1
ANS:
Without specifying a CA file: $ openssl s_client -connect twitter.com:443 -CApath empty_dir verify error:num=20:unable to get local issuer certificate depth=0 CN = twitter.com depth=1 C = US, O = Let's Encrypt, CN = E6

With specifying a CA file: $ openssl s_client -connect twitter.com:443 -CAfile ca-certificates.crt
depth=2 C = US, O = Internet Security Research Group, CN = ISRG Root X1 depth=1 C = US, O = Let's Encrypt, CN = E6 depth=0 CN = twitter.com Verification: OK

- -CAfile tells OpenSSL which root CAs to trust.
- Without it, OpenSSL has no way to verify the certificate chain → verification error.
- With it, OpenSSL can validate the chain → trusted connection.

2
ANS
at the first command: told openssl to connect to twitter.com:443 but to only trust certificates in empty_dir. Since that directory was empty, OpenSSL had no trusted root CAs to verify the certificate chain against.

verify error:num=20:unable to get local issuer certificate
Verify return code: 20 (unable to get local issuer certificate)

- The server (twitter.com) sent its certificate chain:
  - twitter.com (leaf cert, signed by Let’s Encrypt E6)
  - E6 intermediate (signed by ISRG Root X1)
- Normally, OpenSSL would look in its trust store (e.g., ca-certificates.crt) to find the ISRG Root X1, which is the trusted root CA.
- But because you pointed it to an empty CA directory, OpenSSL had nothing to anchor the chain of trust → it couldn’t verify that E6 really leads back to a trusted root.

3
ANS
$ openssl x509 -in twitter_com.cert -text

Certificate: Data: Version: 3 (0x2) Serial Number:
06:53:96:80:57:c6:c6:dc:f5:31:36:db:c5:c4:9f:bc:9e:a8 Signature Algorithm: ecdsa-with-SHA384 Issuer: C = US, O = Let's Encrypt, CN = E6 Validity Not Before: Aug 19 20:42:10 2025 GMT

Not After : Nov 17 20:42:09 2025 GMT Subject: CN = twitter.com Subject Public Key Info: Public Key Algorithm: id-ecPublicKey Public-Key: (256 bit) pub: 04:ba:5c:dc:0e:2c:0f:e4:16:59:54:06:42:42:c0: 8a:7c:a1:8c:cd:1e:46:89:76:88:6a:ea:dd:41:8e: 3b:aa:6e:3b:d3:48:2a:a2:e4:51:97:d9:d0:b3:52: a9:3e:d7:4b:2e:d0:cf:27:64:0e:f7:7c:77:b3:cf: 59:f5:51:03:a6 ASN1 OID: prime256v1 NIST CURVE: P-256 X509v3 extensions: X509v3 Key Usage: critical Digital Signature X509v3 Extended Key Usage: TLS Web Server Authentication, TLS Web Client Authentication X509v3 Basic Constraints: critical CA:FALSE X509v3 Subject Key Identifier: 7B:A9:AE:D2:07:70:3A:3D:71:04:96:D1:F1:1F:F9:B0:4B:6E:DC:F1 X509v3 Authority Key Identifier: 93:27:46:98:03:A9:51:68:8E:98:D6:C4:42:48:DB:23:BF:58:94:D2 Authority Information Access: CA Issuers - URI:http://e6.i.lencr.org/ X509v3 Subject Alternative Name: DNS:*.twitter.com, DNS:cdn.syndication.twitter.com, DNS:twitter.com X509v3 Certificate Policies: Policy: 2.23.140.1.2.1 X509v3 CRL Distribution Points: Full Name: URI:http://e6.c.lencr.org/41.crl CT Precertificate SCTs: Signed Certificate Timestamp: Version : v1 (0x0) Log ID : CC:FB:0F:6A:85:71:09:65:FE:95:9B:53:CE:E9:B2:7C: 22:E9:85:5C:0D:97:8D:B6:A9:7E:54:C0:FE:4C:0D:B0 Timestamp : Aug 19 21:40:40.445 2025 GMT Extensions: none Signature : ecdsa-with-SHA256 30:45:02:21:00:F5:28:54:C5:A8:E1:8A:DB:77:A5:BA: 57:33:94:64:A5:8C:AE:CB:ED:D3:31:44:79:4A:DF:53: 33:ED:5F:51:7C:02:20:48:3B:37:FF:BB:81:DC:FA:72: D7:45:A5:DF:B1:30:9C:87:EE:FD:65:A7:69:6F:ED:3A: E8:21:7C:81:B4:ED:6A Signed Certificate Timestamp: Version : v1 (0x0) Log ID : DD:DC:CA:34:95:D7:E1:16:05:E7:95:32:FA:C7:9F:F8: 3D:1C:50:DF:DB:00:3A:14:12:76:0A:2C:AC:BB:C8:2A Timestamp : Aug 19 21:40:40.459 2025 GMT Extensions: none Signature : ecdsa-with-SHA256 30:45:02:21:00:80:63:29:29:82:54:B0:7E:DF:0E:77: 6A:6C:DB:55:C4:DF:95:2A:10:20:CA:F1:FD:78:C7:DF: D2:F9:35:B1:A4:02:20:1A:EA:F7:5F:17:77:E4:96:47: ED:CC:75:DA:5B:A6:9C:7A:58:4F:38:2C:4E:84:92:26: E6:8A:92:F6:A3:49:CF Signature Algorithm: ecdsa-with-SHA384 Signature Value: 30:66:02:31:00:ec:b1:01:b6:01:df:d1:8e:8c:fd:4c:a7:01: e1:45:b7:27:ac:ba:d3:77:57:57:d0:f8:2c:89:8d:e8:b2:6a: f5:14:61:17:28:58:ff:d5:7e:f4:61:c6:fc:f1:04:a5:57:02: 31:00:a4:98:d5:1b:42:25:94:15:d9:59:95:20:d3:0d:94:69: 60:aa:c9:98:f1:9b:7a:9e:25:5d:fa:4f:f7:56:35:30:97:db: 58:17:a4:6e:a1:fd:a7:44:14:c9:3d:0c:6a:aa -----BEGIN CERTIFICATE-----

MIIDsDCCAzWgAwIBAgISBlOWgFfGxtz1MTbbxcSfvJ6oMAoGCCqGSM49BAMDMDIx
CzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MQswCQYDVQQDEwJF
NjAeFw0yNTA4MTkyMDQyMTBaFw0yNTExMTcyMDQyMDlaMBYxFDASBgNVBAMTC3R3
aXR0ZXIuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEulzcDiwP5BZZVAZC
QsCKfKGMzR5GiXaIaurdQY47qm4700gqouRRl9nQs1KpPtdLLtDPJ2QO93x3s89Z
9VEDpqOCAkUwggJBMA4GA1UdDwEB/wQEAwIHgDAdBgNVHSUEFjAUBggrBgEFBQcD
AQYIKwYBBQUHAwIwDAYDVR0TAQH/BAIwADAdBgNVHQ4EFgQUe6mu0gdwOj1xBJbR
8R/5sEtu3PEwHwYDVR0jBBgwFoAUkydGmAOpUWiOmNbEQkjbI79YlNIwMgYIKwYB
BQUHAQEEJjAkMCIGCCsGAQUFBzAChhZodHRwOi8vZTYuaS5sZW5jci5vcmcvMEIG

A1UdEQQ7MDmCDSoudHdpdHRlci5jb22CG2Nkbi5zeW5kaWNhdGlvbi50d2l0dGVy
LmNvbYILdHdpdHRlci5jb20wEwYDVR0gBAwwCjAIBgZngQwBAgEwLQYDVR0fBCYw
JDAioCCgHoYcaHR0cDovL2U2LmMubGVuY3Iub3JnLzQxLmNybDCCAQQGCisGAQQB
1nkCBAIEgfUEgfIA8AB2AMz7D2qFcQll/pWbU87psnwi6YVcDZeNtql+VMD+TA2w
AAABmMRG8X0AAAQDAEcwRQIhAPUoVMWo4Yrbd6W6VzOUZKWMrsvt0zFEeUrfUzPt
X1F8AiBIOzf/u4Hc+nLXRaXfsTCch+79Zadpb+066CF8gbTtagB2AN3cyjSV1+EW
BeeVMvrHn/g9HFDf2wA6FBJ2Ciysu8gqAAABmMRG8YsAAAQDAEcwRQIhAIBjKSmC
VLB+3w53amzbVcTflSoQIMrx/XjH39L5NbGkAiAa6vdfF3fklkftzHXaW6acelhP
OCxOhJIm5oqS9qNJzzAKBggqhkjOPQQDAwNpADBmAjEA7LEBtgHf0Y6M/UynAeFF
tyesutN3V1fQ+CyJjeiyavUUYRcoWP/VfvRhxvzxBKVXAjEApJjVG0IllBXZWZUg
0w2UaWCqyZjxm3qeJV36T/dWNTCX21gXpG6h/adEFMk9DGqq -----END CERTIFICATE-----

Structure of an X.509 Certificate

1. Version
2. Serial Number
3. Signature Algorithm
4. Issuer
5. Validity
6. Subject
7. Subject Public Key Info
8. Extensions (X.509 v3 extensions)
9. Signature Algorithm (again)
10. Signature Value

4
ANS
Yes — there is an intermediate certificate.
In twitter_com.cert output, under Issuer
Issuer: C = US, O = Let's Encrypt, CN = E6
That means the certificate for twitter.com was signed by Let's Encrypt E6 (an intermediate CA), not directly by the root CA.

The intermediate acts as a bridge between the server certificate and the trusted root CA. Its purpose is to improve security (keeping the root CA offline), provide flexibility in certificate issuance, and allow revocation without affecting the root.

5
ANS
From twitter_com.cert
Subject (server cert): CN = twitter.com
Issuer: C = US, O = Let's Encrypt, CN = E6
That means:

1. twitter.com certificate was issued by the Let's Encrypt E6 intermediate.
2. The E6 intermediate itself is issued by a root CA (e.g. “ISRG Root X2”, depending on the chain).

Yes, there (is) an intermediate CA (Let’s Encrypt E6). However, in this chain all CAs (intermediate and root) belong to the same organization (ISRG / Let’s Encrypt). So while there are multiple certificates, they are not from different organizations.

6
ANS
ca-certificates.crt provides the list of trusted root CAs that OpenSSL use to verify certificate chains.

7
ANS just count "BEGIN CERTIFICATE" or "END CERTIFICATE"

```
theepob@BBIdea3:/etc/ssl/certs$ grep -c "BEGIN CERTIFICATE" /etc/ssl/certs/c
a-certificates.crt
146
```

8
$ openssl x509 -in cert1.pem -text -noout
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 6828503384748696800 (0x5ec3b7a6437fa4e0)
Signature Algorithm: sha1WithRSAEncryption
Issuer: CN = ACCVRAIZ1, OU = PKIACCV, O = ACCV, C = ES
Validity
Not Before: May 5 09:37:37 2011 GMT
Not After : Dec 31 09:37:37 2030 GMT
Subject: CN = ACCVRAIZ1, OU = PKIACCV, O = ACCV, C = ES
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (4096 bit)
Modulus:
00:9b:a9:ab:bf:61:4a:97:af:2f:97:66:9a:74:5f: d0:d9:96:fd:cf:e2:e4:66:ef:1f:1f:47:33:c2:44: a3:df:9a:de:1f:b5:54:dd:15:7c:69:35:11:6f:bb: c8:0c:8e:6a:18:1e:d8:8f:d9:16:bc:10:48:36:5c: f0:63:b3:90:5a:5c:24:37:d7:a3:d6:cb:09:71:b9: f1:01:72:84:b0:7d:db:4d:80:cd:fc:d3:6f:c9:f8: da:b6:0e:82:d2:45:85:a8:1b:68:a8:3d:e8:f4:44: 6c:bd:a1:c2:cb:03:be:8c:3e:13:00:84:df:4a:48: c0:e3:22:0a:e8:e9:37:a7:18:4c:b1:09:0d:23:56: 7f:04:4d:d9:17:84:18:a5:c8:da:40:94:73:eb:ce: 0e:57:3c:03:81:3a:9d:0a:a1:57:43:69:ac:57:6d: 79:90:78:e5:b5:b4:3b:d8:bc:4c:8d:28:a1:a7:a3: a7:ba:02:4e:25:d1:2a:ae:ed:ae:03:22:b8:6b:20: 0f:30:28:54:95:7f:e0:ee:ce:0a:66:9d:d1:40:2d: 6e:22:af:9d:1a:c1:05:19:d2:6f:c0:f2:9f:f8:7b: b3:02:42:fb:50:a9:1d:2d:93:0f:23:ab:c6:c1:0f:

92:ff:d0:a2:15:f5:53:09:71:1c:ff:45:13:84:e6: 26:5e:f8:e0:88:1c:0a:fc:16:b6:a8:73:06:b8:f0:
63:84:02:a0:c6:5a:ec:e7:74:df:70:ae:a3:83:25: ea:d6:c7:97:87:93:a7:c6:8a:8a:33:97:60:37:10:
3e:97:3e:6e:29:15:d6:a1:0f:d1:88:2c:12:9f:6f: aa:a4:c6:42:eb:41:a2:e3:95:43:d3:01:85:6d:8e:
bb:3b:f3:23:36:c7:fe:3b:e0:a1:25:07:48:ab:c9: 89:74:ff:08:8f:80:bf:c0:96:65:f3:ee:ec:4b:68:
bd:9d:88:c3:31:b3:40:f1:e8:cf:f6:38:bb:9c:e4: d1:7f:d4:e5:58:9b:7c:fa:d4:f3:0e:9b:75:91:e4:
ba:52:2e:19:7e:d1:f5:cd:5a:19:fc:ba:06:f6:fb: 52:a8:4b:99:04:dd:f8:f9:b4:8b:50:a3:4e:62:89:
f0:87:24:fa:83:42:c1:87:fa:d5:2d:29:2a:5a:71: 7a:64:6a:d7:27:60:63:0d:db:ce:49:f5:8d:1f:90:
89:32:17:f8:73:43:b8:d2:5a:93:86:61:d6:e1:75: 0a:ea:79:66:76:88:4f:71:eb:04:25:d6:0a:5a:7a:
93:e5:b9:4b:17:40:0f:b1:b6:b9:f5:de:4f:dc:e0: b3:ac:3b:11:70:60:84:4a:43:6e:99:20:c0:29:71:
0a:c0:65 Exponent: 65537 (0x10001)
X509v3 extensions:
Authority Information Access:
CA Issuers - URI:http://www.accv.es/fileadmin/Archivos/ certificados/raizaccv1.crt
OCSP - URI:http://ocsp.accv.es
X509v3 Subject Key Identifier:
D2:87:B4:E3:DF:37:27:93:55:F6:56:EA:81:E5:36:CC:8C:1E:3F:BD
X509v3 Basic Constraints: critical
CA:TRUE
X509v3 Authority Key Identifier:
D2:87:B4:E3:DF:37:27:93:55:F6:56:EA:81:E5:36:CC:8C:1E:3F:BD
X509v3 Certificate Policies:
Policy: X509v3 Any Policy
User Notice:
Explicit Text:
CPS: http://www.accv.es/legislacion_c.htm
X509v3 CRL Distribution Points:
Full Name:
URI:http://www.accv.es/fileadmin/Archivos/certificados/ raizaccv1_der.crl
X509v3 Key Usage: critical
Certificate Sign, CRL Sign
X509v3 Subject Alternative Name:
email:accv@accv.es
Signature Algorithm: sha1WithRSAEncryption
Signature Value:
97:31:02:9f:e7:fd:43:67:48:44:14:e4:29:87:ed:4c:28:66:
d0:8f:35:da:4d:61:b7:4a:97:4d:b5:db:90:e0:05:2e:0e:c6:
79:d0:f2:97:69:0f:bd:04:47:d9:be:db:b5:29:da:9b:d9:ae:
a9:99:d5:d3:3c:30:93:f5:8d:a1:a8:fc:06:8d:44:f4:ca:16:
95:7c:33:dc:62:8b:a8:37:f8:27:d8:09:2d:1b:ef:c8:14:27:
20:a9:64:44:ff:2e:d6:75:aa:6c:4d:60:40:19:49:43:54:63:
da:e2:cc:ba:66:e5:4f:44:7a:5b:d9:6a:81:2b:40:d5:7f:f9:
01:27:58:2c:c8:ed:48:91:7c:3f:a6:00:cf:c4:29:73:11:36:

de:86:19:3e:9d:ee:19:8a:1b:d5:b0:ed:8e:3d:9c:2a:c0:0d:
d8:3d:66:e3:3c:0d:bd:d5:94:5c:e2:e2:a7:35:1b:04:00:f6:
3f:5a:8d:ea:43:bd:5f:89:1d:a9:c1:b0:cc:99:e2:4d:00:0a:
da:c9:27:5b:e7:13:90:5c:e4:f5:33:a2:55:6d:dc:e0:09:4d:
2f:b1:26:5b:27:75:00:09:c4:62:77:29:08:5f:9e:59:ac:b6:
7e:ad:9f:54:30:22:03:c1:1e:71:64:fe:f9:38:0a:96:18:dd:
02:14:ac:23:cb:06:1c:1e:a4:7d:8d:0d:de:27:41:e8:ad:da:
15:b7:b0:23:dd:2b:a8:d3:da:25:87:ed:e8:55:44:4d:88:f4:
36:7e:84:9a:78:ac:f7:0e:56:49:0e:d6:33:25:d6:84:50:42:
6c:20:12:1d:2a:d5:be:bc:f2:70:81:a4:70:60:be:05:b5:9b:
9e:04:44:be:61:23:ac:e9:a5:24:8c:11:80:94:5a:a2:a2:b9:
49:d2:c1:dc:d1:a7:ed:31:11:2c:9e:19:a6:ee:e1:55:e1:c0:
ea:cf:0d:84:e4:17:b7:a2:7c:a5:de:55:25:06:ee:cc:c0:87:
5c:40:da:cc:95:3f:55:e0:35:c7:b8:84:be:b4:5d:cd:7a:83:
01:72:ee:87:e6:5f:1d:ae:b5:85:c6:26:df:e6:c1:9a:e9:1e:
02:47:9f:2a:a8:6d:a9:5b:cf:ec:45:77:7f:98:27:9a:32:5d:
2a:e3:84:ee:c5:98:66:2f:96:20:1d:dd:d8:c3:27:d7:b0:f9:
fe:d9:7d:cd:d0:9f:8f:0b:14:58:51:9f:2f:8b:c3:38:2d:de:
e8:8f:d6:8d:87:a4:f5:56:43:16:99:2c:f4:a4:56:b4:34:b8:
61:37:c9:c2:58:80:1b:a0:97:a1:fc:59:8d:e9:11:f6:d1:0f: 4b:55:34:46:2a:8b:86:3b

Issuer = Subject (CN = ACCVRAIZ1, OU = PKIACCV, O = ACCV, C = ES) → self-signed.

Root certificate: **เซ็นด้วยตัวเอง** (self-signed)**และเป็นที่เชื่อถือโดยเบราว์เซอร์**/OS
Intermediate certificate: Issuer = Root CA, Subject ≠ Issuer
Twitter server certificate: Issuer = Intermediate CA, Subject = twitter.com

In [3]:
```python
from OpenSSL import crypto
import pem
def verifyAll(cert_pairs):
    for domain, target_path, intermediate_path in cert_pairs:
        with open(target_path, 'r') as cert_file:
            cert = cert_file.read()

        with open(intermediate_path, 'r') as int_cert_file:
            int_cert = int_cert_file.read()

        pems = pem.parse_file('./ca-certificates.cert')
        trusted_certs = []
        for mypem in pems:
            trusted_certs.append(str(mypem))

        trusted_certs.append(int_cert)

        verified = verify_chain_of_trust(cert, trusted_certs)

        if verified:
            print(domain + ' Certificate verified')
```

```
        else:
            print(domain + ' Certificate verification failed')


def verify_chain_of_trust(cert_pem, trusted_cert_pems):

    certificate = crypto.load_certificate(crypto.FILETYPE_PEM, cert_pem)

    store = crypto.X509Store()
    for trusted_cert_pem in trusted_cert_pems:
        trusted_cert = crypto.load_certificate(crypto.FILETYPE_PEM,trusted_cert_pem
        store.add_cert(trusted_cert)

    store_ctx = crypto.X509StoreContext(store, certificate)

    result = store_ctx.verify_certificate()

    if result is None:
        return True
    else:
        return False
```

In [4]:
```
verifyAll([
    ('twitter.com', './twitter.cert', './twitter_intermediate.cert'),
    ('google.com', './google.cert', './google_intermediate.cert'),
    ('chula.ac.th', './chula.cert', './chula_intermediate.cert'),
    ('classdeedee.cloud.cp.eng.chula.ac.th', './classdeedee.cert', './classdeedee_i
    ])
```

```
twitter.com Certificate verified
google.com Certificate verified
chula.ac.th Certificate verified
classdeedee.cloud.cp.eng.chula.ac.th Certificate verified
```

11

ANS

Class 1:

- low assurance
- low-risk uses where only domain control is required. Examples: basic website TLS for non-sensitive sites
- cheaper, automated issuance, fewer organizational checks.

Class 3:

- high assurance
- high-risk use cases requiring stronger identity assurance and auditing.
- stronger vetting of the organization
- stricter issuance policies, increased auditing, often HSM protection for keys and higher warranties.

12a

ANS

Possible attacks

- Attacker can issue valid TLS certs for any domain and perform undetectable MitM (inspect/modify traffic).
- They can sign malware/code, forge S/MIME emails, create client auth certs to impersonate users, or mint more intermediates to spread trust.

12b

ANS

No, not reliably. CRL/OCSP only help after detection/revocation. A compromised root can issue fresh valid certs that aren’t revoked yet.