WEBADM AS SUBORDINATE CERTIFICATE AUTHORITY

WebADM As Subordinate Certificate Authority

1. Overview

RCDevs' suite offers a public key infrastructure service and that functionality is mandatory for the proper functioning of RCDevs solutions.

The default setup is to make WebADM/Rsignd a standalone CA. In that scenario, you just need to follow the default WebADM setup.

For customers which already have a CA in place and running, you can configure WebADM as a subordinate CA.

This document will present you with how to configure WebADM as a subordinate certificate authority of your enterprise certificate authority. In that document, we demonstrate it with a root CA configured on Windows Server 2019. You need at least WebADM v.2.1.

The steps to achieve this configuration are the following :

- Have a root/enterprise or subordinate CA already configured (not shown in that document),
- Generate from your enterprise CA a subordinate CA certificate and key pair to be used by WebADM (not shown in that document),
- Import the subordinate certificate and key pair freshly generated in WebADM,
- Generate the cerfticate signing request (CSR) and the key which will be used by WebADM SOAP services (webadm.csr and webadm.key),
- Use OpenSSL to submit the CSR and get back the certificate issued and signed by our new subordinate CA.

Important

These steps must be done before you implement anything with WebADM and its components or before you issued any certificate with WebADM internal PKI. Else you will have to renew all certificates used in each deployed component using the SOAP API (port 8443). Actions must be performed just after the `/opt/webadm/bin/setup` script on the master node before configuring WebADM in cluster mode. Once the master is properly configured as a subordinate CA, the slave setup can be normally done as described in the High Availability documentation.

2. ROOT/Enterprise Certificate Authority

Please, refer to your PKI provider documentation to set up your root/enterprise certificate authority. We assume in that documentation that the root CA is already configured. My enterprise certificate authority is called **SUPCAAD2**. Found below, the details of my root CA :

Found below, the content of my root CA certificate :

```
-----BEGIN CERTIFICATE-----
MIIDpTCCAo2gAwIBAgIQVz4fdQJ2QZhGnJ2/KomwAjANBgkqhkiG9w0BAQsFADBZ
MRMwEQYKCZImiZPyLGQBGRYDY29tMRYwFAYKCZImiZPyLGQBGRYGcmNkZXZzMRcw
FQYKCZImiZPyLGQBGRYHc3VwcG9ydDERMA8GA1UEAxMIU1VQQ0FBRDIwHhcNMjEw
NzA2MTYxNTU5WhcNNDEwNzA2MTYyNTU4WjBZMRMwEQYKCZImiZPyLGQBGRYDY29t
MRYwFAYKCZImiZPyLGQBGRYGcmNkZXZzMRcwFQYKCZImiZPyLGQBGRYHc3VwcG9y
dDERMA8GA1UEAxMIU1VQQ0FBRDIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQDCVs3A/ksabY9Ljz+MhZbbQR4qZB6m/ofHdmzB+Am7Tsv4x9ulbB597jDQ
qNaOoW6QVtUGHZYoYFXswiteyibFUIGTEjdPhDxU/SYvmGnmwZ0bEHIlS4/xJYW/
MvO3ykEul6Ugo8zAVc5ABQ1CjHi+ZtuaLy+A9eyxAeFL1emxvlbUn2mQvmmbUOtX
Xt3c15iY33fAqgE1+pHm6E7MdUrVpESX5ynaQCgeL9BTgNbfuS00rk62UPoCx0q1
C/K9Dql7f7PQRntk56NKdeIb600/xRuUcDx8XK1uyua1HFHZS2q7pjwlZFG8aRV2
ly1S8PxvalPi9RzXEO9AX+Mft/6lAgMBAAGjaTBnMBMGCSsGAQQBgjcUAgQGHgQA
QwBBMA4GA1UdDwEB/wQEAwIBhjAPBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQWBBTa
8I08lOjn2sDzuzREii1NThrF3zAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0B
AQsFAAOCAQEAODLcWmSutShVG3gPpX5dCxOKGsfBeGJirpCrupSWeDy2oAsyZUfl
vww/JYibjBD2RxOOikA8sAtjMmivzvLWaHkTZVpxgTQEUqoHqi0WAkJjTvAZljvt
Oqxlso8HZPqZXAuVHW4wOZaLKeZSoE2qJjAuYfHVI7ZILqUHeQ7i28nuZvQJtnDw
z292RfmbFE6iL9UAIe7i6KdVS6Apx9PJyt8vjN80hLEc2h94KKCK2S/q6cSTbllP
JHMSm8SEGSHOAfdr6cVtLqH9FCxNQmyoPI8ISyQa6/HqrdwENoeExQ2XI059vkgy
/RjlEukzs8c3B7cw9C3ViG7AVPwxr0bosw==
-----END CERTIFICATE-----
```

Information contained in my CA certificate read with OpenSSL :

```
openssl x509 -in SUPCAAD2.crt -text -noout
```

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            57:3e:1f:75:02:76:41:98:46:9c:9d:bf:2a:89:b0:02
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: DC = com, DC = rcdevs, DC = support, CN = SUPCAAD2
        Validity
            Not Before: Jul  6 16:15:59 2021 GMT
            Not After : Jul  6 16:25:58 2041 GMT
        Subject: DC = com, DC = rcdevs, DC = support, CN = SUPCAAD2
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:c2:56:cd:c0:fe:4b:1a:6d:8f:4b:8f:3f:8c:85:
```

```
                00:c2:56:cd:c0:fe:4b:1a:6d:8f:4b:8f:3f:8c:65:
                96:db:41:1e:2a:64:1e:a6:fe:87:c7:76:6c:c1:f8:
                09:bb:4e:cb:f8:c7:db:a5:6c:1e:7d:ee:30:d0:a8:
                d6:8e:a1:6e:90:56:d5:06:1d:96:28:60:55:ec:c2:
                2b:5e:ca:26:c5:50:81:93:12:37:4f:84:3c:54:fd:
                26:2f:98:69:e6:c1:9d:1b:10:72:25:4b:8f:f1:25:
                85:bf:32:f3:b7:ca:41:2e:97:a5:20:a3:cc:c0:55:
                ce:40:05:0d:42:8c:78:be:66:db:9a:2f:2f:80:f5:
                ec:b1:01:e1:4b:d5:e9:b1:be:56:d4:9f:69:90:be:
                69:9b:50:eb:57:5e:dd:dc:d7:98:98:df:77:c0:aa:
                01:35:fa:91:e6:e8:4e:cc:75:4a:d5:a4:44:97:e7:
                29:da:40:28:1e:2f:d0:53:80:d6:df:b9:2d:34:ae:
                4e:b6:50:fa:02:c7:4a:b5:0b:f2:bd:0e:a9:7b:7f:
                b3:d0:46:7b:64:e7:a3:4a:75:e2:1b:eb:4d:3f:c5:
                1b:94:70:3c:7c:5c:ad:6e:ca:e6:b5:1c:51:d9:4b:
                6a:bb:a6:3c:25:64:51:bc:69:15:76:97:2d:52:f0:
                fc:6f:6a:53:e2:f5:1c:d7:10:ef:40:5f:e3:1f:b7:
                fe:a5
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        1.3.6.1.4.1.311.20.2:
            ...C.A
        X509v3 Key Usage: critical
            Digital Signature, Certificate Sign, CRL Sign
        X509v3 Basic Constraints: critical
            CA:TRUE
        X509v3 Subject Key Identifier:
            DA:F0:8D:3C:94:E8:E7:DA:C0:F3:BB:34:44:8A:2D:4D:4E:1A:C5:DF
        1.3.6.1.4.1.311.21.1:
            ...
Signature Algorithm: sha256WithRSAEncryption
     38:32:dc:5a:64:ae:b5:28:55:1b:78:0f:a5:7e:5d:0b:13:8a:
     1a:c7:c1:78:62:62:ae:90:ab:ba:94:96:78:3c:b6:a0:0b:32:
     65:47:e5:bf:0c:3f:25:88:9b:8c:10:f6:47:13:8e:8a:40:3c:
     b0:0b:63:32:68:af:ce:f2:d6:68:79:13:65:5a:71:81:34:04:
     52:aa:07:aa:2d:16:02:42:63:4e:f0:19:96:3b:ed:3a:ac:65:
     b2:8f:07:64:fa:99:5c:0b:95:1d:6e:30:39:96:8b:29:e6:52:
     a0:4d:aa:26:30:2e:61:f1:d5:23:b6:48:2e:a5:07:79:0e:e2:
     db:c9:ee:66:f4:09:b6:70:f0:cf:6f:76:45:f9:9b:14:4e:a2:
     2f:d5:00:21:ee:e2:e8:a7:55:4b:a0:29:c7:d3:c9:ca:df:2f:
     8c:df:34:84:b1:1c:da:1f:78:28:a0:8a:d9:2f:ea:e9:c4:93:
     6e:59:4f:24:73:12:9b:c4:84:19:21:ce:01:f7:6b:e9:c5:6d:
     2e:a1:fd:14:2c:4d:42:6c:a8:3c:8f:08:4b:24:1a:eb:f1:ea:
     ad:dc:04:36:87:84:c5:0d:97:23:4e:7d:be:48:32:fd:18:e5:
     12:e9:33:b3:c7:37:07:b7:30:f4:2d:d5:88:6e:c0:54:fc:31:
     af:46:e8:b3
```

3. Subordinate CA certificate

Please, refer to your PKI provider documentation to generate a Subordinate CA certificate which will be used by WebADM. Found below, the details of my subordinate certificate which will be imported on WebADM to make WebADM a subordinate certificate

authority of your enterprise CA. My WebADM server which will be configured as SubCA is named **webadm2.support.rcdevs.com**. Found below the details:

Below, the proof that is has been issued by my root/enterprise certificate authority:

Found below the content of my subordinate certificate file and his key which will be used by WebADM as ca.crt and ca.key in `/opt/webadm/pki/ca/` folder:

Subordinate CA certificate (ca.crt):

```
-----BEGIN CERTIFICATE-----
MIIFWDCCBECgAwIBAgITHAAAAAqFCxCd4ea1rgAAAAAACjANBgkqhkiG9w0BAQsF
ADBZMRMwEQYKCZImiZPyLGQBGRYDY29tMRYwFAYKCZImiZPyLGQBGRYGcmNkZXZz
MRcwFQYKCZImiZPyLGQBGRYHc3VwcG9ydDERMA8GA1UEAxMIU1VQQ0FBRDIwHhcN
MjEwNzI4MTU1NDMxWhcNMjMwNzI4MTYwNDMxWjCBpzELMAkGA1UEBhMCTFUxEzAR
BgNVBAgTCkx1eGVtYm91cmcxDzANBgNVBAcTBkJlbHZhbDEXMBUGA1UEChMOU3Vw
cG9ydCBSQ0RldnMxCzAJBgNVBAsTAklUMSMwIQYDVQQDExp3ZWJhZG0yLnN1cHBv
cnQucmNkZXZzLmNvbTEnMCUGCSqGSIb3DQEJARYYeW9hbm5Ac3VwcG9ydC5yY2Rl
dnMuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCXQsn7nClgZmS4XbFy
CkNPmiT7c9rgCdZeR9jo1uKQkXzzJ8p8bmSPd+Xd39IupHJbSRYbNyDtN425Q/wW
FYBTmwdBCWMQkGF5itke6sim6pOG8a8EVbMKH/WLEzfBoeR2zIzEFTueBp+NGwXy
1VVgdYman26wNsXPvQEI2YkxbQIDAQABo4ICTDCCAkgwDgYDVR0PAQH/BAQDAgGG
MBkGCSsGAQQBgjcUAgQMHgoAUwB1AGIAQwBBMB0GA1UdDgQWBBQtOARJo1wNiAcp
L4GsyoIlL7aoGzAfBgNVHSMEGDAWgBTa8I08lOjn2sDzuzREii1NThrF3zCBzwYD
VR0fBIHHMIHEMIHBoIG+oIG7hoG4bGRhcDovLy9DTj1TVVBDQUFEMixDTj1BRDE5
LTIsQ049Q0RQLENOPVB1YmxpYyUyMEtleSUyMFNlcnZpY2VzLENOPVNlcnZpY2Vz
LENOPUNvbmZpZ3VyYXRpb24sREM9c3VwcG9ydCxEQz1yY2RldnMsREM9Y29tP2Nl
cnRpZmljYXRlUmV2b2NhdGlvbkxpc3Q/YmFzZT9vYmplY3RDbGFzcz1jUkxEaXN0
cmlidXRpb25Qb2ludDCB9wYIKwYBBQUHAQEEgeowgecwgbEGCCsGAQUFBzAChoGk
bGRhcDovLy9DTj1TVVBDQUFEMixDTj1BSUEsQ049UHVibGljJTIwS2V5JTIwU2Vy
dmljZXMsQ049U2VydmljZXMsQ049Q29uZmlndXJhdGlvbixEQz1zdXBwb3J0LERD
PXJjZGV2cyxEQz1jb20/Y0FDZXJ0aWZpY2F0ZT9iYXNlP29iamVjdENsYXNzPWNl
cnRpZmljYXRpb25BdXRob3JpdHkwMQYIKwYBBQUHMAGGJWh0dHA6Ly9BRDE5LTIu
c3VwcG9ydC5yY2RldnMuY29tL29jc3AwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG
9w0BAQsFAAOCAQEAoR5d7dImJflXfbnqLS+kvpIp59X7HQGt8MnXj74iesoxRQES
i0LmizyroD4UGS0zzTXjQUVfRQqoSesdSKEtekXJSPue3KS0i+HCfCPhK4YxTojV
rdmYrpCjf0SoYaOaQoNmQdt28WvJ+WlQU91gGRdiyN5zVuzyz8cR2NLIhSdKCvba
zAZeUdkKlX16KlVhthdvmtJfhEjePef8/BOpzHVVAnNkhqRuPZOEk2bbH05wBJKa
fHFnLy1o/BI08b0ye7fnTLYiWxoJahD5PYF8QlcMMgl6glBD67VSWvyCH7NVv46Y
jLlj9pUdR7bis7OJKtkXoJoIHRrlrARAjYx8rA==
-----END CERTIFICATE-----
```

Subordinate CA key (ca.key) :

```
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAJdCyfucKWBmZLhd
sXIKQ0+aJPtz2uAJ1l5H2OjW4pCRfPMnynxuZI935d3f0i6kcltJFhs3IO03jblD
/BYVgFObB0EJYxCQYXmK2R7qyKbqk4bxrwRVswof9YsTN8Gh5HbMjMQVO54Gn40b
BfLVVWB1iZqfbrA2xc+9AQjZiTFtAgMBAAECgYB1jozYJbfqmemxd+/Ox7ckiJIx
cwsqj7qxL3mfDFKtNMU9XTF7n3g8IFzgIXGCmn/d/mFV6XSHqGNXF0U2DKPcJHgX
VFhooj52v6yh3a18L0ngAFd84jyMvNM1PsG9iHPvn2sGDy05qCdmnLhABWksdhHY
00Bf6FuHKgLL1MTxaQJBAMgV4UCIIBylVIwuQKSHDJYrU5UjTSUIUwUQHEAWJxQ7
rHnZszHDuAYWXOUJvNOuJ4a6UXi3qiqiB7rytDe5Bz8CQQDBh/9pZo9LR35WmO1m
x0m/SDAX6/rFyM6FFAEntz1bs9PJEjumUwM8sXGR4NhXbtHr5Jywkz3vIcNgU3DA
FihTAkADsCOOxPOSk5mTW+blIXgh7HqF7Timzhh5p2pd5AqkXNU5CcI70Je7xP3B
WwSYAkXIPfbyerAwSPxLfd3EiSyRAkBrNv+Nkc7iwonASdqDbPZzLPfP2ODFv9iB
qzJ0oTQx4G783sgC/cw2TIuBaJIR5ggP6kfQHtJZ71eAvtkg4WWXAkEAvpelsk/5
2Mv0m+VG+nWWCtA9ncwqfwfIZ9HGHOs4KZ2uhglac17XQWvkKbpIB8sYKvG9qNey
HlDEqQ7fouU/+w==
-----END PRIVATE KEY-----
```

Information contained in the certificate read with OpenSSL :

```
openssl x509 -in ca.crt -text -noout
```

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            1c:00:00:00:0a:85:0b:10:9d:e1:e6:b5:ae:00:00:00:00:00:0a
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: DC = com, DC = rcdevs, DC = support, CN = SUPCAAD2
        Validity
            Not Before: Jul 28 15:54:31 2021 GMT
            Not After : Jul 28 16:04:31 2023 GMT
        Subject: C = LU, ST = Luxembourg, L = Belval, O = Support RCDevs, OU = IT, CN =
webadm2.support.rcdevs.com, emailAddress = yoann@support.rcdevs.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (1024 bit)
                Modulus:
                    00:97:42:c9:fb:9c:29:60:66:64:b8:5d:b1:72:0a:
                    43:4f:9a:24:fb:73:da:e0:09:d6:5e:47:d8:e8:d6:
                    e2:90:91:7c:f3:27:ca:7c:6e:64:8f:77:e5:dd:df:
                    d2:2e:a4:72:5b:49:16:1b:37:20:ed:37:8d:b9:43:
                    fc:16:15:80:53:9b:07:41:09:63:10:90:61:79:8a:
                    d9:1e:ea:c8:a6:ea:93:86:f1:af:04:55:b3:0a:1f:
                    f5:8b:13:37:c1:a1:e4:76:cc:8c:c4:15:3b:9e:06:
                    9f:8d:1b:05:f2:d5:55:60:75:89:9a:9f:6e:b0:36:
                    c5:cf:bd:01:08:d9:89:31:6d
```

```
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            1.3.6.1.4.1.311.20.2:
                .
.S.u.b.C.A
            X509v3 Subject Key Identifier:
                2D:38:04:49:A3:5C:0D:88:07:29:2F:81:AC:CA:82:25:2F:B6:A8:1B
            X509v3 Authority Key Identifier:
                keyid:DA:F0:8D:3C:94:E8:E7:DA:C0:F3:BB:34:44:8A:2D:4D:4E:1A:C5:DF

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:ldap:///CN=SUPCAAD2,CN=AD19-
2,CN=CDP,CN=Public%20Key%20Services,CN=Services,CN=Configuration,DC=support,DC=rcdevs,DC=
certificateRevocationList?base?objectClass=cRLDistributionPoint

            Authority Information Access:
                CA Issuers -
URI:ldap:///CN=SUPCAAD2,CN=AIA,CN=Public%20Key%20Services,CN=Services,CN=Configuration,DC
cACertificate?base?objectClass=certificationAuthority
                OCSP - URI:http://AD19-2.support.rcdevs.com/ocsp

            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         a1:1e:5d:ed:d2:26:25:f9:57:7d:b9:ea:2d:2f:a4:be:92:29:
         e7:d5:fb:1d:01:ad:f0:c9:d7:8f:be:22:7a:ca:31:45:01:12:
         8b:42:e6:8b:3c:ab:a0:3e:14:19:2d:33:cd:35:e3:41:45:5f:
         45:0a:a8:49:eb:1d:48:a1:2d:7a:45:c9:48:fb:9e:dc:a4:b4:
         8b:e1:c2:7c:23:e1:2b:86:31:4e:88:d5:ad:d9:98:ae:90:a3:
         7f:44:a8:61:a3:9a:42:83:66:41:db:76:f1:6b:c9:f9:69:50:
         53:dd:60:19:17:62:c8:de:73:56:ec:f2:cf:c7:11:d8:d2:c8:
         85:27:4a:0a:f6:da:cc:06:5e:51:d9:0a:95:7d:7a:2a:55:61:
         b6:17:6f:9a:d2:5f:84:48:de:3d:e7:fc:fc:13:a9:cc:75:55:
         02:73:64:86:a4:6e:3d:93:84:93:66:db:1f:4e:70:04:92:9a:
         7c:71:67:2f:2d:68:fc:12:34:f1:bd:32:7b:b7:e7:4c:b6:22:
         5b:1a:09:6a:10:f9:3d:81:7c:42:57:0c:32:09:7a:82:50:43:
         eb:b5:52:5a:fc:82:1f:b3:55:bf:8e:98:8c:b9:63:f6:95:1d:
         47:b6:e2:b3:b3:89:2a:d9:17:a0:9a:08:1d:1a:e5:ac:04:40:
         8d:8c:7c:ac
```

This certificate and the key will replace the default `ca.crt` and `ca.key` generated after the WebADM setup script in `/opt/webadm/pki/ca/` folder. Remove the existing `ca.crt` and `ca.key` files available in that folder and copy the new ones. The new `ca.crt` file must also be copied in `/opt/webadm/pki/trusted/` folder. All CA certificates chain between your root CA and the WebADM CA must also be copied in the `trusted/` folder. Once the 2 (or more) CA certificates are copied in the `trusted/` folder do the make command inside the folder:

```
[root@webadm2 tmp]# cd /opt/webadm/pki/trusted/
[root@webadm2 trusted]# make
ADROOT.crt      ... 148c96ae.0
ca.crt          ... faff8618.0

[root@webadm2 trusted]# ls -al
lrwxrwxrwx. 1 root root   10 Dec  8 11:54 148c96ae.0 -> ADROOT.crt
-rw-r--r--. 1 root root 1326 Aug 16 11:37 ADROOT.crt
-rw-r--r--. 1 root root 1071 Dec  8 10:46 Makefile
-rw-r--r--. 1 root root 1915 Aug 16 11:38 ca.crt
lrwxrwxrwx. 1 root root    6 Dec  8 11:54 faff8618.0 -> ca.crt
```

My WebADM server is now configured as a subordinate CA of my root/enterprise CA. The next step is to generate the certificate for WebADM services usage.

3. Regenerate WebADM certificate and key based on CSR

WebADM/Rsignd is now configured as a subordinate CA of our enterprise CA and we need now to issue a certificate and its associate key which will be used by WebADM SOAP services and signed by our new subordinate CA.

On your WebADM server, navigate in `/opt/webadm/pki/` and you should find the default `webadm.crt`, `webadm.key` and `webadm.csr` generated during the first WebADM setup script.

```
[root@webadm2 trusted]# cd /opt/webadm/pki/
[root@webadm2 pki]# ls -al
total 12
drwx------. 2 root root     69 Dec  8 10:46 ca
drwxr-xr-x. 2 root root     90 Dec  8 11:54 trusted
-rw-r-----. 1 root webadm 1164 Aug 26 10:44 webadm.crt
-rw-r--r--. 1 root root   1094 Aug 16 11:28 webadm.csr
-rw-r-----. 1 root webadm 1708 Aug 16 11:27 webadm.key
```

You can remove `webadm.crt` and keep the existing `webadm.key` :

```
[root@webadm2 pki]# rm -f webadm.crt
```

The following command will generate a new certificate based on the existing CSR (webadm.csr) and generate the associate certificate signed by our subordinate CA (WebADM):

```
[root@webadm2 pki]# openssl x509 -req -days 365 -in /opt/webadm/pki/webadm.csr -out
/opt/webadm/pki/webadm.crt -CA /opt/webadm/pki/ca/ca.crt -CAkey
/opt/webadm/pki/ca/ca.key -CAserial /opt/webadm/pki/ca/serial
```

> **Note**
>
> In the previous command, I configure the validity to 365 days. You can extend the validity if needed. WebADM should auto-renew its certificate when the certificate is near expiration after a restart of WebADM services.

You should have the following output if eveything goes fine :

```
Signature ok
subject=C = LU, ST = Luxembourg, L = LU, O = RCDevs Support, OU = ITSEC, CN = webadm2,
emailAddress = yoann@support.rcdevs.com
Getting CA Private Key
```

And the certificate should now be there :

```
[root@webadm2 pki]# ls -al
total 16
drwx------. 2 root root     69 Dec  8 10:46 ca
drwxr-xr-x. 2 root root     90 Dec  8 11:54 trusted
-rw-r--r--. 1 root root   1164 Dec  8 12:23 webadm.crt
-rw-r--r--. 1 root root   1094 Aug 16 11:28 webadm.csr
-rw-r-----. 1 root webadm 1708 Aug 16 11:27 webadm.key
```

Below, I execute the OpenSSL verification command to check the newly generated certificate match with the old key:

```
[root@webadm2 pki]# openssl rsa -modulus -noout -in webadm.key | openssl md5
(stdin)= 67ded38c0ad5bf0ab68dac8e87e0f5fb

[root@webadm2 pki]# openssl x509 -modulus -noout -in webadm.crt | openssl md5
(stdin)= 67ded38c0ad5bf0ab68dac8e87e0f5fb
```

It matches!

Now, I verify if the issuer is correct for my new WebADM certificate. It must be issued by my WebADM Subordinate CA :

```
[root@webadm2 pki]# openssl x509 -in webadm.crt -noout -text

Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 58 (0x3a)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = LU, ST = Luxembourg, L = Belval, O = Support RCDevs, OU = IT, CN =
webadm2.support.rcdevs.com, emailAddress = yoann@support.rcdevs.com
        Validity
            Not Before: Dec  8 11:23:33 2021 GMT
            Not After : Dec  8 11:23:33 2022 GMT
        Subject: C = LU, ST = Luxembourg, L = LU, O = RCDevs Support, OU = ITSEC, CN =
webadm2, emailAddress = yoann@support.rcdevs.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:cc:39:17:43:8d:a9:be:16:1c:79:25:1f:e6:9f:
                    23:ed:64:6f:c6:53:87:9a:65:d1:a5:52:04:46:91:
                    2e:01:3d:39:24:82:33:48:e1:cc:08:32:86:fa:3a:
                    a5:67:9d:79:43:07:07:a2:43:50:b0:b3:fe:e9:41:
                    d1:af:3a:ae:9c:8c:cc:9f:fb:66:5e:af:53:8b:a5:
                    d6:5e:4f:83:a4:58:ce:b8:8a:de:ff:46:cd:02:90:
                    8d:75:16:0b:87:cf:eb:f4:bd:91:6f:d2:fe:06:5c:
                    3a:e5:fd:1d:73:25:20:80:8e:73:99:eb:ef:8b:41:
                    ce:1c:f5:f8:27:aa:85:07:e0:76:8d:4a:97:e3:98:
                    83:ba:c8:20:87:08:60:e6:7d:19:a8:17:55:a1:c1:
                    26:6d:5a:6b:c1:a0:3c:70:b2:92:b3:80:92:e7:f2:
                    3a:61:0c:ec:15:cd:c6:d8:ff:ed:f9:8c:c3:e6:11:
                    2e:5a:4e:7c:c6:2c:cc:c6:73:2d:6b:9d:63:26:92:
                    c0:6d:b9:5b:dd:27:50:3d:cc:3c:ee:de:5a:e8:6a:
                    a0:b8:21:8e:47:72:b0:a3:67:58:aa:17:55:0c:44:
                    eb:89:b1:6f:e0:74:b8:c7:70:30:82:9b:96:ab:a3:
                    43:9a:4f:a4:9c:56:3e:7f:a3:8e:63:5b:3d:d1:15:
                    89:2f
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
         32:50:a6:75:f5:6f:1a:c4:a1:ea:77:51:fb:85:a4:e6:99:e9:
         57:ed:4d:e8:38:4a:72:b5:49:8a:04:70:23:64:94:40:cb:b5:
         a5:ab:26:9d:08:41:23:1e:6f:e3:6e:0b:65:a1:45:a9:70:51:
         91:49:fd:3c:9a:bf:fd:88:84:e4:93:a6:b8:57:af:28:2e:9e:
         41:46:d5:4d:eb:8c:90:7f:29:03:98:53:bf:f8:46:8c:db:3b:
         ac:dd:f5:02:cb:c8:81:7f:45:ca:1b:25:d9:31:db:8a:ad:17:
         64:c3:3f:63:c2:4b:60:f1:17:f6:78:a6:af:50:e6:a7:ff:f2:
         33:af
```

Everything looks good.

You can now start/restart WebADM services and check the Rsignd service (PKI) is working correctly :

```
[root@webadm2 pki]# /opt/webadm/bin/webadm start
hecking libudev dependency... Ok
Checking system architecture... Ok
Checking server configurations... Ok

Found Trial license (RCDEVSSUPPORT)
Licensed by RCDevs Security SA to RCDevs Support
Licensed product(s): OpenOTP,SpanKey

Starting WebADM PKI service... Ok
Starting WebADM Session service... Ok
Starting WebADM Watchd service... Ok
Starting WebADM HTTP service... Ok

Checking server connections...
Connected LDAP server: AD1 (192.168.4.2)
Connected SQL server: SQL19 (192.168.4.4)
Connected PKI server: PKI Server (localhost)
Connected Mail server: SMTP Server (192.168.4.1)
Connected Session server: Session Server 1 (192.168.4.20)

Checking LDAP proxy user access... Ok
Checking SQL database access... Ok
Checking PKI service access... Ok
Checking Mail service access... Ok
Checking Cloud service access... Ok

Cluster mode enabled with 2 nodes (I'm master)
```

In the previous output, you can see the 3 following lines which indicate everything is fine :

```
Starting WebADM PKI service... Ok
...
Connected PKI server: PKI Server (localhost)
...
Checking PKI service access... Ok
```

Everything is now configured correctly.

You can double check the certificate used on your WebADM through your web browser :

4. Issue clients, servers and users certificates

Now our WebADM is configured as a subordinate CA, we will check that WebADM can generate the different kinds of certificates which can be issued by WebADM/Rsignd.

4.1 Issue an OpenOTP or Spankey client certificate

Login on WebADM administrator portal, click `Admin` tab and then click on `Issue Server or Client SSL Certificate`.

You arrive on the following page :

To generate a client cerificate, change the `certificate type` to a `client` and provide the information of your client. Here, I restricted the certificate usage to OpenOTP service only :

Once you completed all needed information, press `Ok` button to submit the request to Rsignd service and then the certificate and key for your OpenOTP client are generated:

Download the certificate and key pair and you can use them on your OpenOTP clients (e.g OpenOTP Credential Provider).

4.2 Issue a server certificate (for Radius Bridge, WAPRoxy…)

Login on WebADM administrator portal, click `Admin` tab and then click on `Issue Server or Client SSL Certificate`.

You arrive on the following page :

To generate a server cerificate, choose the `certificate type` to `Server` and provide the information of your server.

Once you have completed all needed information, press the `Ok` button to submit the request to the Rsignd service and then the certificate and key for your OpenOTP client are generated:

Download the certificate and the key and you can import them on your WAProxy server.

4.3 Issue a user certificate

User/admin certificates can be used to log in on RCDevs Web applications (selfdesk, helpdesk, pwreset…), WebADM admin portal (only with admin certificate) or with custom integrations using OpenOTP PKI logins method.

Log in on the WebADM administrator portal and click on the user account in the left LDAP tree for who you want to generate a new certificate.

Once you are on the user account, in `LDAP Actions` box, click on `Create certificate` button and you are then prompted for the following :

Configure the `validity` for that certificate, the `certificate usage` and optionally the `alternative names`.

The certificate usage can be configured to `Admin` or `User`. `Admin` certificates are allowed only for super_admin or other_admin users. `User` certificates are for all regular users.

`Admin` certificates allow you to log in on WebADM Admin GUI, not `user` certificate.

Click `Create Cert` and the certificate will be created.

Copy the password and download the certificate bundle. When you will try to import it into your certificate store, the password will be required.

Below, the details of that generated certificate :