



Securing the Web

Mastering HTTPS for Developers

PRAIRIE DEV CON

WEB | DEV | CLOUD | AI



Lotlinx

online
business systems



EXCELLENCE IN RECRUITMENT



Survey link:





RFC 2818: HTTP Over TLS

Conceptually, **HTTP/TLS** is very simple. Simply use **HTTP over TLS** precisely as you would use **HTTP over TCP**.

Thanks!



Keep in touch

✉ aj@bitovi.com

in /softwarebyaj/



bitovi

Agenda

- 
- **Introduction**
 - **Theory of HTTPS**
 - **Practicing HTTPS**
 - **Conclusion**



Introduction

AJ Wiebe

- I like coding
- I really like coding
- Mostly Frontend
- Consultant at Bitovi
- Principal for Fortune 500 Client
- Core Internal Libraries
- Architecture & Direction
- Husband, Father

For Whom,

Want to make sense of the HTTPS landscape and feel empowered to bring apps to production not relying on software as a service

Not for you if...

Authentication / Authorization

HTTPS is for privacy during communication not user Auth(z)

Mathematical Cryptography

Only science here is applied science

Selling My Library

I don't get money if you use HTTPS

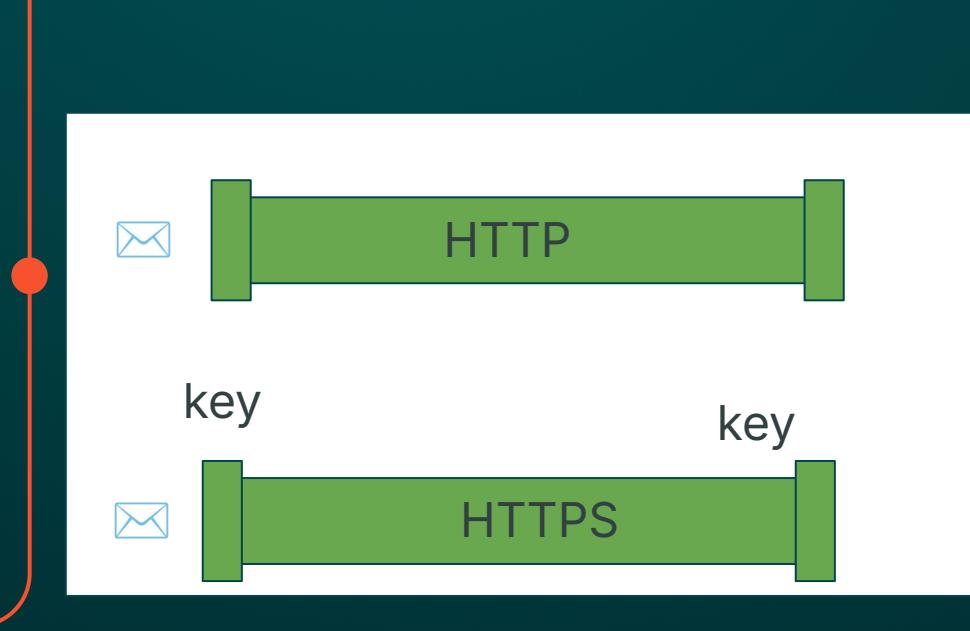
I'm not here to shill some library

HTTPS

Definitely someone else's problem

Until...

It wasn't



HTTP + Secure

Just say what the acronym is



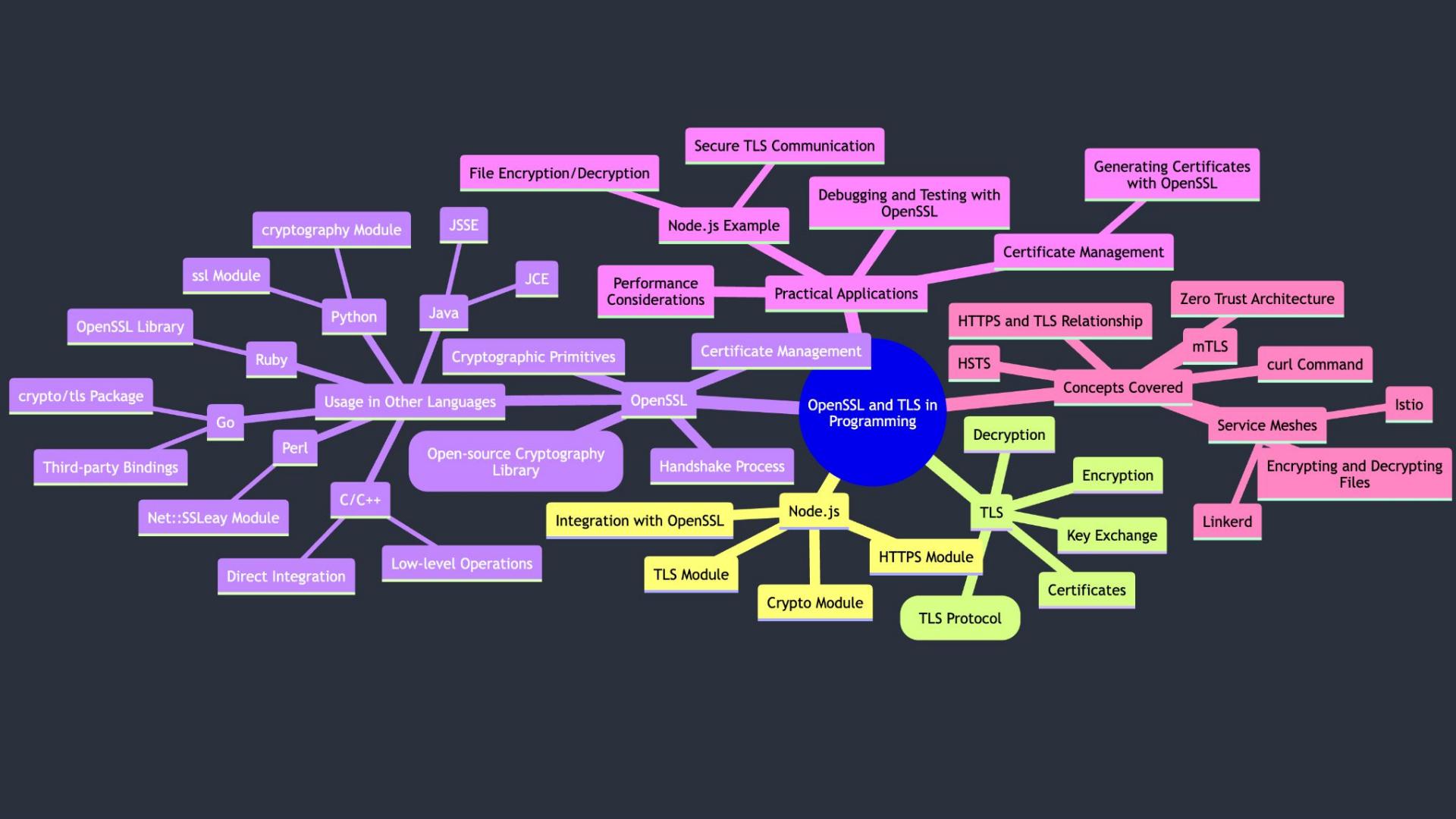
Key and ...or cert?

Think back to that one youtube video you saw 8 years ago



Buy what?

Maybe the person at the keystore can tell me how...



I'm just a
dev –
looking at
an
endpoint
asking if I
should use
HTTPS... for
localhost



Testing Security Features

HSTS requires HTTPS



Consistency

Local match dev, test, and prod



Avoid Mixed Content Issues

If there's already some https apis



Consistency

Local match dev, test, and prod



I'm just a
dev –
looking at
an
endpoint
asking if I
should use
HTTPS... for
localhost



Testing Security Features

HSTS requires HTTPS



Consistency

Local match dev, test, and prod



Avoid Mixed Content Issues

If there's already some https apis



External API

Most external APIs will be HTTPS



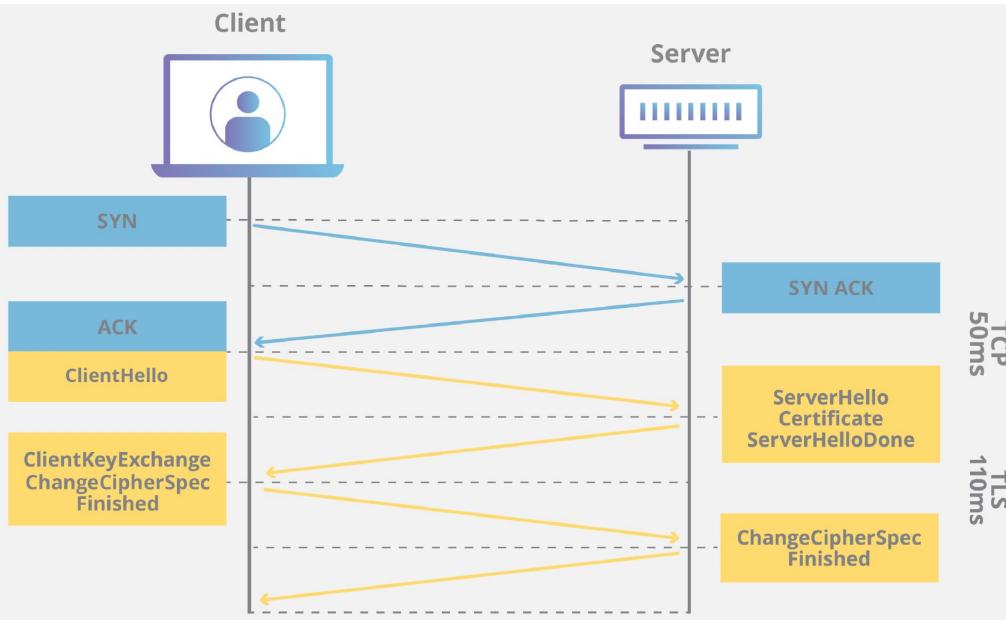


Theory of HTTPS

Theory of HTTPS

- **Scaffold out topic**
- **HTTPS process**
- **Cryptography intro**
- **Common tools**
- **Vulnerabilities**

HTTPS Process: Part 1 TLS Handshake



Kirk Bater
@KirkBater

Follow

This image is a TCP/IP Joke. This tweet is a UDP joke. I don't care if you get it.

Thread

iamkirkbater and jkjustjoshing

iamkirkbater Aug 23rd, 2017 at 9:37 AM
in #www

Do you want to hear a joke about TCP/IP?



7 replies

- Jkjustjoshing 5 months ago
Yes, I'd like to hear a joke about TCP/IP
- iamkirkbater 5 months ago
Are you ready to hear the joke about TCP/IP?
- Jkjustjoshing 5 months ago
I am ready to hear the joke about TCP/IP
- iamkirkbater 5 months ago
Here is a joke about TCP/IP.
- iamkirkbater 5 months ago
Did you receive the joke about TCP/IP?
- Jkjustjoshing 5 months ago
I have received the joke about TCP/IP.
- iamkirkbater 5 months ago
Excellent. You have received the joke about TCP/IP. Goodbye.

HTTPS Process: Part 2 HTTP Req/Resp

1. Client sends request
(note: sent over TLS using symmetric session key)
2. Server processes
3. Server responds
(note: sent over TLS using symmetric session key)
4. Client decrypts
5. Optional: Session Resumption

Cryptography intro: Symmetric vs Asymmetric Encryption

In Genral:

Asymmetric:

Public key: share-able, encrypts data

Private key: not share-able, decrypts data

Symmetric:

Shared key: encrypts/decrypts data

Cryptography intro: Symmetric vs Asymmetric Encryption

Applied to HTTPS

Asymmetric:

Securely exchange a secret (symmetric key) between client/server

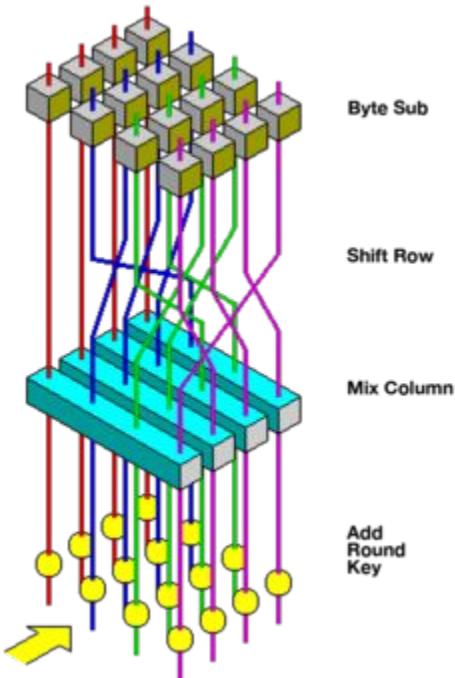
RSA, Diffie-Hellman, Elliptic Curve

Symmetric:

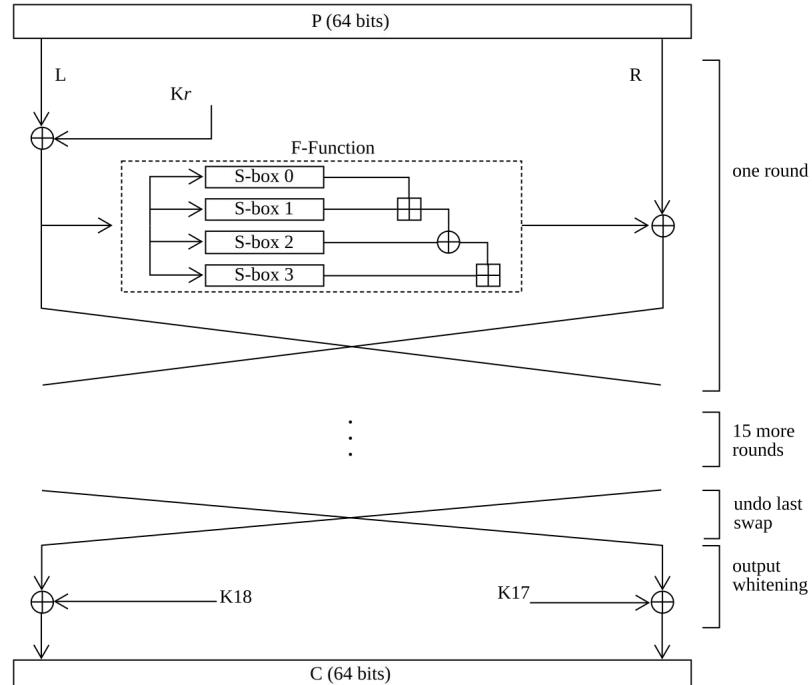
Shared key: encrypts/decrypts data

- Data Encryption Standard (DES) and Triple DES (3DES) 
- Advanced Encryption Standard (AES) 
- Twofish 
- Blowfish 

AES / Blowfish Diagrams



AES



P =Plaintext; C =Ciphertext; K_x = P -array-entry x
 \oplus = xor \boxplus = addition mod 2^{32}

Blowfish

Cryptography intro: Hashing

- Crypto = Hash(Crypto, Hash(Crypto))
- Hash function create a hash
 - data of fixed length based on input variable length data
- Small change to input result in large change to hash

```
› npm view @angular/cli

@angular/cli@18.2.4 | MIT | deps: 17 | versions: 829
CLI tool for Angular
https://github.com/angular/angular-cli

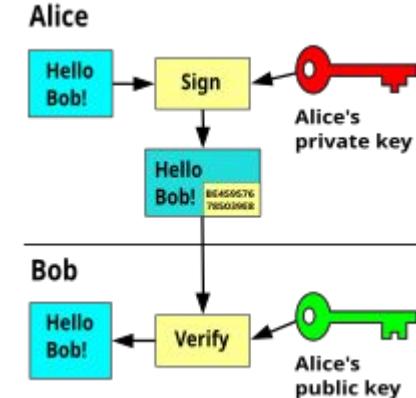
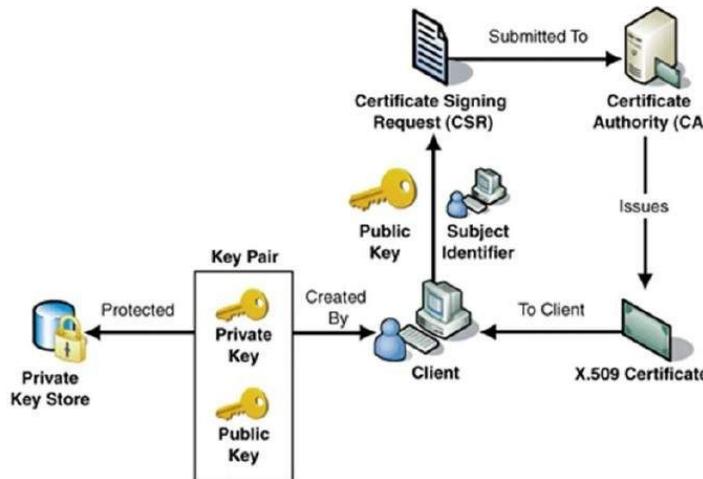
keywords: Angular CLI, Angular DevKit, angular, angular-cli, devkit, sdk
bin: ng

dist
.tarball: https://registry.npmjs.org/@angular/cli/-/cli-18.2.4.tgz
.shasum: a68fe4c606dd31cf2f49a29f6894533ec8589fb2
.integrity: sha512-n+Y2xlgcpTZ+MZmycf2b3ceVvANDJFkDEodobVtyG63WvG0hkZ3aGhT7sHguKpAQwJLicSf8zF2z+v1Yi0DvRw==
.unpackedSize: 634.8 kB
```

NPM
Hash

Cryptography intro: Digital Signatures

- When setting up HTTPS you get a digital certificate from a trusted Certificate Authority (CA)
- Cert has server's key
- CA signs the server's cert using the CA's private key
- CA distributes its public key



Cryptography intro: TLS / SSL

TLS 1.3 RFC

The 's' in https

Email (SMTP, IMAP, POP3) can use TLS

DNS Queries can use TLS

FTPS (Secure FTP) uses TLS

RFC 8446

TLS

August 2018

The "extension_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),

    /* Legacy algorithms */
    rsa_pkcs1_sha1(0x0201),
    ecdsa_sha1(0x0203),

    /* Reserved Code Points */
    private_use(0xFE00..0xFFFF),
    (0xFFFF)
} SignatureScheme;

struct {
    SignatureScheme supported_signature_algorithms<2..2^16-2>;
} SignatureSchemeList;
```

Note: This enum is named "SignatureScheme" because there is already "SignatureAlgorithm" type in TLS 1.2, which this replaces. We use the term "signature algorithm" throughout the text.

Common Tools: OpenSSL

- Alternatives
 - Java Secure Socket Extension JSSE (Java Keytool)
 - Dotnet System.Security.Cryptography Namespace
 - Golang crypto/tls
- C/C++, Python, Nodejs, Ruby and more all use OpenSSL
- You've probably already used openssl... even if you don't remember
- ```
openssl req -x509 -newkey rsa:2048 -nodes -sha256 -subj
'/CN=localhost' -keyout private-key.pem -out certificate.pem
```
- ```
openssl pkcs12 -certpbe AES-256-CBC -export -out test_cert.pfx -inkey  
private-key.pem -in certificate.pem -passout pass:sample
```

Common Tools: OpenSSL

- `openssl req -x509 -newkey rsa:2048 -nodes -sha256 -subj '/CN=localhost' -keyout private-key.pem -out certificate.pem`
- Generates a self signed X.509 cert
- **openssl req**: starts the OpenSSL command to create certificate signing requests (CSR).
- **-x509**: This indicates that you're creating a self-signed certificate instead of a CSR. X.509 is the standard format for public key certificates.
- **-newkey rsa:2048**: This generates a new key along with the certificate. It creates a 2048-bit RSA key pair.
- **-nodes**: This flag means "**no DES**" (Data Encryption Standard) and ensures that the private key will not be encrypted. The private key will be stored in plain text (unprotected).
- **-sha256**: Specifies that the certificate should be signed using the SHA-256 hashing algorithm (more secure than older algorithms like MD5 or SHA-1).
- **-subj '/CN=localhost'**: specifies the subject (identity information) for the certificate CN=localhost sets the Common Name (CN) field to "localhost" (usually the hostname).
- **-keyout private-key.pem**: This writes the private key to the specified file (private-key.pem).
- **-out certificate.pem**: This writes the self-signed certificate to the specified file

Common Tools: OpenSSL

- `openssl pkcs12 -certpbe AES-256-CBC -export -out test_cert.pfx -inkey private-key.pem -in certificate.pem -passout pass:sample`
- packages private key and certificate into a PKCS#12 file (.pfx or .p12)
- **openssl pkcs12**: This starts the OpenSSL command to work with PKCS#12 files.
- **-certpbe AES-256-CBC**: Specifies the encryption algorithm (AES-256-CBC)
- **AES-256-CBC**: AES (Advanced Encryption Standard) with a 256-bit key and Cipher Block Chaining (CBC) mode for encrypting the certificate.
- **-export**: specifies that you are exporting the key and certificate into a PKCS#12 file.
- **-out test_cert.pfx**: the name of the PKCS#12 output file (test_cert.pfx).
- **-inkey private-key.pem**: the input private key file (private-key.pem).
- **-in certificate.pem**: the input certificate file (certificate.pem).
- **-passout pass:sample**: the password to protect the .pfx file. In this case, the password is set to sample. This is needed to protect the private key within the PKCS#12 bundle.

Common Tools: Certificate Authorities

- <https://letsencrypt.org/>
- <https://www.digicert.com/>
- [https://wwwentrustcom/](https://wwwentrustcom)
- <https://www.comodo.com/>
- Your cloud provider (Azure, AWS, Google Cloud etc)
- Already mentioned the CA purpose
 - Facilitate the digital signature

Common Tools: Curl / Postman / Rest Client

- curl CLI to execute requests (https, imap, ftp, telnet, wss)
- Postman
- Rest Client

The screenshot shows the Postman application interface. At the top, there are tabs for 'Cats ex' (selected), 'GET Cats C', 'POST Cre', and 'dev'. Below the tabs is a toolbar with icons for Save, Edit, and Chat. The main area shows a 'POST' request to 'http://localhost:8081/prairie-dev-con'. The 'Headers' tab is selected, showing '(8)' entries. The 'Body' tab has a green dot indicating it's selected. Below the body tab, there are sections for 'Query Params' and 'Params'. The 'Response' section at the bottom contains a cartoon illustration of an astronaut launching a rocket.

The screenshot shows a code editor window titled 'example.http'. It contains a series of API requests in a specific format:

```
1 GET https://api.github.com/users/Huachao
2 ###
3 GET https://api.github.com/repos/Huachao/vscode-restclient HTTP/1.1
4 ###
5 POST http://requestb.in/tcdce4tc?k=a%20bc
6 Date: {$guid}
7
8 {
9   "key": "{$guid}",
10   "key2": "{$timestamp - 1 d}",
11   "key3": "{$guid}",
12   "key4": "{$Guid}"
13   #rest
14   "key5": "{$timestampfasdfa}",
15   "key6": "{$}",
16   "key7": "{$(d)}",
17   "key8": "{$(timestamp)}",
18   "key9": "{$randomInt 1 19}"
19   "key10": "{$randomInt 21 19}",
20 }
```

At the bottom of the editor, there are status indicators: 0 0 0 1188ms. The bottom right corner shows file details: Ln 6, Col 1, Spaces: 4, UTF-8, CR LF, and HTTP.

Common Tools: Wireshark

380	13.076516	192.168.128.23	224.0.0.251	MDNS	395	Standard query response 0x0000 PTR Chrome...
381	13.214084	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0x28f867cf...
382	13.419427	192.168.145.80	224.0.0.251	MDNS	206	Standard query 0x0000 ANY Chris' iPhone...
383	13.419428	192.168.148.47	224.0.0.251	MDNS	83	Standard query 0x0000 PTR _oculusal_sp._...
384	13.521830	fe80::44c4:97e1:b6...	ff02::fb	MDNS	103	Standard query 0x0000 PTR _oculusal_sp._...
385	13.521831	192.168.148.47	224.0.0.251	MDNS	293	Standard query response 0x0000 PTR Mango...
386	13.521832	fe80::44c4:97e1:b6...	ff02::fb	MDNS	313	Standard query response 0x0000 PTR Mango...
387	13.521833	fe80::cdf:152c:ab5...	ff02::fb	MDNS	226	Standard query 0x0000 ANY Chris' iPhone...
388	13.523022	192.168.128.23	224.0.0.251	MDNS	395	Standard query response 0x0000 PTR Chrome...
389	13.523023	192.168.128.23	224.0.0.251	MDNS	395	Standard query response 0x0000 PTR Chrome...
390	13.576628	104.192.142.14	192.168.145.107	TLSv1...	219	Application Data
391	13.576844	192.168.145.107	104.192.142.14	TCP	66	61297 -> 443 [ACK] Seq=618 Ack=962 Win=20...
392	13.577467	192.168.145.107	192.168.65.2	TCP	176	49319 -> 8009 [PSH, ACK] Seq=221 Ack=225 V...

```
> Frame 382: 206 bytes on wire (1648 bits), 206 bytes captured (1648 bits) on interface en0, id 0
> Ethernet II, Src: 72:d6:3f:a1:ef:87 (72:d6:3f:a1:ef:87), Dst: IPv4mcast_fb (01:00:5e:00:00:fb)
> Internet Protocol Version 4, Src: 192.168.145.80, Dst: 224.0.0.251
> User Datagram Protocol, Src Port: 5353, Dst Port: 5353
< Multicast Domain Name System (query)
  > Transaction ID: 0x0000
    > Flags: 0x0000 Standard query
      Questions: 2
      Answer RRs: 0
      Authority RRs: 3
      Additional RRs: 1
    < Queries
      > Chris' iPhone._rdlink._tcp.local: type ANY, class IN, "QU" question
      > Chris-iPhone.local: type ANY, class IN, "QU" question
      > Authoritative nameservers
      > Additional records
```

```
0000 01 00 5e 00 00 fb 72 d6 3f a1 ef 87 08 00 45 00 .^.r.?....E.
0010 00 c0 2d 12 00 00 ff 11 5b 26 c0 a8 91 50 e0 00 .-. [&...P...
0020 00 fb 14 e9 14 e9 00 ac f2 31 00 00 00 00 02 .....1.....
0030 00 00 00 03 00 01 0f 43 68 72 69 73 e2 80 99 20 ..C hris...
0040 69 50 68 6f 6e 65 07 5f 72 64 6c 69 6e 6b 04 5f iPhone._rdlink.
0050 74 63 70 05 6c 6f 63 61 6c 00 00 ff 80 01 0c 43 _tcp.local....C
0060 68 72 69 73 2d 69 50 68 6f 6e 65 c0 29 00 ff 80 hris-iPh one...
0070 01 c0 0c 00 21 00 01 00 00 00 78 00 08 00 00 00 .....!....x...
0080 00 c0 02 c0 34 c0 34 00 1c 00 01 00 00 00 78 00 .....4-4....x...
0090 10 fe 80 00 00 00 00 00 00 c0 df 15 2c 0a b5 a0 .....4.....x...
00a0 19 c0 34 00 01 00 01 00 00 00 78 00 04 c0 a8 91 .....4.....x...
00b0 50 00 00 29 05 a0 00 00 11 94 00 12 00 04 00 0e P...). .....
00c0 00 ab 3a 2f 92 02 70 8e 72 d6 3f a1 ef 87 ..:/..p.r?...
```

Common Tools: Web Server Configuration

- More on this in the next section

HTTPS Vulnerabilities

- OWASP
- Outdated cryptography
- HTTPS “man in the middle” attack
 - Does your google cert look like→

Certificate Viewer: *.google.com

General Details

Issued To

Common Name (CN)	*.google.com
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	WR2
Organization (O)	Google Trust Services
Organizational Unit (OU)	<Not Part Of Certificate>

Validity Period

Issued On	Monday, August 12, 2024 at 12:33:49 AM
Expires On	Monday, November 4, 2024 at 12:33:48 AM

SHA-256 Fingerprints

Certificate	a51995e61988f1c2fa9cdf7ecf8a5657909bae15547c3ef3e10d4e 462341a34e
Public Key	f60473d34377c4a094027fde23d4efb19deba0a36101d6c3ed7a 3e314dcca142



Practicing HTTPS

Practicing HTTPS



Smallest example

Deploy nodejs app to azure



Custom certificates

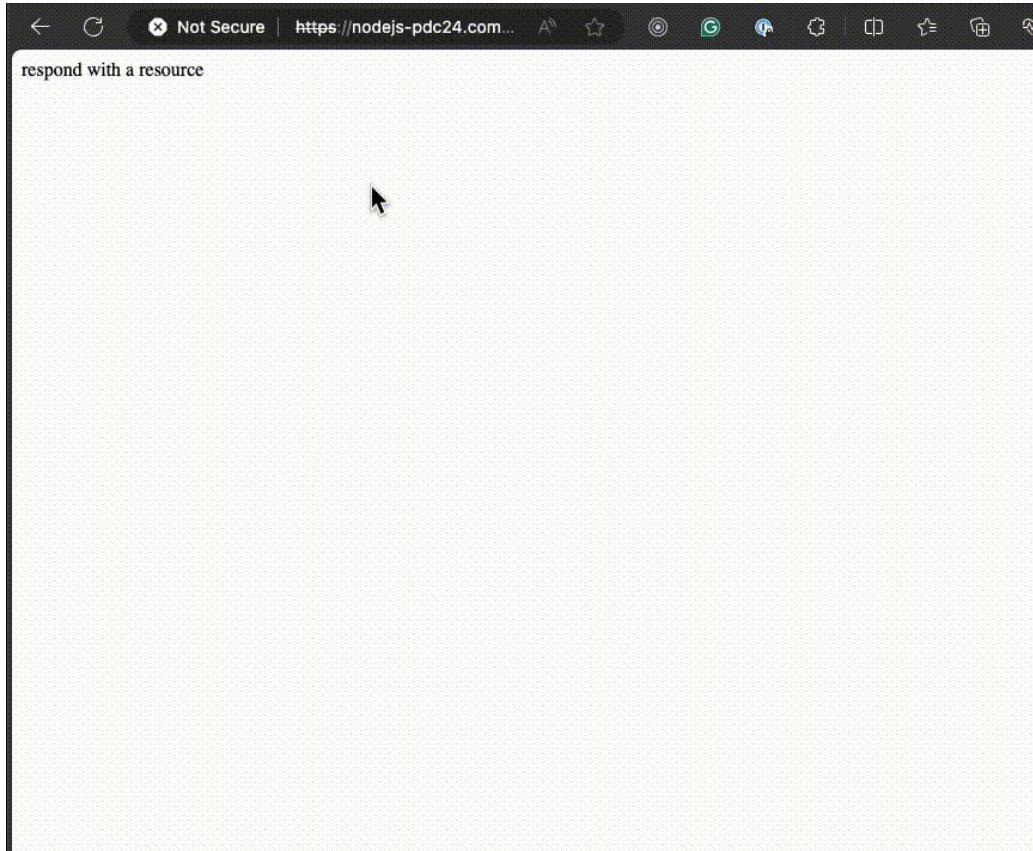
Same as above with custom cert



Consuming custom certificates

Issues and resolutions

How to get a website's certificate



Certificate Viewer: *.azurewebsites.net

X

General

Details

Issued To

Common Name (CN) *.azurewebsites.net
Organization (O) Microsoft Corporation
Organizational Unit (OU) <Not Part Of Certificate>

Issued By

Common Name (CN) Microsoft Azure RSA TLS Issuing CA 08
Organization (O) Microsoft Corporation
Organizational Unit (OU) <Not Part Of Certificate>

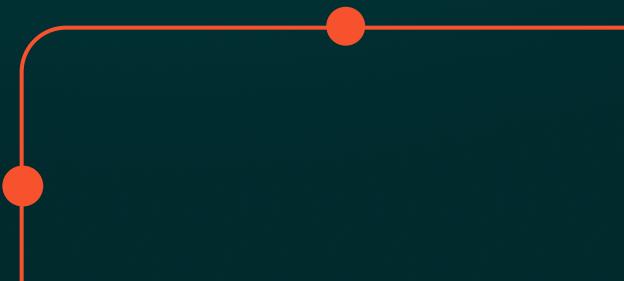
Validity Period

Issued On Friday, May 24, 2024 at 1:32:00 AM
Expires On Monday, May 19, 2025 at 1:32:00 AM

SHA-256 Fingerprints

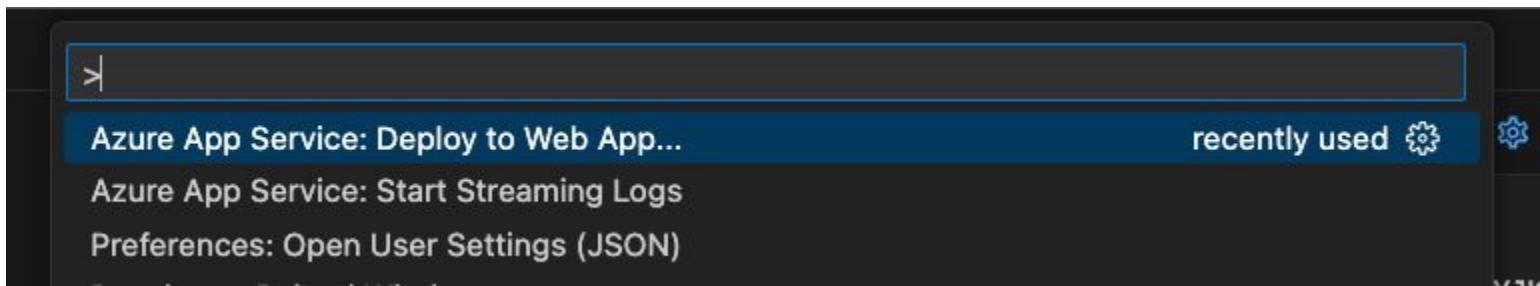
Certificate 15a44592bb0f16e6c196cc910dcb7e9b6461bdc34fea245134ae
465e011855f4
Public Key f3a468bdedec0906bd680e6cab7a8ccb1e0688264dd3575fac9
5e50c5180de2d

Smallest example



Smallest Example

- Create a new Express app
 - `npx express-generator myExpressApp --view ejs`
- Deploy it to Azure (free tier)
- ???
- Profit





Certificate Viewer: *.azurewebsites.net

X

General

Details

Issued To

Common Name (CN) *.azurewebsites.net
Organization (O) Microsoft Corporation
Organizational Unit (OU) <Not Part Of Certificate>

Issued By

Common Name (CN) Microsoft Azure RSA TLS Issuing CA 08
Organization (O) Microsoft Corporation
Organizational Unit (OU) <Not Part Of Certificate>

Validity Period

Issued On Friday, May 24, 2024 at 1:32:00 AM
Expires On Monday, May 19, 2025 at 1:32:00 AM

SHA-256 Fingerprints

Certificate 15a44592bb0f16e6c196cc910dcb7e9b6461bdc34fea245134ae
465e011855f4
Public Key f3a468bdedec0906bd680e6cab7a8ccb1e0688264dd3575fac9
5e50c5180de2d

General**Details****Issued To**

Common Name (CN) nodejs-pdc24.com
Organization (O) My Company
Organizational Unit (OU) <Not Part Of Certificate>

Issued By

Common Name (CN) nodejs-pdc24.com
Organization (O) My Company
Organizational Unit (OU) <Not Part Of Certificate>

Validity Period

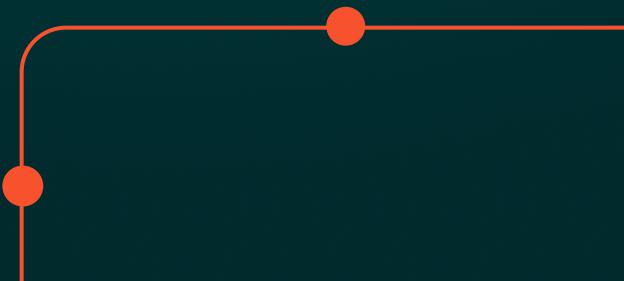
Issued On Tuesday, September 17, 2024 at 4:18:42 PM
Expires On Wednesday, September 17, 2025 at 4:18:42 PM

SHA-256 FINGERPRINTS

Certificate a8f04e34c0d872fb106466c45d00eb1bd3f071e5c71394fe95dd0ab7a2e0c42
Public Key 4ecc50936a5606ef4daac4c1e62941a7ea41c3b0b04e6d0b91c96af8b4d579a8

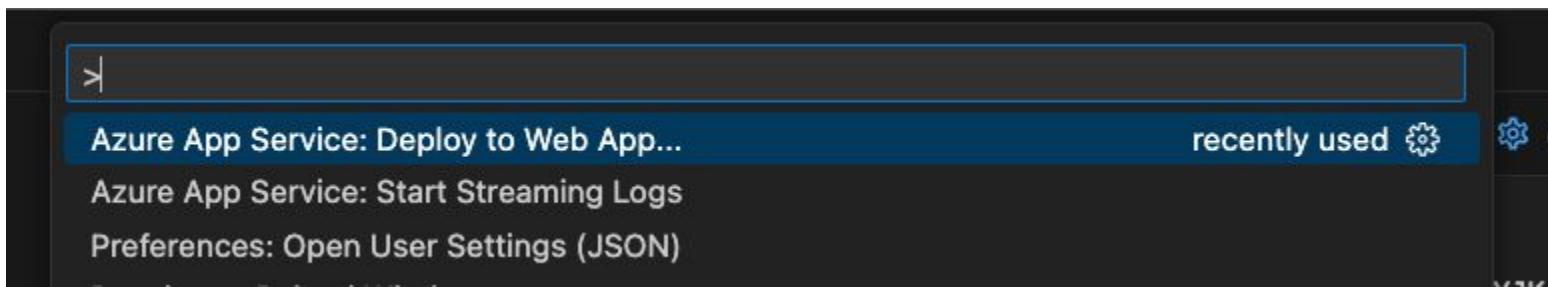
Custom Certificates

How to add your own certificate



Custom Certificates

- Create a new Express app
 - `npx express-generator myExpressApp2 --view ejs`
- Deploy it to Azure (not so free tier)
- Get a custom domain
- Create / acquire cert
- Upload cert
- Apply to custom domain



Custom Certificates

- Get a custom domain

The screenshot shows the Azure portal interface for managing custom domains. At the top, there are refresh and troubleshoot buttons. A message bar indicates that one or more domains are not secured and provides a link to learn more. Below this, there is a section to configure and manage custom domains assigned to the app. It shows fields for IP address and Custom Domain Verification ID, both of which are redacted. There are also filter and add filter buttons. The main table displays three items:

Custom domains	Status	Solution	Binding type	Certificate used	Actions
<input type="checkbox"/> www.nodejs-pdc24.com	✖ No binding	Add binding	-	-	...
<input type="checkbox"/> nodejs-pdc24.com	✓ Secured	-	SNI SSL	appsvc_linux_centralus_basic-Cent...	...
azure-nodejs-pdc24-selfcert.azur...	✓ Secured	-	-	-	...

Custom Certificates

- Create / Require certs

```
openssl req -x509 -newkey  
rsa:2048 -nodes -sha256 -keyout  
private-key.pem -out  
certificate.pem -days 365
```

-config openssl.cnf

Refresh Troubleshoot Send us your feedback

Managed certificates Bring your own certificates (.pfx) Public key certificates (.cer)

Private key certificates (.pfx) can be used for TLS/SSL bindings and can be loaded to the certificate store for your app to consume. To understand how to load the certificates for your app to consume click on the learn more link. [Learn more](#)

Filter by keywords Add filter

3 items

Add certificate Delete

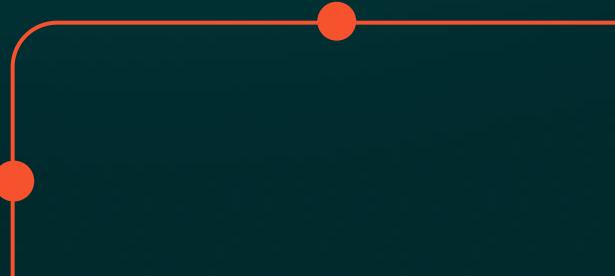
Certificate	Status	Domain	Certificate Name	Expiration	Thumbprint
<input type="checkbox"/>	⚠ Expiring soon	localhost	appsvc_linux_centralus_basic-CentralUSw...	2024-10-15	0100340796F62C
<input type="checkbox"/>	⚠ Expiring soon	nodejs-pdc24.com	appsvc_linux_centralus_basic-CentralUSw...	2024-10-17	1CD68ABE6106B
<input checked="" type="checkbox"/>	✓ No action needed	nodejs-pdc24.com	appsvc_linux_centralus_basic-CentralUSw...	2025-09-17	101448A4233FF8

```
openssl.cnf  
1 [ req ]  
2 default_bits = 2048  
3 distinguished_name = req_distinguished_name  
4 req_extensions = req_ext  
5 x509_extensions = v3_ca # The extensions to add to the self-signed cert  
6  
7 [ req_distinguished_name ]  
8 countryName = Country Name (2 letter code)  
9 countryName_default = US  
10 stateOrProvinceName = State or Province Name (full name)  
11 stateOrProvinceName_default = California  
12 localityName = Locality Name (eg, city)  
13 localityName_default = San Francisco  
14 organizationName = Organization Name (eg, company)  
15 organizationName_default = My Company  
16 commonName = Common Name (e.g. server FQDN or YOUR name)  
17 commonName_default = nodejs-pdc24.com  
18 commonName_max = 64  
19  
20 [ req_ext ]  
21 subjectAltName = @alt_names  
22  
23 [ alt_names ]  
24 DNS.1 = nodejs-pdc24.com  
25  
26 [ v3_ca ]  
27 subjectKeyIdentifier=hash  
28 authorityKeyIdentifier=keyid(always,issuer)  
29 basicConstraints = CA:FALSE  
30 keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment  
31 extendedKeyUsage = serverAuth
```

A large, red, stylized silhouette of a dog's head and upper body, facing left. The dog has a dark blue collar with a small tag. The background is a solid dark teal.

Consuming custom certificates

With nest.js



Nest to Smallest

```
15  getSmallest(): Observable<string> [] {
16    return this.httpService
17      .get('https://azure-nodejs-pdc24.azurewebsites.net/users')
18      .pipe(
19        map(({ data }) => {
20          return data;
21        }),
22        catchError(error: AxiosError) => {
23          console.error(error);
24          throw new Error('Error fetching data');
25        }),
26      );
27    }
28 }
```

It just works, thanks microsoft

Nest to Custom Cert

```
29  getCustom(): Observable<string> {
30    return this.httpService.get('https://nodejs-pdc24.com/users').pipe(
31      map(({ data }) => {
32        return data;
33      }),
34      catchError((error: AxiosError) => {
35        console.error(error);
36        throw new Error('Error fetching data');
37      }),
38    );
39  }
```

Nest to Custom Cert

```
    cause: Error: self-signed certificate
      at TLSSocket.onConnectSecure (node:_tls_wrap:1674:34)
      at TLSSocket.emit (node:events:514:28)
      at TLSSocket._finishInit (node:_tls_wrap:1085:8)
      at ssl.onhandshakedone (node:_tls_wrap:871:12) {
        code: 'DEPTH_ZERO_SELF_SIGNED_CERT'
      }
    }
[Nest] 49390 - 2024-09-19, 6:58:41 p.m.  ERROR [ExceptionHandler] Error fetching data
Error: Error fetching data
  at <anonymous> (/Users/aj/dev/azure-test/my-nest-app/src/app.service.ts:36:15)
  at <anonymous> (/Users/aj/dev/azure-test/my-nest-app/node_modules/rxjs/src/internal/operators/catchError.ts:117:35)
  at OperatorSubscriber.OperatorSubscriber._this._error (/Users/aj/dev/azure-test/my-nest-app/node_modules/rxjs/src/internal/operators/OperatorSubscriber.ts:11:24)
  at OperatorSubscriber.Subscriber.error (/Users/aj/dev/azure-test/my-nest-app/node_modules/rxjs/src/internal/Subscriber.ts:91:12)
  at OperatorSubscriber.Subscriber._error (/Users/aj/dev/azure-test/my-nest-app/node_modules/rxjs/src/internal/Subscriber.ts:124:24)
  at OperatorSubscriber.Subscriber.error (/Users/aj/dev/azure-test/my-nest-app/node_modules/rxjs/src/internal/Subscriber.ts:91:12)
  at /Users/aj/dev/azure-test/my-nest-app/node_modules/@nestjs/axios/dist/http.service.js:65:28
  at process.processTicksAndRejections (node:internal/process/task_queues:95:5)
```



Could also be *like*:
UNABLE_TO_VERIFY_LEAF_SIGNATURE

Nest to Custom Cert Improved

19 Answers

Sorted by: Highest score (default)



Note: the following is dangerous, and will allow API content to be intercepted and modified between the client and the server.

228

This also worked



```
process.env['NODE_TLS_REJECT_UNAUTHORIZED'] = '0';
```



[Share](#) [Improve this answer](#) [Follow](#)

edited May 18, 2018 at 17:24



mikemaccana

121k ⚡ 108 🔍 421 🏷 521

answered Nov 20, 2013 at 15:51



ThomasReggi

58.6k ⚡ 96 🔍 254 🏷 455



Yolo; 'lunch in 5' solution

Nest to Custom Cert Improved (a bit)

```
41  getCustomFixed(): Observable<string> {
42    return this.httpService
43      .get('https://nodejs-pdc24.com/users', {
44        httpsAgent: new https.Agent({
45          rejectUnauthorized: false,
46        }),
47      })
48      .pipe(
49        map(({ data }) => {
50          return data;
51        }),
52        catchError((error: AxiosError) => {
53          console.error(error);
54          throw new Error('Error fetching data');
55        }),
56      );
57}
```

Just ignore ssl issues

Nest to Custom Cert Improved (legit)

```
41     getCustomFixed(): Observable<string> {
42         return this.httpService
43             .get('https://nodejs-pdc24.com/users', {
44                 httpsAgent: new https.Agent({
45                     >         ca: `-----BEGIN CERTIFICATE-----...
46                     ,
47                     [
48                         ],
49                     }
50                     .pipe(
51                         map(({ data }) => {
52                             return data;
53                         }),
54                         catchError((error: AxiosError) => {
55                             console.error(error);
56                             throw new Error('Error fetching data');
57                         }),
58                     );
59                 });
60             });
61     }
62 }
```

Just use the custom certificate



Conclusion

Securing the web

Theory of Https

What is Https made of

OpenSSL

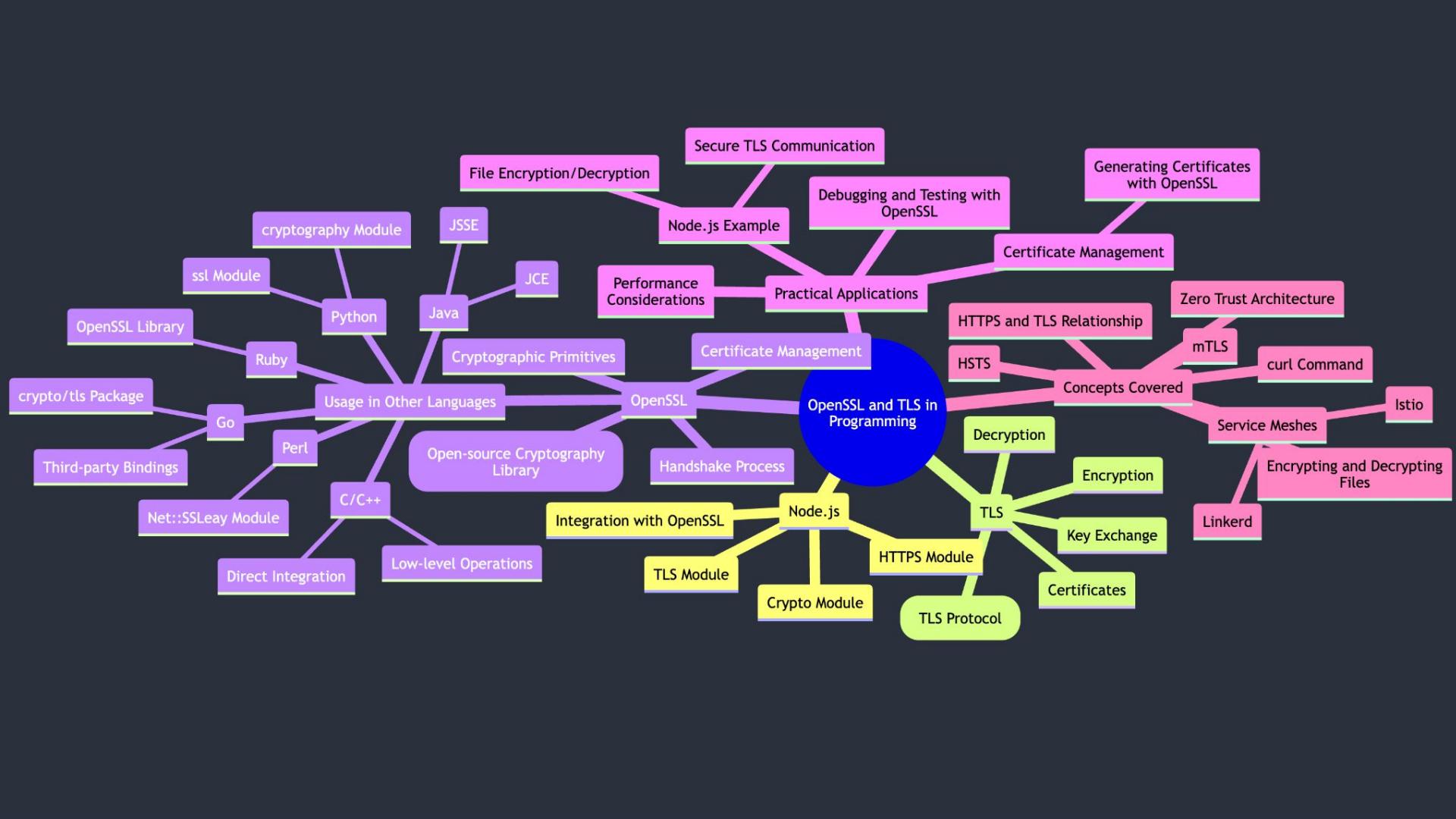
Commands to google
every time

Practice of Https

Couple examples of
Https in action

Certificates

On by default 'til
Enterprises
man-in-the-middle



Thanks!



Keep in touch

✉ aj@bitovi.com

in /softwarebyaj/



bitovi