TLS with Certificate, Private Key and Pass Phrase

Asked 4 years, 2 months ago Modified 4 years, 2 months ago Viewed 5k times

0

I am integrating with a bank server which has provided me with the certificates. I create a `pem` file out of the certificates, so now I have the Certificates, Private Key in a `pem` file and the Pass Phrase for the key separately.

The newly generated file `pem` is working for making an SSL connection using the OpenSSL command as follows:

```
openssl s_client -connect host:port -key key.pem -cert cert.pem
```

This command requests for the passphrase and I am able to connect. But I am not able to connect to the same using my Go code, which looks like this:

```
package main

import (
    "crypto/tls"
    "crypto/x509"
    "fmt"
    "net/http"
)

func main() {
    caCert := []byte(`certs pem data`) // this contains both private key and
certificates
    caCertPool := x509.NewCertPool()
    caCertPool.AppendCertsFromPEM(caCert)

    // Setup HTTPS client
    tlsConfig := &tls.Config{
        RootCAs:            caCertPool,
        InsecureSkipVerify: true,
    }
    tlsConfig.BuildNameToCertificate()
    transport := &http.Transport{TLSClientConfig: tlsConfig}
```

ssl go openssl rsa tls1.2

Share Improve this question Follow

edited May 14, 2019 at 11:50
Jonathan Hall
74.5k 15 140 186

asked May 14, 2019 at 11:36
Sumit Agarwal
4,081 8 32 49

Your PEM data is apparently invalid. But since we can't see what data you're using, it's pretty difficult to validate/debug. – Jonathan Hall May 14, 2019 at 11:51

but I am using same pem data for my openssl command which work fine for making the connection – Sumit Agarwal May 14, 2019 at 11:52

Also I read that it pulls system cert pools, is that the case? I haven't added these certs to my mac keychain – Sumit Agarwal May 14, 2019 at 11:53

1 Answer

Sorted by: Highest score (default)

9

You seem to be confusing certificate authorities with client certificates. Client certificates prove to the server that you are who you say you are (much like a username and password would), and CAs are used so that you know that you're talking to the correct server.

Judging from the openssl command that works for you, your bank gave you a client certificate and key (although that is highly unusal; no one except yourself should ever hold your private key and especially the passphrase).

The `tls.Config.Certificates` field, if used by a client, is used to configure client certificates.

> Certificates contains one or more certificate chains to present to the other side of the connection. [...] Clients doing client-authentication may set either Certificates or GetClientCertificate.

```
import (
    "crypto/tls"
    "crypto/x509"
    "encoding/pem"
    "fmt"
    "log"
    "net/http"
)

var bundle = []byte(`
-----BEGIN EC PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,99586A658F5D2DAC4A8A3CA387CF71CE

25EtKb7ycOI/5R47fYwpiaNERgYnCxCtcrMXJuOgueuxUXjiU0n93hpUpIQqaTLH
dDKhsR1UHvGJVTV4h577RQ+nEJ5z8K5Y9NWFqzfa/Q5SY43kqqoJ/fS/OCnTmH48
z4bL/dJBDE/a5HwJINgqQhGi9iUkCWUiPQxriJQ0i2s=
-----END EC PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIB2TCCAX+gAwIBAgIUUTZvgwwnbC05WHgIHMXxrbZzr6wwCgYIKoZIzj0EAwIw
QjELMAkGA1UEBhMCWFgxFTATBgNVBAcMDERlZmF1bHQgQ2l0eTEcMBoGA1UECgwT
RGVmYXVsdCBDb21wYW55IEx0ZDAeFw0xOTA1MTQxMzAwMDJaFw0xOTA1MTUxMzAw
MDJaMEIxCzAJBgNVBAYTAlhYMRUwEwYDVQQHDAxEZWZhdWx0IENpdHkxHDAaBgNV
BAoME0RlZmF1bHQgQ29tcGFueSBMdGQwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNC
AAScgLGx6SXchEo/s0X3AoF0mQkh3bGf9QY0s/2dPqf3/9irwz35DiDGoaP+FDZv
HnUX+D3tUEPhxkLyzWKKT9HHo1MwUTAdBgNVHQ4EFgQU3eB8oRcmvzZrx9Dkb6ma
MMtu1MkwHwYDVR0jBBgwFoAU3eB8oRcmvzZrx9Dkb6maMMtu1MkwDwYDVR0TAQH/
BAUwAwEB/zAKBggqhkjOPQQDAgNIADBFAiAvw/FqAmGbSlBklp6AHJy9kf9VPyhe
RA93ccNQ+7m1fAIhAOXr8c2QsH2oOYRTbn6bPZjkYQ2jLMaxatKhChBIuyZA
-----END CERTIFICATE-----
`)

func main() {
    keyBlock, certsPEM := pem.Decode(bundle)

    fmt.Println(x509.IsEncryptedPEMBlock(keyBlock)) // Output: true

    // Decrypt key
    keyDER, err := x509.DecryptPEMBlock(keyBlock, []byte("foobar"))
    if err != nil {
        log.Fatal(err)
    }

    // Update keyBlock with the plaintext bytes and clear the now obsolete
    // headers.
    keyBlock.Bytes = keyDER
    keyBlock.Headers = nil
```

```
        }
    }
```

Share Improve this answer Follow

answered May 14, 2019 at 13:03

Peter

29.2k 5 48 60

I've seen DecryptPEMBlock is deprecated since go 1.16. Is there any alternative to this? I really appreciate your answer. – Gudari Jan 29 at 19:59

1 @Urko, you have no choice but to decrypt with DecryptPEMBlock initially, but after that you can re-encrypt with any authenticated cipher, such as AES-GCM. – Peter Jan 29 at 20:39

Hi Peter, I've decided to create a project to be able to decrypt certificate using openssl. Take a look please and give me your opinion. Kind regards gitea.urkob.com/urko/go-grpc-certificate/src/branch/main/pkg/... – Gudari Mar 6 at 16:50