Support for multiple certificates and keys using LineRate

Ashok Mudukutore, 2014-15-12

Overview

Stronger Keys and Faster Security with ECC and Why ECC and PFS matter talk about Perfect Forward Secrecy (PFS) and the advantages of using ECC-based cryptography over RSA, both in terms of performance and security. As web servers transition to using ECC cipher suites over RSA, we need to be mindful that there are several clients that don't support ECC ciphers (for example, older versions of certain browsers, older mobile devices, etc.). Also, web sites have been slow to adopt new algorithms because they want to maintain support for legacy browsers that don't support the new algorithms. Even as late as 2012, out of 13 million TLS certificates found in a scan of the internet, fewer than 50 use an ECDSA key pair [1]. To handle all classes of clients, servers would need to deliver certificates with different cipher types based on the client's capabilities. LineRate helps customers deliver the most secure and compatible security encryption available by implementing ECC with RSA certificate fallback. Further, by using LineRate as a Reverse Proxy, a web service provider need not worry about ECC certificate compatibility with their web servers, as LineRate will take care of this for you in one, simple to manage solution. In this article, I'll show you how LineRate supports this use case by allowing multiple certificates of different cipher types to be associated with a single SSL profile. This allows the LineRate system to negotiate ECC certificates with clients that support ECC ciphers while falling back to RSA certificates for clients that only support RSA ciphers.

Configuration

The following set of articles provide an excellent tutorial on how to generate ECC keys and certificates, and configuring them on a LineRate for SSL offloading:

- Choosing ECC Curves and Preparing SSL Certificates
- Configuring SSL Offload on LineRate
- Confirming the Operation of SSL Offloading

In this section, I list the configuration that specifically addresses the use case where LineRate is used for SSL offloading and negotiating with clients using ECC or RSA certificates.

Note: The example configurations shown below use the LineRate CLI. The LineRate REST API provides a more programatic way to accomplish the same set of actions.

1. Add SSL certificates and keys

Configure certificates and private keys for ECC and RSA on a LineRate system using the **certificate** and **key** commands:

```
certificate cert_ecc
 pem-format
-----BEGIN CERTIFICATE-----
MIIBlzCCAT6gAwIBAgIJAJ38j2+cTN4rMAkGByqGSM49BAEwSzELMAkGA1UEBhMC
VVMxETAPBgNVBAgTCENvbG9yYWRvMREwDwYDVQQKEwhMaW5lUmF0ZTEWMBQGA1UE
AxMNbHJvcy10ZXN0LUludDAeFw0xNDEyMTYxMTE4MDBaFw0xNTAxMDUxMTE4MDBa
MEcxCzAJBgNVBAYTAlVTMREwDwYDVQQIEwhDb2xvcmFkbzERMA8GA1UEChMITGlu
ZVJhdGUxEjAQBgNVBAMTCWxyb3MtdGVzdDBZMBMGByqGSM49AgEGCCqGSM49AwEH
A0IABDqeCgA+8VuU/y2m6uF4Xebzo0v/Jn2FD27rEtFGhyLGklUf8C8jc3rz7ig/
0AH2jvXJeZyEmTEBLqyUfolohwujEDAOMAwGA1UdEwQFMAMBAf8wCQYHKoZIzj0E
AQNIADBFAiEAlZBFvZHHu+7de1aFL08Rx93ZLCTYlArqL4iPQGizxh0CIBZ0+hei
03MmNMREqTsvFLQc5BNEAq8O3CIZFhjEPzK3
-----END CERTIFICATE-----
quit
```

```
!
key key_ecc
 pem-format
-----BEGIN EC PARAMETERS-----
BggqhkjOPQMBBw==
-----END EC PARAMETERS-----
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIANFgtofG5wxCgRMKjE/j5RpdCCj2kDMf2JwOxfDJLcEoAoGCCqGSM49
AwEHoUQDQgAEOp4KAD7xW5T/Labq4Xhd5vOjS/8mfYUPbusS0UaHIsaSVR/wLyNz
evPuKD/QAfaO9cl5nISZMQEurJR+iWiHCw==
-----END EC PRIVATE KEY-----
quit
!
certificate cert_rsa
 pem-format
-----BEGIN CERTIFICATE-----
MIICIDCCAYmgAwIBAgIJAJ38j2+cTODTMA0GCSqGSIb3DQEBBQUAMEsxCzAJBgNV
BAYTAlVTMREwDwYDVQQIEwhDb2xvcmFkbzERMA8GA1UEChMITGluZVJhdGUxFjAU
BgNVBAMTDWxyb3MtdGVzdC1JbnQwHhcNMTQxMjE2MTExODAwWhcNMTUwMTA1MTEx
ODAwWjBHMQswCQYDVQQGEwJVUzERMA8GA1UECBMIQ29sb3JhZG8xETAPBgNVBAoT
CExpbmVSYXRlMRIwEAYDVQQDEwlscm9zLXRlc3QwgZ8wDQYJKoZIhvcNAQEBBQAD
gY0AMIGJAoGBAKq+j/vj0YlsufuCBOyCYJqDv2MDaag6BpkH0N6z280XjY4Ckahm
n8tEWH/AwODCIkTTYwGnw2BJn6woSLxlbant1U/dNN62B3IAwL+Ze4H76ZJqjofm
K9oTA+KPxs4+MLFKqCSqsaf0IDY+Xqs8QTZCXrfIvV527k91WuFf4eJ3AgMBAAGj
EDAOMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEASgrLLg7Fh7dd1X1W
u2KsUeu19q5m+8YEnRQoElA3cL/AquW8soFk4VjnYj2My1DChR0luCHeW0Uv5b9b
k0uWKjtMkk2b7aIlr5tudDvrgFFths01kdQ1/2zvnNRMWkQMkPhVKwJMm3Pc9cNW
e0b0E1f/RchR9U+HQjtED7pnaO4=
-----END CERTIFICATE-----
quit
!
key key_rsa
 pem-format
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQCqvo/749GJbLn7ggTsgmCag79jA2moOgaZB9Des9vNF42OApGo
Zp/LRFh/wMDgwiJE02MBp8NgSZ+sKEi8ZW2p7dVP3TTetgdyAMC/mXuB++mSao6H
5ivaEwPij8bOPjCxSqgkqrGn9CA2Pl6rPEE2Ql63yL1edu5PdVrhX+HidwIDAQAB
AoGANDk29tc2hq7vr6KT+Pbjrz7usc0gaujcjb/bPdKLPw6eKDpW7Mf+xgAwhVqi
Y9xc+Ooi1SXH11KNeO2VWbI4OQpshl3BhrSejRJbF+Z2pfVTBMH1zM0yk+pysIlO
emyUujRviplklM+QeHVhAfbVz4mwdV7RiCWjF2bn5vVlhSECQQDWE3giZJRIBjc1
CTHtSrQeTIkMinAgBYRSosgRewJsHB/BjB06jp21xLmkaGqiOk4EJdCYW21laqba
pLvsdETnAkEAzC6x4oiczXDxoWDOxf6OfrklJSAqylwC2lXnPgfoVo4DYGXfP2mF
oz8sK39PJm/oY9IoUaPklrctaap85FUz8QJBAMBgcp8Fn7Mx0r7YaoOBlGd+A9K/
PY+pJYZVPlFnJ4AbrfXygfmzW8qS5Pj31IvAtYdIrDGeR6rRsuvdCfutaJUCQQCH
X3IrCn5Dq11YrHGLTiG1nRUmjzQvpF9z2L6PHvkI4fEVKZWhWlnzCQBE+oxEpK+D
9yMqNappzpr6UsGpNWBRAkAtqkAEYEspd1R0b+KVFZFur8g4E3h/9bMLRqkRiHYj
7ROHbGhTNBbzu4lOGGy715WOz5/G3aotWZNZwRLIppUk
-----END RSA PRIVATE KEY-----
quit
```

2. Verify Certificates

Issue the 'show certificate brief' command to ensure the certificates we entered above were properly accepted by the system:

```
LROS# show certificate brief
Certificate                Subject Common Name (CN)
---------------------------------------------------------
cert_ecc                   lros-test
```

```
cert_rsa                    lros-test
```

3. Create SSL Profile

SSL profile entities on a LineRate system can have multiple certificate and key attachments. This is where the magic of attaching different cipher-types happens, allowing ECC encryption to be negotiated with clients that support ECC, and falling back to RSA encryption if needed. Of course, one can always use only one type of crypto, if desired. See documentation for additional details on SSL profiles.The configuration shown below does the following:

1. create an SSL profile entity
2. attach the ECC and RSA certificates and private keys configured in the previous step
3. set the list of allowed ciphers to "HIGH" (per openssl, these refer to ciphers with key lengths greater than or equal to 128 bits)

```
ssl profile ecc_rsa
 attach certificate cert_ecc
 attach certificate cert_rsa
 attach key key_ecc
 attach key key_rsa
 cipher-list openssl "HIGH"
```

4. Verify SSL Profile

Now that the LineRate system has been configured to support both ECC and RSA, let's take a look at how the system displays this on the SSL Profile:

```
LROS# show ssl profile ecc_rsa
Configuration:
  Primary Certificates:
    Name                    Origin          Type      Matching Key              In Effect
    cert_ecc                set locally     ECC       key_ecc                   Yes
    cert_rsa                set locally     RSA       key_rsa                   Yes
  Private Keys:
    Name                    Origin          Type      Matching Certificates
    key_ecc                 set locally     ECC       cert_ecc
    key_rsa                 set locally     RSA       cert_rsa
  Chained Cert Name:

  Disabled Protocols List:  SSLv2:SSLv3 default
  Cipher List:              HIGH        set locally
  ECC Curve List:           prime256v1  default
  SSL Session Cache Mode:   auto size   default
  SSL Session Cache Size:   10 Mi       default
  SSL Session Tickets Mode: enabled     default
  Active Protocols: TLSv1:TLSv1.1:TLSv1.2
  Ordered Cipher List:
    Name                          Certificate      Key
    ECDHE-ECDSA-AES256-SHA384     cert_ecc         key_ecc
    ECDHE-ECDSA-AES256-SHA        cert_ecc         key_ecc
    AES256-GCM-SHA384             cert_rsa         key_rsa
    AES256-SHA256                 cert_rsa         key_rsa
    AES256-SHA                    cert_rsa         key_rsa
    ECDHE-ECDSA-AES128-SHA256     cert_ecc         key_ecc
    ECDHE-ECDSA-AES128-SHA        cert_ecc         key_ecc
    AES128-GCM-SHA256             cert_rsa         key_rsa
    AES128-SHA256                 cert_rsa         key_rsa
    AES128-SHA                    cert_rsa         key_rsa
```

```
    ECDHE-ECDSA-DES-CBC3-SHA       cert_ecc         key_ecc
    DES-CBC3-SHA                   cert_rsa         key_rsa
```

The output of 'show ssl profile' points out some interesting details:

1. There is one ECC certificate / key pair in effect (cert_ecc, key_ecc).
2. There is one RSA certificate / key pair in effect (cert_rsa, key_rsa).
3. The ordered cipher list shows the ciphers supported by the SSL profile and the associated certificate / key pair. Note that all ciphers listed in there have an associated certificate / key pair.

5. Attach SSL Profile to Virtual IP

Configure SSL session termination on the LineRate system by attaching the SSL profile to a Virtual IP. The configuration shown below does the following:

1. create virtual IP
2. listen for secure HTTP requests on https://10.10.11.11:443 using the SSL profile that was previously configured

```
virtual-ip vip-offload
 service http
 ip address 10.10.11.11 443
 attach ssl profile ecc_rsa
 admin-status online
```

6. Other proxy configuration

Create a virtual-server that is associated with the virtual IP created in the previous step, and one or more real servers on the back end:

```
virtual-server vs-offload
 service http
 attach virtual-ip vip-offload default
 attach real-server ...
```

Note: The associated real-server configurations are not shown here since they need to be tailored to match the users environment. The LineRate documentation on Configuring Load Balancing provides detailed information on this.

Testing

Now we initiate an SSL connection to the LineRate system using openssl s_client to verify the following:

- negotiate ECC certificates when the client advertises its support for ECC ciphers
- negotiate RSA certificates when the client advertises its support for RSA ciphers
- disallow connection for unsupported ciphers (ie. unconfirmed on LineRate)

1. Verify clients using ECC

```
client-host:~$ openssl s_client -connect 10.10.11.11:443 -cipher ECDH < /dev/null | openssl x509 -noo
depth=0 C = US, ST = Colorado, O = LineRate, CN = lros-test
...
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 11384131667612196395 (0x9dfc8f6f9c4cde2b)
    Signature Algorithm: ecdsa-with-SHA1
        Issuer: C=US, ST=Colorado, O=LineRate, CN=lros-test-Int
        Validity
            Not Before: Dec 16 11:18:00 2014 GMT
```

```
            Not After : Jan  5 11:18:00 2015 GMT
        Subject: C=US, ST=Colorado, O=LineRate, CN=lros-test
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:3a:9e:0a:00:3e:f1:5b:94:ff:2d:a6:ea:e1:78:
                    5d:e6:f3:a3:4b:ff:26:7d:85:0f:6e:eb:12:d1:46:
                    87:22:c6:92:55:1f:f0:2f:23:73:7a:f3:ee:28:3f:
                    d0:01:f6:8e:f5:c9:79:9c:84:99:31:01:2e:ac:94:
                    7e:89:68:87:0b
                ASN1 OID: prime256v1
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: ecdsa-with-SHA1
         30:45:02:21:00:95:90:45:bd:91:c7:bb:ee:dd:7b:56:85:2f:
         4f:11:c7:dd:d9:2c:24:d8:94:0a:ea:2f:88:8f:40:68:b3:c6:
         1d:02:20:16:74:fa:17:a2:d3:73:26:34:c4:44:a9:3b:2f:14:
         b4:1c:e4:13:44:02:af:0e:dc:22:19:16:18:c4:3f:32:b7
```

2. Verify clients using RSA

```
client-host:~$ openssl s_client -connect 10.10.11.11:443 -cipher RSA < /dev/null | openssl x509 -noou
depth=0 C = US, ST = Colorado, O = LineRate, CN = lros-test
...
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 11384131667612197075 (0x9dfc8f6f9c4ce0d3)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=Colorado, O=LineRate, CN=lros-test-Int
        Validity
            Not Before: Dec 16 11:18:00 2014 GMT
            Not After : Jan  5 11:18:00 2015 GMT
        Subject: C=US, ST=Colorado, O=LineRate, CN=lros-test
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:aa:be:8f:fb:e3:d1:89:6c:b9:fb:82:04:ec:82:
                    60:9a:83:bf:63:03:69:a8:3a:06:99:07:d0:de:b3:
                    db:cd:17:8d:8e:02:91:a8:66:9f:cb:44:58:7f:c0:
                    c0:e0:c2:22:44:d3:63:01:a7:c3:60:49:9f:ac:28:
                    48:bc:65:6d:a9:ed:d5:4f:dd:34:de:b6:07:72:00:
                    c0:bf:99:7b:81:fb:e9:92:6a:8e:87:e6:2b:da:13:
                    03:e2:8f:c6:ce:3e:30:b1:4a:a8:24:aa:b1:a7:f4:
                    20:36:3e:5e:ab:3c:41:36:42:5e:b7:c8:bd:5e:76:
                    ee:4f:75:5a:e1:5f:e1:e2:77
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
         4a:0a:cb:2e:0e:c5:87:b7:5d:d5:7d:56:bb:62:ac:51:eb:b5:
         f6:ae:66:fb:c6:04:9d:14:28:12:50:37:70:bf:c0:aa:e5:bc:
```

```
        b2:81:64:e1:58:e7:62:3d:8c:cb:50:c2:85:1d:25:b8:21:de:
        5b:45:2f:e5:bf:5b:93:4b:96:2a:3b:4c:92:4d:9b:ed:a2:25:
        af:9b:6e:74:3b:eb:80:51:6d:86:cd:35:91:d4:35:ff:6c:ef:
        9c:d4:4c:5a:44:0c:90:f8:55:2b:02:4c:9b:73:dc:f5:c3:56:
        7b:46:f4:13:57:ff:45:c8:51:f5:4f:87:42:3b:44:0f:ba:67:
        68:ee
```

3. Verify failure when using unsupported cipher

```
client-host:~$ openssl s_client -connect 10.10.11.11:443 -cipher ECDH-RSA-AES128-GCM-SHA256 < /dev/nu
CONNECTED(00000003)
139905649854112:error:140740B5:SSL routines:SSL23_CLIENT_HELLO:no ciphers available:s23_clnt.c:469:
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 0 bytes and written 0 bytes
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
```

Availability

This feature was added as part of LineRate 2.5.0. If you'd like to try it out, LineRate offers a free tier that you can download today.

Ready to try LineRate? Visit https://linerate.f5.com/try
Want to learn more about LineRate? Visit https://linerate.f5.com/learn

References

- [1] Nick Sullivan: ECDSA: The digital signature algorithm of a better internet
- LineRate Documentation

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com