3n14/InFNOsHICnA5AnJzQdSQQjvcZHN2NyW
+t4f
-----END CERTIFICATE-----
issuing_ca          -----BEGIN CERTIFICATE-----
MIIDNTCCAh2gAwIBAgIUJqrw/9EDZbp4DExaLjh0vSAHyBgwDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAxMLbXl2YXVsdC5jb20wHhcNMTcxMjA4MTkyMzIwWhcNMjcx
MjA2MTkyMzQ5WjAWMRQwEgYDVQQDEwtteXZhdWx0LmNvbTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAKY/vJ6sRFym+yFYUneoVtDmOCaDKAQiGzQw0IXL
wgMBBb82iKpYj5aQjXZGIl+VkVnCi+M2AQ/iYXWZf1kTAdle4A6OC4+VefSIa2b4
eB7R8aiGTce62jB95+s5/YgrfIqk6igfpCSXYLE8ubNDA2/+cqvjhku1UzlvKBX2
hIlgWkKlrsnybHN+B/3Usw9Km/87rzoDR3OMxLV55YPHiq6+olIfSSwKAPjH8LZm
uM1ITLG3WQUl8ARF17Dj+wOKqbUG38PduVwKL5+qPksrvNwlmCP7Kmjncc6xnYp6
5lfr7V4DC/UezrJYCIb0g/SvtxoN1OuqmmvSTKiEE7hVOAcCAwEAAaN7MHkwDgYD
VR0PAQH/BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFECKdYM4gDbM
kxRZA2wR4f/yNhQUMB8GA1UdIwQYMBaAFECKdYM4gDbMkxRZA2wR4f/yNhQUMBYG
A1UdEQQPMA2CC215dmF1bHQuY29tMA0GCSqGSIb3DQEBCwUAA4IBAQCCJKZPcjjn
7mvD2+sr6lx4DW/vJwVSW8eTuLtOLNu6/aFhcgTY/OOB8q4n6iHuLrEt8/RV7RJI
obRx74SfK9BcOLt4+DHGnFXqu2FNVnhDMOKarj41yGyXlJaQRUPYf6WJJLF+ZphN
nNsZqHJHBfZtpJpE5Vywx3pah08B5yZHk1ItRPEz7EY3uwBI/CJoBb+P5Ahk6krc
LZ62kFwstkVuFp43o3K7cRNexCIsZGx2tsyZ0nyqDUFsBr66xwUfn3C+/1CDc9YL
zjq+8nI2ooIrj4ZKZCOm2fKd1KeGN/CZD7Ob6uNhXrd0Tjwv00a7nffvYQkl/1V5
BT55jevSPVVu
-----END CERTIFICATE-----
private_key         -----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA1CU93lVgcLXGPxRGTRT3GM5wqytCo7Z6gjfoHyKoPCAqjRdj
sYgp1FMvumNQKjUat5KTtr2fypbOnAURDCh4bN/omcj7eAqtldJ8mf8CtKUaaJ1k
p3R6RRFY/u96BnmKUG8G7oDeEDsKlXuEuRcNbGlGF8DaM/O1HFa57cM/8yFB26Nj
5wBoG5Om6ee5+W+14Qee8AB6OJbsf883Z+zvhJTaB0QM4ZUquAMoMVEutWhdI5EF
m5OjtMeMu2U+iJl2XqqgQ/JmLRjRdMn1qd9TzTaVSnjoZ97sjHK444Px1m45einL
qKUJ+Ia2ljXYkkItJj9Ut6ZSAP9fHlAtX84W3QIDAQABAoIBAQCf5YIANfF+gkNt
/+YM6yRi+hZJrU2I/1zPETxPW1vaFZR8y4hEoxCEDD8JCRm+9k+w1TWoorvxgkEv
r1HuDALYbNtwLd/71nCHYCKyH1b2uQpyl07qOAyASlb9r5oVjz4E6eobkd3N9fJA
QN0EdK+VarN968mLJsD3Hxb8chGdObBCQ+LO+zdqQLaz+JwhfnK98rm6huQtYK3w
ccd0OwoVmtZz2eJl11TJkB9fi4WqJyxl4wST7QC80LstB1deR78oDmN5WUKU12+G
4Mrgc1hRwUSm18HTTgAhaA4A3rjPyirBohb5Sf+jJxusnnay7tvWeMnIiRI9mqCE
dr3tLrcxAoGBAPL+jHVUF6sxBqm6RTe8Ewg/8RrGmd69oB71QlVUrLYyC96E2s56
19dcyt5U2z+F0u9wlwR1rMb2BJIXbxlNk+i87IHmpOjCMS38SPZYWLHKj02eGfvA
MjKKqEjNY/md9eVAVZIWSEy63c4UcBK1qUH3/5PNlyjk53gCOI/4OXX/AoGBAN+A
Alyd6A/pyHWq8WMyAlV18LnzX8XktJ07xrNmjbPGD5sEHp+Q9V33NitOZpu3bQL+
gCNmcrodrbr9LBV83bkAOVJrf82SPaBesV+ATY7ZiWpqvHTmcoS7nglM2XTr+uWR
Y9JGdpCE9U5QwTc6qfcn7Eqj7yNvvHMrT+1SHwsjAoGBALQyQEbhzYuOF7rV/26N
```

```
ci+z+0A39vNO++b5Se+tk0apZlPlgb2NK3LxxR+LHevFed9GRzdvbGk/F7Se3CyP
cxgswdazC6fwGjhX1mOYsG1oIU0V6X7f0FnaqWETrwf1M9yGEO78xzDfgozIazP0
s0fQeR9KXsZcuaotO3TIRxRRAoGAMFIDsLRvDKm1rkL0B0czm/hwwDMu/KDyr5/R
2M2OS1TB4PjmCgeUFOmyq3A63OWuStxtJboribOK8Qd1dXvWj/3NZtVY/z/j1P1E
Ceq6We0MOZa0Ae4kyi+p/kbAKPgv+VwSoc6cKailRHZPH7quLoJSIt0IgbfRnXC6
ygtcLNMCgYBwiPw2mTYvXDrAcO17NhK/r7IL7BEdFdx/w8vNJQp+Ub4OO3Iw6ARI
vXxu6A+Qp50jra3UUtnI+hIirMS+XEeWqJghK1js3ZR6wA/ZkYZw5X1RYuPexb/4
6befxmnEuGSbsgvGqYYTf5Z0vgsw4tAHfNS7TqSulYH06CjeG1F8DQ==
-----END RSA PRIVATE KEY-----
private_key_type    rsa
serial_number       59:0b:af:a4:ca:40:db:29:b7:e8:4a:22:63:27:f7:3a:ce:54:78:8a
```

Теперь StarVault сгенерировал новый набор учетных данных с использованием конфигурации роли `example-dot-com`. Здесь мы видим закрытый ключ и сертификат, сгенерированные динамически.

Используя ACL, можно ограничить использование бэкенда `pki` таким образом, чтобы доверенные операторы могли управлять определениями ролей, а пользователи и приложения были ограничены в полномочиях, которые им разрешено читать.

Если возникли сложности, просто запустите `starvault path-help pki` или с подпутем для получения интерактивной справки.

6. API

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

2025 orionsoft. Все права защищены.

Примитивы ротации

Механизм секретов PKI StarVault поддерживает несколько эмитентов в одной точке монтирования. Используя приведенные ниже типы сертификатов, можно осуществлять ротацию в различных ситуациях с участием как root, так и промежуточных CA, управляемых StarVault.

1. Поля сертификата X.509

X.509 — сложная спецификация; современные реализации обычно ссылаются на RFC 5280 для получения подробной информации. Для проверки сертификатов как RFC 5280, так и RFC 6125 по проверке TLS важны для понимания того, как добиться ротации.

Ниже приводится упрощенное описание этих стандартов для целей данного документа.

Каждый X.509-сертификат начинается с генерации асимметричной пары ключей с использованием RSA или ECDSA. Эта пара служит основой для создания запроса на подписание сертификата (CSR), в котором указываются желаемые поля будущего сертификата. Однако удостоверяющий центр (CA) сам решает, какие из них включить, а какие отклонить. В CSR также входит открытый ключ, подписанный соответствующим закрытым ключом, что подтверждает владение ключевой парой. Обычно заявитель указывает желаемые значения в поле Subject и расширении Subject Alternative Name, но окончательное решение остаётся за CA. После одобрения запроса, удостоверяющий центр формирует сертификат: в поле Issuer указывает себя как издателя, присваивает сертификату уникальный серийный номер и подписывает его своим закрытым ключом. Если речь идёт о корневом (self-signed) сертификате, CA может использовать для подписи ту же пару ключей, что и в CSR.

Далее указано несколько важных ограничений:

- Один сертификат может иметь только одного эмитента, но этот эмитент идентифицируется по субъекту на выдающем сертификате и его открытому ключу.
- Одна пара ключей может использоваться для нескольких сертификатов, но у одного сертификата может быть только один опорный ключевой материал.

Следующие поля итогового сертификата имеют отношение к ротации:

- Материалы подтверждающие открытый и закрытый ключ (Subject Public Key Info).
    - Закрытый ключ не включён в сам сертификат, но однозначно определяется по открытому ключу.

- Поле Subject
    - Указывает, какому субъекту выдан сертификат. Хотя значения в SAN (Subject Alternative Name) полезны при проверке TLS-сертификатов серверов по hostname и URI, они, как правило, не играют роли при проверке цепочки сертификатов или при ротации.
- Период действия сертификата (Validity)
    - RFC 5280 не предъявляет никаких требований к сроку действия выданного сертификата по отношению к сроку действия сертификата-эмитента. Однако в нем говорится, что сертификаты должны быть отозваны, если их статус не может быть сохранен до даты NotAfter. Именно поэтому конечная точка конфигурации `/pki/issuer/:issuer_ref` в StarVault поддерживает поведение `leaf_not_after_behavior` для каждого эмитента, а не для каждой роли.
    - Некоторые браузеры полностью доверяют сертификатам, хранящимся в их хранилищах доверия, даже если срок действия этих сертификатов истек.
        - Обратите внимание, что это относится только к сертификатам в доверительном хранилище; для сертификатов, не входящих в хранилище (например, промежуточных), сроки действия по-прежнему будут соблюдаться.
- **Issuer** и **signatureValue** сертификата
    - В поле **Issuer** выданного сертификата выдающий сертификат помещает свое собственное значение Subject. Это позволяет впоследствии идентифицировать эмитента (без необходимости проверять подпись на всех известных локальных сертификатах) при проверке представленного сертификата и цепочки.
    - Подпись под всем сертификатом (закрытым ключом эмитента) помещается в поле **signatureValue**.
- Необязательное поле **Authority Key Identifier**
    - Это поле может содержать одно или оба значения:
        - Хеш открытого ключа эмитента. Это расширение установлено, и это значение заполняется StarVault.
        - Тема и серийный номер эмитента. Это значение не задается StarVault.
    - Последнее ограничение опасно для ротации: оно препятствует перекрестному подписанию и перевыпуску, так как новые сертификаты (хотя и имеют тот же материал ключа) будут иметь разные серийные номера. Подробнее об этом ограничении см. в разделе "Ограничения примитивов" ниже.
- Серийный номер сертификата
    - Это поле уникально для конкретного эмитента; когда сертификат перевыпускается его удостоверяющим центром, он всегда будет иметь другое поле серийного номера.
- Поле точки распространения CRL

- Это поле, в котором указывается, где ожидается существование CRL для данного сертификата и какими эмитентами CRL (по умолчанию это сам выдавший сертификат) он должен быть подписан. Это в основном информационное поле, и для серверного программного обеспечения, такого как nginx, метод Cert Auth от StarVault и Apache. CRL предоставляются серверу и не заставляют сервер автоматически получать CRL для сертификатов.
- root сертификаты (в хранилищах доверия браузеров) обычно не считаются отзываемыми. Однако если промежуточный сертификат будет отозван последовательно, он появится в CRL родительского сертификата и может помешать ротации.

2. Примитивы ротации X.509

Ротация (с точки зрения организации) может безопасно происходить только при наличии определенных промежуточных сертификатов X.509. Чтобы различать два типа сертификатов, используемых для достижения ротации, в этом документе они обозначены как *примитивы*.

Ротация сертификата конечной сущности тривиальна с точки зрения цепочки доверия X.509; этот процесс происходит каждый день и должен зависеть только от того, что находится в хранилище доверия, а не от самого сертификата конечной сущности. В StarVault запросчик обращается к различным конечным точкам выдачи (`/pki/issue/:name` или `/pki/sign/:name` - или использует небезопасную `/pki/sign-verbatim`), меняет старый сертификат на новый и перезагружает конфигурацию или перезапускает службу. Другие части организации могут использовать ACME для выпуска и ротации сертификатов, особенно если служба является публичной (и поэтому должна быть выпущена публичным центром сертификации). Учитывая, что сертификат подписан доверенным корнем, любые устройства, подключающиеся к службе, не заметят разницы.

Ротация промежуточных сертификатов почти так же проста. Если предположить, что все работает нормально (когда при выпуске конечного сертификата полная цепочка сертификатов обновляется в конфигурации службы), это должно быть так же просто, как создать новый промежуточный CA, подписать его на корневом CA, а затем начать выпуск на новом промежуточном сертификате. В StarVault, если промежуточный генерируется в существующем пути монтирования (или перемещается в него), запрашивающая организация не должна сильно беспокоиться. В ACME компания Let’s Encrypt успешно ротировала промежуточные сертификаты, чтобы представить цепочку с перекрестной подписью (для старых устройств Android). Если предположить, что родительский(ие) сертификат(ы) старого посредника все еще действителен(ы) и ему(им) доверяют, сертификаты, выпущенные под старыми посредниками, должны продолжать подтверждаться.

Самая сложная часть ротации, требующая использования этих примитивов - ротация корневых сертификатов. Они хранятся в хранилище доверия каждого устройства, и их трудно обновлять с точки зрения операционной деятельности всей организации. Если организация не может менять корневые сертификаты практически мгновенно и одновременно (например, с помощью агента), не пропуская ни одного устройства, этот процесс, скорее всего, растянется на месяцы.

Чтобы сделать этот процесс менее рискованным, существуют различные примитивные типы сертификатов, которые используют вышеуказанные поля сертификата. Ключом к их успеху является следующее примечание:

Хотя сертификаты добавляются в хранилище доверия, в конечном итоге доверие определяется связанным с ними ключевым материалом: два сертификата эмитента с одинаковым субъектом, но разными открытыми ключами не могут подтвердить один и тот же сертификат листа; только если ключи одинаковы, это может произойти.

2.1. Примитив с перекрестной подписью

Это наиболее распространенный тип примитива ротации. Общий CSR подписывается двумя CA, в результате чего создаются два сертификата. Эти сертификаты должны иметь одинаковый субъект (но могут иметь разных эмитентов и разные серийные номера) и одинаковый материал базового ключа, чтобы подписанным ими сертификатам можно было доверять в любом варианте.

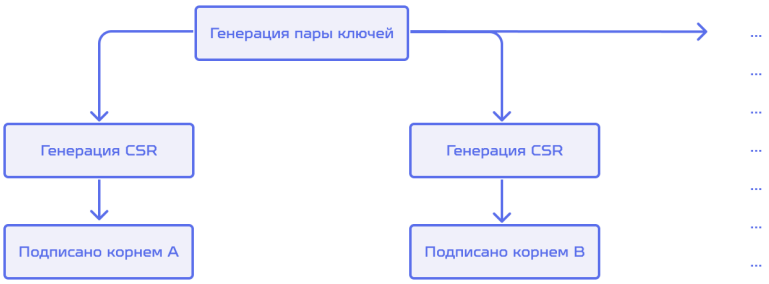
Из-за ограничений в использовании и проверке сертификатов конечных субъектов (сервисы и библиотеки проверки ожидают только один), перекрестная подпись чаще всего применяется только к промежуточным.

2.1.1. Заметка о корнях с перекрестной подписью

Технически перекрестное подписание может происходить между двумя корнями, позволяя пучкам доверия с одним из корней подтверждать сертификаты, выданные через другой. Однако этот процесс создает сертификат, который фактически является промежуточным (поскольку он больше не является самоподписанным) и обычно должен подаваться рядом с цепочкой доверия. Учитывая это ограничение, предпочтительнее перекрестно подписывать промежуточные сертификаты верхнего уровня под корнем, за исключением случаев, когда старый корневой сертификат был использован для прямого выпуска сертификатов листьев.

Таким образом, остальная часть этого процесса предполагает, что промежуточный сертификат подписывается перекрестной подписью, так как это более распространено.

2.1.2. Технологический процесс



Здесь пара ключей была сгенерирована в определенный момент времени. Создаются два CSR и отправляются в два разных корневых центра (Root A и Root B). В результате получаются два отдельных сертификата (потенциально с разными сроками действия) с одним и тем же субъектом и одним и тем же ключевым материалом подложки.

Перекрестное подписание не обязательно должно происходить одновременно; между первым и вторым сертификатом может быть разрыв в несколько лет. Кроме того, нет ограничений на количество перекрестно подписанных "дубликатов" (используется с тем же субъектом и ключом) сертификатов: они могут быть перекрестно подписаны многими различными корневыми сертификатами.

2.1.3. Иерархия сертификатов

Вышеописанный процесс приводит к созданию двух путей доверия: любой из root A или root B (или оба) может существовать в хранилищах доверия клиента, и листовой сертификат будет подтвержден правильно. Поскольку для обоих промежуточных сертификатов (C и D) используется один и тот же ключевой материал, поле подписи выданного листового сертификата будет одинаковым вне зависимости от того, к какому из промежуточных сертификатов было произведено обращение.

Таким образом, перекрестное подписание является объединяющим примитивом; два отдельных пути доверия теперь объединяются в один, поскольку поле эмитента сертификата листа указывает на два отдельных пути (через дублирование сертификата в цепочке) и будет условно проверяться на основе того, какой корень присутствует в хранилище доверия.

Эта конструкция документирована и используется в нескольких местах:

- https://letsencrypt.org/certificates/
- https://scotthelme.co.uk/cross-signing-alternate-trust-paths-how-they-work/
- https://security.stackexchange.com/questions/14043/what-is-the-use-of-cross-signing-certificates-in-x-509

2.1.4. Исполнение в StarVault

Для создания сертификата с перекрестной подписью в StarVault используйте конечную точку `/intermediate/cross-sign`. Здесь при создании перекрестной подписи для всех сертификатов B, которые должны быть подтверждены сертификатом A, укажите значения (`key_ref`, все части Subject и т. д.) для сертификата B во время промежуточной генерации. Затем подпишите этот CSR (используя конечную точку `/issuer/:issuer_ref/sign-intermediate`) ссылкой на `cert A` и предоставьте необходимые значения из `cert B` (например, части Subject). `cert A` может находиться вне StarVault. Наконец, импортируйте кросс-подписанный сертификат в StarVault с помощью конечной точки `/issuers/import/cert`.

Если этот процесс прошел успешно, и оба сертификата A и B и их ключевой материал находятся в StarVault, только что импортированный кросс-подписанный сертификат будет иметь поле ответа `ca_chain` при чтении, содержащее сертификат A, а `ca_chain` сертификата B будет содержать кросс-подписанный сертификат и его значение `ca_chain`.

Независимо от типа эмитента, важно предоставить все соответствующие параметры в том виде, в котором они были изначально; StarVault не выводит, например, параметры имени субъекта из существующего эмитента; он просто повторно использует тот же самый ключевой материал.

2.1.5. Записи в `manual_chain`

Если промежуточный элемент имеет перекрестную подпись и импортирован в то же хранилище, что и его пара, StarVault не обнаружит пары с перекрестной подписью при автоматическом построении цепочки. В результате при выдаче листа будет создана цепочка, включающая только одну из этих пар цепочек. Это связано с тем, что параметр `ca_chain` для выдачи листа копирует значение непосредственно у подписывающего эмитента, а не вычисляет свою собственную копию цепочки.

Чтобы исправить это, обновите поле `manual_chain` у эмитентов, включив в него цепочки обеих пар. Например, если даны `intA`, подписанный `rootA`, и `intB`, подписанный `rootB`, как его кросс-подписанная версия, можно сделать следующее:

```bash
$ starvault patch pki/issuer/intA manual_chain=self,rootA,intB,rootB
$ starvault patch pki/issuer/intB manual_chain=self,rootB,intA,rootA
```

Это позволит гарантировать, что при выдаче любой из копий промежуточного документа при подписании листовых сертификатов будет отображаться полная цепочка перекрестных

подписей.

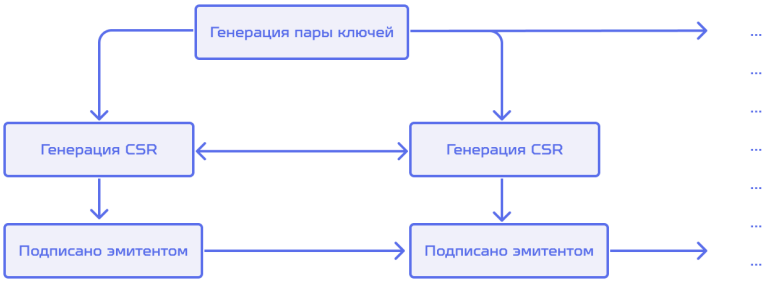
2.2. Примитив переиздания

Второй наиболее распространенный тип примитива ротации. В этой схеме существующий ключевой материал используется для генерации нового сертификата, обычно в гораздо более поздний момент времени по сравнению с существующим выпуском.

Хотя эта схема похожа на примитив с перекрестной подписью, она отличается тем, что обычно перевыпуск происходит после того, как срок действия оригинального сертификата истекает или близок к истечению, и перевыпускается оригинальным корневым центром сертификации. В случае самоподписанного сертификата (например, корневого сертификата) этот родительский сертификат будет являться самим собой. В обоих случаях содержимое сертификата изменяется (за счет нового серийного номера), но все существующие leaf подписи продолжают проверяться.

В отличие от примитива с перекрестной подписью, этот тип примитива может использоваться для всех типов сертификатов (включая leaves, промежуточные и корневые).

2.2.1. Технологический процесс



В этом процессе в определенный момент времени генерируется и хранится одна пара ключей. CSR (с теми же запрошенными полями) создается на основе этого общего ключевого материала и подписывается одним и тем же эмитентом в разные моменты времени, сохраняя все критические поля (субъект, эмитент и т. д.). Хотя количество перевыпусков ключа строго не ограничено, в определенный момент безопасность потребует ротации ключевого материала вместо постоянного перевыпуска.

2.2.2. Иерархия сертификатов

Обратите внимание, что хотя это снова приводит к двум путям доверия, в зависимости от того, какой промежуточный сертификат представлен и все еще действителен, доверять нужно только корневому. Когда перевыпущенный сертификат является корневым сертификатом, канал выдачи просто самозацикливается. Но в этом случае обратите внимание, что оба сертификата являются (технически) действительными эмитентами друг друга. Это означает, что в цепочке сертификатов TLS должно быть возможно предоставить перевыпущенный корневой сертификат, и он будет связан с существующим корневым сертификатом в хранилище доверия.

Таким образом, этот примитивный тип является возрастающим примитивом; жизненный цикл существующего ключа продлевается на будущее путем выдачи нового сертификата с тем же материалом ключа существующим центром сертификации.

2.2.3. Исполнение в хранилище

Чтобы создать перевыпущенный корневой сертификат в StarVault, используйте конечную точку `/issuers/generate/root/existing`. Это позволит сгенерировать новый корневой сертификат с существующим ключевым материалом (через параметр запроса `key_ref`). Если процесс прошел успешно, то при чтении эмитента (через `GET /issuer/:issuer_ref`) оба эмитента (старый и перевыпущенный) будут отображаться в поле ответа `ca_chain` друг друга (если это не запрещено значением `manual_chain`).

Создание перевыпущенного промежуточного сертификата в StarVault состоит из трех шагов:

1. Используйте конечную точку `/issuers/generate/intermediate/existing`, чтобы сгенерировать новый CSR с существующим ключевым материалом с параметром запроса `key_ref`.
2. Подпишите этот CSR с помощью того же процесса подписания под тем же эмитентом. Этот шаг зависит от родительского CA, который может быть или не быть StarVault.
3. Наконец, используйте конечную точку `/intermediate/set-signed` для импорта подписанного сертификата из шага 2.

Если процесс перевыпуска промежуточного сертификата прошел успешно, то при чтении эмитента (через `GET /issuer/:issuer_ref`) оба эмитента (старый и перевыпущенный) будут иметь одинаковое поле ответа `ca_chain`, за исключением первой записи (если это не запрещено значением `manual_chain`).

Независимо от типа эмитента, важно предоставить все соответствующие параметры в том виде, в котором они были изначально; StarVault не выводит, например, параметры имени субъекта из существующего эмитента; он просто повторно использует тот же самый ключевой материал.

2.3. Временные примитивы

Мы можем использовать вышеуказанные примитивные типы для замены корневых и промежуточных сертификатов на новые ключи и продления срока их службы. Такая ротация по времени в конечном итоге позволяет нам заменять корневые сертификаты.

Существует два основных варианта этого метода: **прямой примитив**, в котором старый сертификат используется для получения нового ключевого материала, и **обратный примитив**, в котором новый сертификат используется для получения старого ключевого материала. Оба этих примитива независимо используются Let's Encrypt в вышеупомянутом документе цепочки доверия:

- Ссылка от DST Root CA X3 к ISRG Root X1 является примером прямого примитива.
- Ссылка от ISRG Root X1 к R3 (которая первоначально была подписана DST Root CA X3) является примером обратного примитива.

Для большинства организаций с иерархической структурой CA для ротации корня достаточно перекрестной подписи всех промежуточных центров как в новом, так и в старом корневом CA.

Однако для организаций, которые напрямую выпускали сертификаты листьев от корня, старый корень нужно будет перевыпустить под новым корнем (с меньшим сроком действия), чтобы эти сертификаты могли продолжать подтверждаться. Это объединяет оба вышеупомянутых примитива (перекрестное подписание и перевыпуск) в один обратный примитивный шаг. В будущем эти организации, вероятно, должны перейти к более стандартной, иерархической структуре.

2.4. Ограничения примитивов

Поле расширения Authority Key Identifier сертификата может содержать либо keyIdentifier эмитента (хэш открытого ключа), либо оба поля - **Subject** и **Serial Number**. Генерация сертификатов со включенным последним (к счастью, в StarVault это невозможно, тем более что StarVault использует строго случайные серийные номера) не позволяет построить правильную кросс-подписанную цепочку без повторного выпуска одного и того же серийного номера, что не будет работать с хранилищами доверия и механизмами проверки большинства браузеров из-за кэширования сертификатов, использованных в успешных проверках. При использовании примитива перекрестной подписи (от другого CA) промежуточный сертификат может быть перевыпущен с тем же серийным номером, при условии, что предыдущий сертификат не был выпущен этим CA с таким серийным номером.

Это не работает при использовании примитива перевыпуска, поскольку технически это один и тот же орган, и, следовательно, этот орган должен выпускать сертификаты с уникальными серийными номерами.

3. Предлагаемая процедура корневой ротации

Ниже приведен рекомендуемый процесс, позволяющий легко добиться ротации и не оказывающий негативного влияния на организацию в целом, при условии, что используются следующие методы. Потребуется некоторая адаптация.

Обратите внимание, что этот процесс требует времени. Количество времени зависит от уровня автоматизации и оперативной осведомленности организации.

1. Создайте новый корневой сертификат. Предлагается использовать новое общее имя, чтобы отличить его от старого корневого сертификата. Материал ключа не обязательно должен быть тем же самым.
2. Перекрестно подпишите все существующие промежуточные сертификаты. Важно обновить ручную цепочку для эмитентов, как обсуждалось в этом разделе, поскольку мы предполагаем, что серверы настроены на объединение поля `certificate` с полем `ca_chain` при обновлении и выдаче что позволяет получить перекрестно подписанные промежуточные сертификаты.
3. Используйте ротацию для подбора новых промежуточных сертификатов с перекрестной подписью. Для недолговечных сертификатов это должно происходить автоматически. Однако для некоторых долгоживущих сертификатов рекомендуется ротировать их вручную и с упреждением. Этот шаг требует времени и зависит от типов выданных сертификатов (например, серверные сертификаты, сертификаты подписи кода или клиентские сертификаты).
4. После обновления всех цепочек новые системы можно выводить в сеть только с новым корневым сертификатом и подключаться ко всем существующим системам
5. Существующие системы теперь можно перенести с помощью одновременного переключения корневого сертификата: можно добавить новый корень и одновременно удалить старый. Если предположить, что вышеупомянутый шаг 3 может быть выполнен за разумное время, это сократит время, необходимое для перехода большинства систем на полное использование нового корня и отказа от доверия к старому корню. Этот шаг также требует времени, в зависимости от того, насколько быстро организация сможет перенести корни и обеспечить перенос всех таких систем. Если некоторые системы находятся в автономном режиме и лишь изредка выходят в сеть (или если в них хранятся жестко закодированные сертификаты и им необходимо сначала устареть), организация может быть не готова к переходу на следующие этапы.
6. На этом этапе, поскольку все системы теперь используют новый корень, можно удалить или заархивировать старый корень и промежуточные элементы, обновив цепочку

инструкций, чтобы она указывала строго на новый промежуточный элемент+корень.

На этом ротация полностью завершена.

4. API

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

2025 orionsoft. Все права защищены.

Настройка и использование

В этом документе приводится краткий обзор настройки и использования механизма секретов PKI.

1. Настройка

Большинство механизмов секретов должны быть предварительно настроены, прежде чем они смогут выполнять свои функции. Эти шаги обычно выполняются оператором или инструментом управления конфигурацией.

1. Включите механизм секретов PKI:

   ```
   $ starvault secrets enable pki
   Success! Enabled the pki secrets engine at: pki/
   ```
   BASH

   По умолчанию механизм секретов будет монтироваться по имени механизма. Чтобы подключить механизм секретов по другому пути, используйте аргумент `-path`.

2. Увеличьте TTL, настроив параметры механизма секретов. Значение по умолчанию 30 дней может быть слишком коротким, поэтому увеличьте его до 1 года:

   ```
   $ starvault secrets tune -max-lease-ttl=8760h pki
   Success! Tuned the secrets engine at: pki/
   ```
   BASH

   Обратите внимание, что отдельные роли могут ограничить это значение для каждого сертификата. Это настраивает глобальный максимум для данного механизма секретов.

3. Настройте сертификат CA и закрытый ключ. StarVault может принять существующую пару ключей или сгенерировать собственный root сертификат. В общем случае мы рекомендуем поддерживать root CA вне StarVault и предоставлять StarVault подписанный промежуточный CA.

   ```
   $ starvault write pki/root/generate/internal \
       common_name=my-website.com \
       ttl=8760h
   ```
   BASH

   Вывод:

Key	Value

| --- | ---- |
certificate	-----BEGIN CERTIFICATE-----…
expiration	1536807433
issuing_ca	-----BEGIN CERTIFICATE-----…
serial_number	7c:f1:fb:2c:6e:4d:99:0e:82:1b:08:0a:81:ed:61:3e:1d:fa:f5:29

Возвращаемый сертификат носит исключительно информационный характер. Закрытый ключ надежно хранится внутри StarVault.

4. Обновление местоположения CRL и сертификатов выпуска. Эти значения могут быть обновлены в будущем.

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

5. Настройте роль, связывающая имя в StarVault с процедурой генерации сертификатов. Когда пользователи или машины генерируют учетные данные, они генерируются в соответствии с этой ролью:

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

2. Использование

После того как механизм секретов настроен и у пользователя/машины есть токен StarVault с соответствующими правами, он может генерировать учетные данные.

Сгенерируйте новые учетные данные, записав в конечную точку `/issue` имя роли:

```bash
$ starvault write pki/issue/example-dot-com \
    common_name=www.my-website.com
```

Вывод:

Key	Value

| --- | ---- |
certificate	-----BEGIN CERTIFICATE-----…
issuing_ca	-----BEGIN CERTIFICATE-----…
private_key	-----BEGIN RSA PRIVATE KEY-----…
private_key_type	rsa
serial_number	1d:2e:c6:06:45:18:60:0e:23:d6:c5:17:43:c0:fe:46:ed:d1:50:be

На выходе мы получим динамически сгенерированный закрытый ключ и сертификат, который соответствует заданной роли и истекает через 72 часа (как диктует наше определение роли). Для простоты автоматизации также возвращается CA выдачи и цепочка доверия.

3. API

Движок секретов PKI имеет полный HTTP API. Более подробная информация приведена в разделе API движка PKI secrets.

2025 orionsoft. Все права защищены.