I’m tidying up Centralised RSyslog setup on the newly reinstalled becky Raspberry Pi system. One of the tasks at hand was to configure TLS based encrypted log shipping from my dedicated servers to home RSyslog server, this post shows the command and technique I use.

How To Check SSL Connection

What we do is run openssl command with the s_client option and specify remote server we’re testing connection to. It can be an HTTPS connection (port 443) to a website (will do a post about it some other time), but in my case I’m connecting to home office server becky.ts.fm with port 6514 (TLS encrypted port for Syslog):

```
root@s2:/ # openssl s_client -connect becky.ts.fm:6514
CONNECTED(00000003)
```
depth=0 CN = becky.ts.fm, O = Tech Stack Solutions, L = Dublin, C = IE
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 CN = becky.ts.fm, O = Tech Stack Solutions, L = Dublin, C = IE
verify error:num=21:unable to verify the first certificate
verify return:1
Certificate chain
  0 s:/CN=becky.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
   i:/CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
Server certificate
-----BEGIN CERTIFICATE-----
MIIEJDCCAoygAwIBAgIUJGqZcuyXa7ekrK+U8yfB2Cu54FYwDQYJKoZIhvcNAQEL
jMNHiZ0zdzolHWzkV6iKc20MxOv3ftQ1TsE7vg+/Z2fTSv2f0uirPZUPe65wzr
75
ABRIDGED
9n7UHknn7/mV+llclo0A8oyXB5zeVf+lxLufVRyhEIpLFVtriu0Go6PW0gjwMo
PM
QB/0E6WgtSDMf43f9qzSdtKNgHFw1MpxVQdULSahnI6n0gpfuUIvKDBmBazgh6
lR
RtZqUqz09pE=
-----END CERTIFICATE-----
subject=/CN=becky.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
issuer=/CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
Acceptable client certificate CA names
   /CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
Client Certificate Types: RSA sign, DSA sign, ECDSA sign
   Requested Signature Algorithms:
      RSA+SHA256:0x09+0x08:0x04+0x08:ECDSA+SHA256:0x07+0x08:
      RSA+SHA384:0x0A+0x08:0x05+0x08:ECDSA+SHA384:RSA+SHA512:
      0x0B+0x08:0x06+0x08:ECDSA+SHA512:RSA+SHA1:ECDSA+SHA1
   Shared Requested Signature Algorithms:
      RSA+SHA256:ECDSA+SHA256:RSA+SHA384:ECDSA+SHA384:RSA+SHA512:
      ECDSA+SHA512:RSA+SHA1:ECDSA+SHA1
Peer signing digest: SHA512
Server Temp Key: ECDH, P-256, 256 bits
SSL handshake has read 1704 bytes and written 427 bytes
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
  Protocol : TLSv1.2
  Cipher    : ECDHE-RSA-AES256-GCM-SHA384
  Session-ID:
    224B0D3C5183426D7DDAD5A5FB361BC9C5175EC9CB0AA6A2F396DAAEE7178080
  Session-ID-ctx:
  Master-Key:
    6BE67A8AD4E22029DE1B3D0DE1F4351FD0488AB1D8ABC7E25187
    Key-Arg : None
    Krb5 Principal: None
    PSK identity: None
    PSK identity hint: None
    Start Time: 1583790681
    Timeout : 300 (sec)
    Verify return code: 21 (unable to verify the first certificate)

As you can see, there is a problem. I’m checking the last line in the output, which should return code ok, but it tells me that my s2 server can’t verify the first (only) certificate presented by my RSyslog server becky.ts.fm.

From experience, I know that’s because s2 somehow needs to acknowledge the certificate authority (CA) that issued certificate for becky.ts.fm.

How Successful TLS Connection Looks in OpenSSL
So if I specify this CA cert as a command line option, openssl will establish TLS connection and confirm code ok:

Now let’s specify the CA certificate I used for RSyslog, the connection and certificates verification should work just fine now:

```
root@s2:/ # openssl s_client -CAfile /etc/rsyslog.d/ca.crt -connect becky.ts.fm:6514
CONNECTED(00000003)
  depth=1 CN = syslog.ts.fm, O = Tech Stack Solutions, L = Dublin, C = IE
  verify return:1
  depth=0 CN = becky.ts.fm, O = Tech Stack Solutions, L = Dublin, C = IE
  verify return:1
Certificate chain
  0 s:/CN=becky.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
       i:/CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
Server certificate
    -----BEGIN CERTIFICATE-----
    MIIEJDCCAoygAwIBAgIUJGqZcuyXa7ekrK+U8yfB2Cu54FYwDQYJKoZIhvcNAQEL
    BQAwVDEVMBMGA1UEAxMMc3lzbg9nLnRzLmZtMR0wGwYDVQQKEyxRUZWNoIFN0YW
    hLy23Z4sAABIDIGHJMNHiZ0zdzo1HWzkm6iKc20Mx0v3ftQ1TsE7vg+/Z2fTSv2f0uirPZUPegSzwr
    75
    9n7UHkn7r/mV+llclo0A8oyXB5zeVf+lxLufVRyhEIpLFVtRiu0Go6PW0gjwMo
    PM
    QB/0E6WgtSDMf43f9qzSdtKnGhFw1MpxVQdULSaben6n0gpfuUIvKDBmBazgh6
    lR
    RtZqUqz09pE=
    -----END CERTIFICATE-----
    subject=/CN=becky.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
    issuer=/CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
```

Acceptable client certificate CA names

```
/CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
```
Client Certificate Types: RSA sign, DSA sign, ECDSA sign
Requested Signature Algorithms:
RSA+SHA256:0x09+0x08:0x04+0x08:ECDSA+SHA256:0x07+0x08:RSA+SHA384:
0x0A+0x08:0x05+0x08:ECDSA+SHA384:RSA+SHA512:0x0B+0x08:
0x06+0x08:ECDSA+SHA512:RSA+SHA1:ECDSA+SHA1
Shared Requested Signature Algorithms:
RSA+SHA256:ECDSA+SHA256:RSA+SHA384:ECDSA+SHA384:RSA+SHA512:
ECDSA+SHA512:RSA+SHA1:ECDSA+SHA1
Peer signing digest: SHA512
Server Temp Key: ECDH, P-256, 256 bits
SSL handshake has read 1704 bytes and written 427 bytes
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
Protocol : TLSv1.2
Cipher : ECDHE-RSA-AES256-GCM-SHA384
Session-ID:
C6797515EEA312D7A9EC6685F895AE004798550FF70619E85F24AB5ACF80F0A9
Session-ID-ctx:
Master-Key:
4B84DF3CFE9697EEC634DC271B2A490D94B7A7AB1CA218F016B1ED141FA1479C
Key-Arg : None
Krb5 Principal: None
PSK identity: None
PSK identity hint: None
Start Time: 1583790782
Timeout : 300 (sec)
Verify return code: 0 (ok)
^C

That’s it – this means secure connection establishes successfully, so I can enjoy secure log shipping from s2 to becky.ts.fm.
See Also

- Centralised RSyslog Server
- RSyslog with grc