Link to Forum

How to create an IdM certificate profile to request wildcard certificates?

SOLUTION VERIFIED - Updated June 14 2024 at 6:18 PM - English

Environment

- Red Hat Enterprise Linux
- Identity-Management (IPA)

Issue

- I need to create a wildcard certificate like *.example.com.
- My users are using latest web browsers, therefore I need to create certificates that not only contain the wildcard DNS name as Subject Common Name (CN) but also as Subject Alternative Name (SAN) extension.

Resolution

The following KCS article already shows how existing certificates can be reissued with a DNS name as Subject Alternative Name (SAN) extension: https://access.redhat.com/solutions/3027401.

This article shows how a new certificate profile can be created that allows wildname certificates to be issued with the DNSName also defined as Subject Alternative Name (SAN) extension.

To create a new certificate profile, just export an existing profile and customize it as explained below:

```
 $ ipa certprofile-show caIPAserviceCert --out wildcard.cfg
--------------------------------------------------
Profile configuration stored in file 'wildcard.cfg'
--------------------------------------------------
  Profile ID: caIPAserviceCert
  Profile description: Standard profile for network services
  Store issued certificates: TRUE
```

In the `wildcard.cfg` file, change the following two parameters to look like this. Make sure to keep the correct DNS name that matches your environment:

```
policyset.serverCertSet.1.default.params.name=CN=*.$request.req_subject_name.cn$, O=EXAMPLE.COM
profileId=caWildcardCert
```

In addition to that, please add the following lines to the profile:

```
policyset.serverCertSet.12.constraint.class_id=noConstraintImpl
policyset.serverCertSet.12.constraint.name=No Constraint
policyset.serverCertSet.12.default.class_id=subjectAltNameExtDefaultImpl
policyset.serverCertSet.12.default.name=Subject Alternative Name Extension Default
policyset.serverCertSet.12.default.params.subjAltNameNumGNs=2
policyset.serverCertSet.12.default.params.subjAltExtGNEnable_0=true
policyset.serverCertSet.12.default.params.subjAltExtType_0=DNSName
policyset.serverCertSet.12.default.params.subjAltExtPattern_0=*.$request.req_subject_name.cn$
policyset.serverCertSet.12.default.params.subjAltExtGNEnable_1=true
policyset.serverCertSet.12.default.params.subjAltExtType_1=DNSName
policyset.serverCertSet.12.default.params.subjAltExtPattern_1=$request.req_subject_name.cn$
```

Please note, the index used in this example does not matter as long as it is different from the other profile policy components. But please be sure to add the index to the directive containing the list of profile policies:

```
policyset.serverCertSet.list=1,2,3,4,5,6,7,8,9,10,11,12
```

This configuration will cause two SAN DNSName values to be added to the certificate – one using the CN from the CSR, and the other using the CN from the CSR preceded by a wildcard label.

Finally, be aware that because the subjectAltNameExtDefaultImpl component adds the SAN extension to a certificate, it conflicts with the `userExtensionDefault` component when configured to copy the SAN extension from a CSR to the new certificate. This profile component will have a configuration like the following:

```
policyset.serverCertSet.11.constraint.class_id=noConstraintImpl
policyset.serverCertSet.11.constraint.name=No Constraint
policyset.serverCertSet.11.default.class_id=userExtensionDefaultImpl
policyset.serverCertSet.11.default.name=User Supplied Extension Default
policyset.serverCertSet.11.default.params.userExtOID=2.5.29.17
```

Again the numerical index is indicative only, but the OID is not; `2.5.29.17` is the OID for the SAN extension. If your starting profile configuration contains the same directives, remove them from the configuration, and remove the index from the policy list too:

```
policyset.serverCertSet.list=1,2,3,4,5,6,7,8,9,10,12
```

Finally import the new certificate profile:

```
$  ipa certprofile-import caWildcardCert --file wildcard.cfg --desc 'Wildcard certificates'     --store 1
-------------------------------
Imported profile "caWildcardCert"
-------------------------------
  Profile ID: caWildcardCert
  Profile description: Wildcard certificates
  Store issued certificates: TRUE
```

You also need to define a `CA ACL` to allow the wildcard profile to be used with certain example.com host. In this example the hostgroup "webservers" is used. Please adjust based on your environment:

```
$ ipa caacl-add wildcard-hosts
-----------------------------
Added CA ACL "wildcard-hosts"
-----------------------------
  ACL name: wildcard-hosts
  Enabled: TRUE

$ ipa caacl-add-profile wildcard-hosts --certprofiles wildcardCert
  ACL name: wildcard-hosts
  Enabled: TRUE
  Profiles: caWildcardCert
-------------------------
Number of members added 1
-------------------------

$ ipa caacl-add-host wildcard-hosts --hostgroups webservers
  ACL name: wildcard-hosts
  Enabled: TRUE
  Profiles: caWildcardCert
  Host Groups: webservers
-------------------------
Number of members added 1
-------------------------
```

An additional step is required in case IPA 4.4 (RHEL 7.3) and later is used. It does not apply to IPA < 4.4:

```
$  ipa caacl-add-ca wildcard-hosts --cas ipa
  ACL name: wildcard-hosts
  Enabled: TRUE
  CAs: ipa
-------------------------
Number of members added 1
-------------------------
```

Finally create a CSR with subject CN=web01.example.com and issue the certificate. The process how to create the CSR is out of scope for this aricle but is explained in great detail in the Identity-Management guide:

Raw

```
$ ipa cert-request cert.csr --principal host/web01.example.com     --profile caWildcardCert
Issuing CA: ipa
  Certificate: MIIE[...]
  Subject: CN=*.web01.example.com,O=EXAMPLE.COM
  Issuer: CN=Certificate Authority,O=EXAMPLE.COM
  Not Before: Wed Jun 28 08:27:37 2017 UTC
  Not After: Sat Jun 29 08:27:37 2019 UTC
  Fingerprint (MD5): c2:02:6c:43:b5:81:53:9d:56:d6:a7:1c:58:28:db:ee
  Fingerprint (SHA1): c8:0a:0e:16:df:0e:33:bb:98:ed:74:bd:34:6f:f1:17:e4:71:a3:92
  Serial number: 18
  Serial number (hex): 0x12
```

The new certificate has been issued and you can find the certificate in the IdM database using the `ipa cert-find` command. To verify the new certificate is using a wildcard DNS name as Subject Common Name (CN) and also as Subject Alternative Name (SAN) extension you can use the following command:

Raw

```
$ ipa cert-show <cert-serial> --out cert.crt | openssl x509 -in cert.crt -noout -text|grep -e "Subject:" -e "DNS"
        Subject: O=EXAMPLE.COM, CN=*.web01.example.com
                DNS:*.web01.example.com, DNS:web01.example.com
```

Another way to issue the certificate is to tell certmonger to do the work for you. In this case the command needs to be executed on the host where you want to use the certificate. Here is a simple example. Please make sure to use the appropriate certmonger options to meet the requirements in your environment:

Raw

```
$ ipa-getcert request -d /etc/httpd/alias -p /etc/httpd/alias/pwdfile.txt -n wildcardCert -T caWildcardCert
New signing request "20170628084112" added.

$ getcert list -i 20170628084112
Number of certificates and requests being tracked: 9.
Request ID '20170628084112':
    status: MONITORING
    stuck: no
    key pair storage: type=NSSDB,location='/etc/httpd/alias',nickname='wildcardCert',token='NSS Certificate
DB',pinfile='/etc/httpd/alias/pwdfile.txt'
    certificate: type=NSSDB,location='/etc/httpd/alias',nickname='wildcardCert',token='NSS Certificate DB'
    CA: IPA
    issuer: CN=Certificate Authority,O=EXAMPLE.COM
    subject: CN=*.web01.example.com,O=EXAMPLE.COM
    expires: 2019-06-29 08:41:15 UTC
    dns: *.web01.example.com,web01.example.com
    key usage: digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment
    eku: id-kp-serverAuth,id-kp-clientAuth
    pre-save command:
    post-save command:
    track: yes
    auto-renew: yes
```

Root Cause

DNS name assertions via the CN field are deprecated for a long time already. Please check the details in RFC2818. Some modern client software finally started to remov CN name processing support. The Chrome browser (starting with v.58) is such an example.

Product(s) Red Hat Enterprise Linux **Category** Configure **Tags** certificates ipa

This solution is part of Red Hat's fast-track publication program, providing a huge library of solutions that Red Hat engineers have created while supporting our customers. To give you the knowledge you need the instant it becomes available, these articles may be presented in a raw and unedited form.