Securing the Web

Mastering HTTPS for Developers

bitovi

PRAIRIE DEV CON

WEB | DEV | CLOUD | AI

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.

bitovi

Thanks!

Keep in touch

aj@bitovi.com

in /softwarebyaj/

bitovi

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

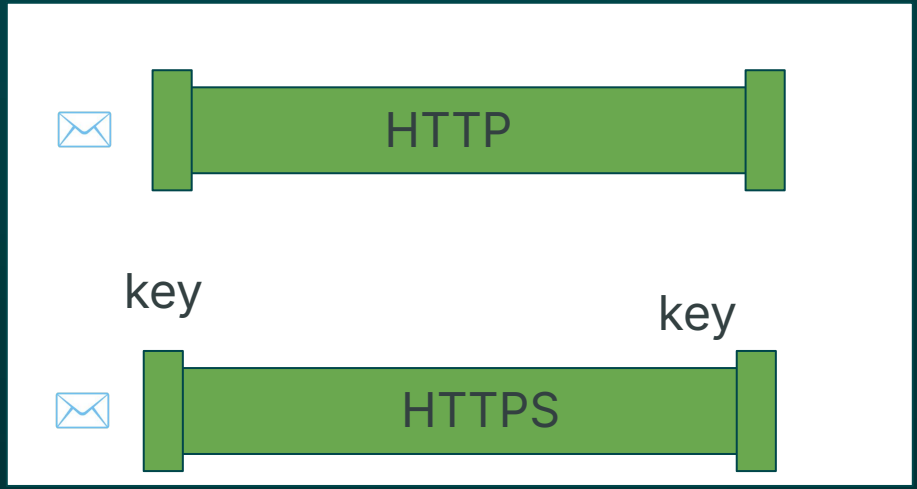
I'm not here to shill some library

HTTPS

Definitely someone else's problem

Until...

It wasn't



HTTP + Secure

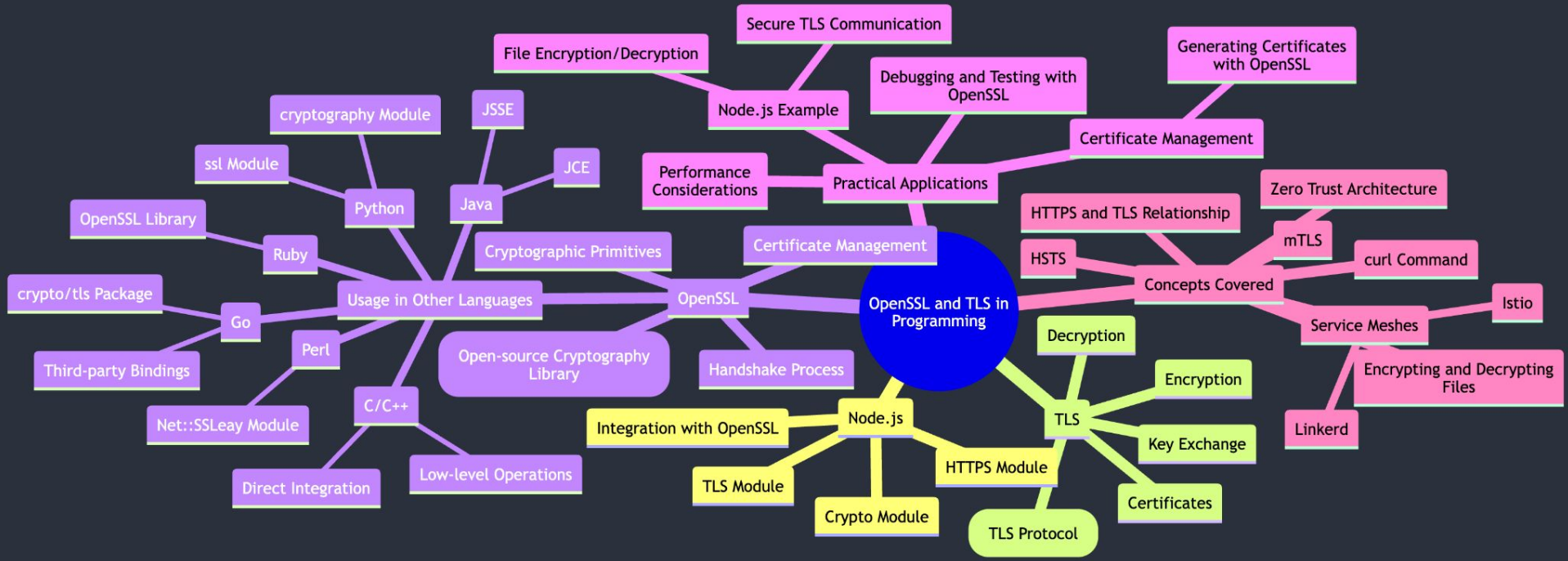
Just say what the acronym is

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...


OpenSSL and TLS in Programming
Usage in Other Languages
Python
cryptography Module
ssl Module
Java
JSSE
JCE
Ruby
OpenSSL Library
Go
crypto/tls Package
Third-party Bindings
Perl
Net::SSLeay Module
C/C++
Direct Integration
Low-level Operations
OpenSSL
Cryptographic Primitives
Open-source Cryptography Library
Certificate Management
Handshake Process
Practical Applications
Node.js Example
File Encryption/Decryption
Secure TLS Communication
Performance Considerations
Debugging and Testing with OpenSSL
Certificate Management
Generating Certificates with OpenSSL
Node.js
Integration with OpenSSL
TLS Module
Crypto Module
HTTPS Module
TLS
Decryption
Encryption
Key Exchange
Certificates
TLS Protocol
Concepts Covered
HTTPS and TLS Relationship
HSTS
mTLS
Zero Trust Architecture
curl Command
Service Meshes
Istio
Linkerd
Encrypting and Decrypting Files

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

Testing Security Features

HSTS requires HTTPS

Consistency

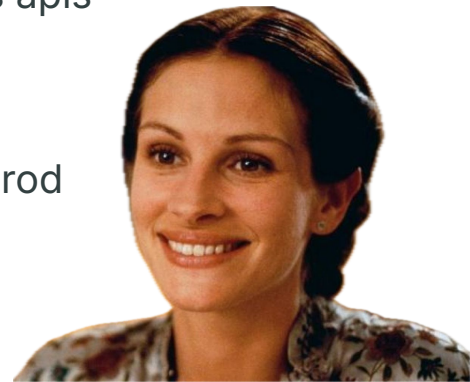
Local match dev, test, and prod

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

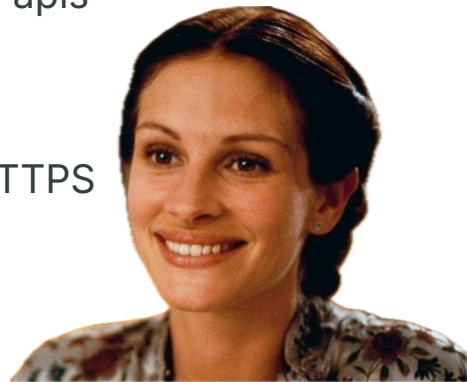
Local match dev, test, and prod

Avoid Mixed Content Issues

If there’s already some https apis

External API

Most external APIs will be HTTPS

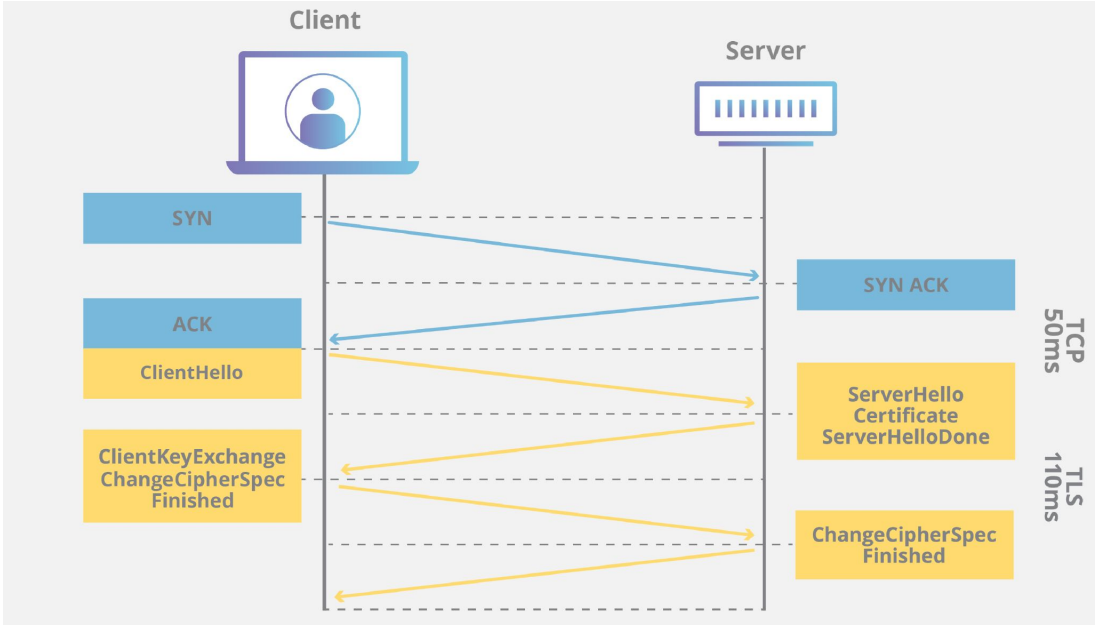


Theory of HTTPS

Theory of HTTPS

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

HTTPS Process: Part 1 TLS Handshake



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:

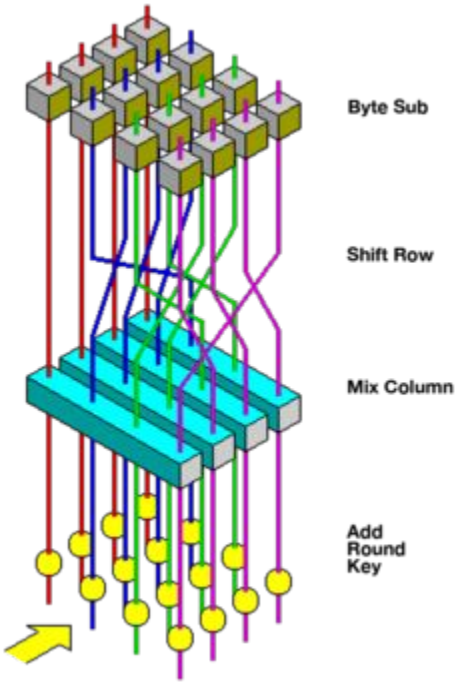
> Securely exchange a secret (symmetric key) between client/server
>
> RSA, Diffie-Hellman, Elliptic Curve

Symmetric:

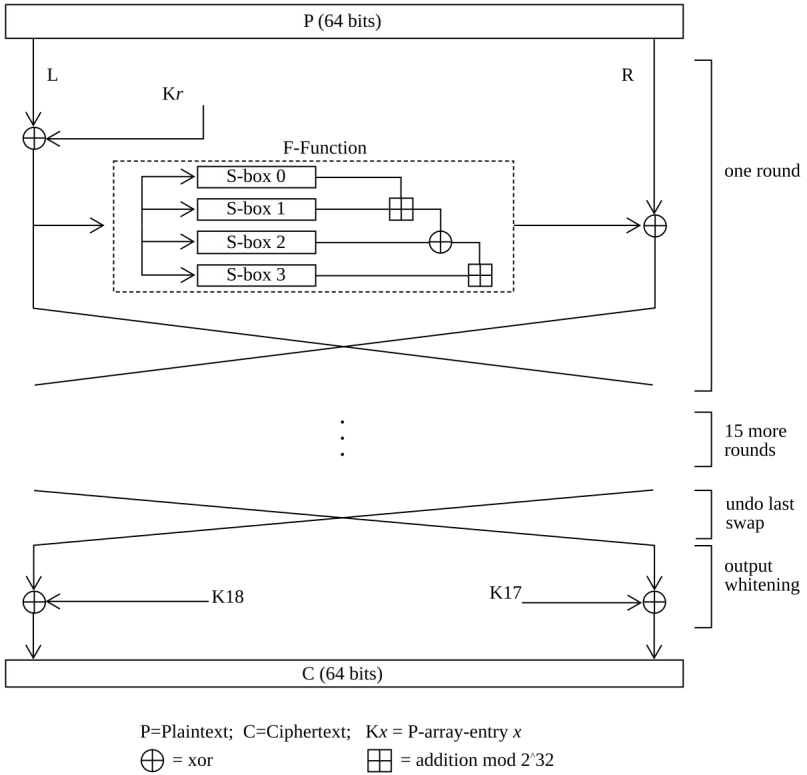
> Shared key: encrypts/decrypts data

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

AES / Blowfish Diagrams



AES



Blowfish

Cryptography intro: Hashing

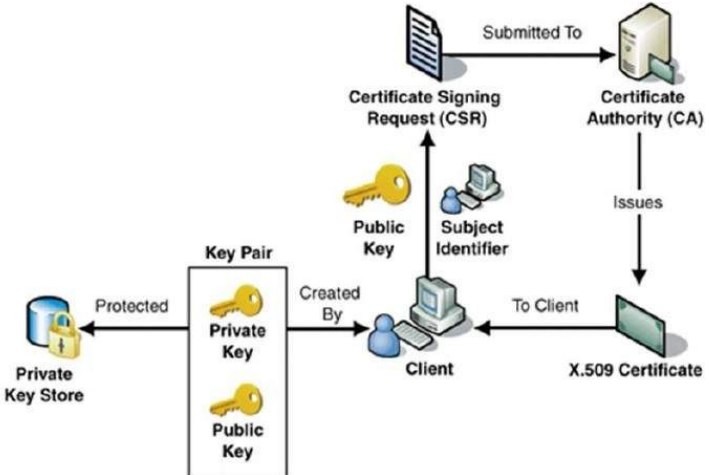
- Crypto = Hash(Crypto, Hash(Crypto))
- Hash function create a hash
  - data of fixed length based on input variable length data
- Small change to input result in large change to hash

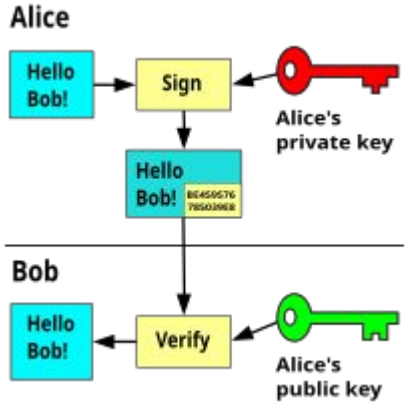
NPM Hash

```
❯ 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+MZmycf2b3ceVvANDJFkDEodobVtyG63WvGOhkZ3aGhT7sHguKpAQwJLicSf8zF2z+v1Yi0DvRw==
.unpackedSize: 634.8 kB
```

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


```
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 a
"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.

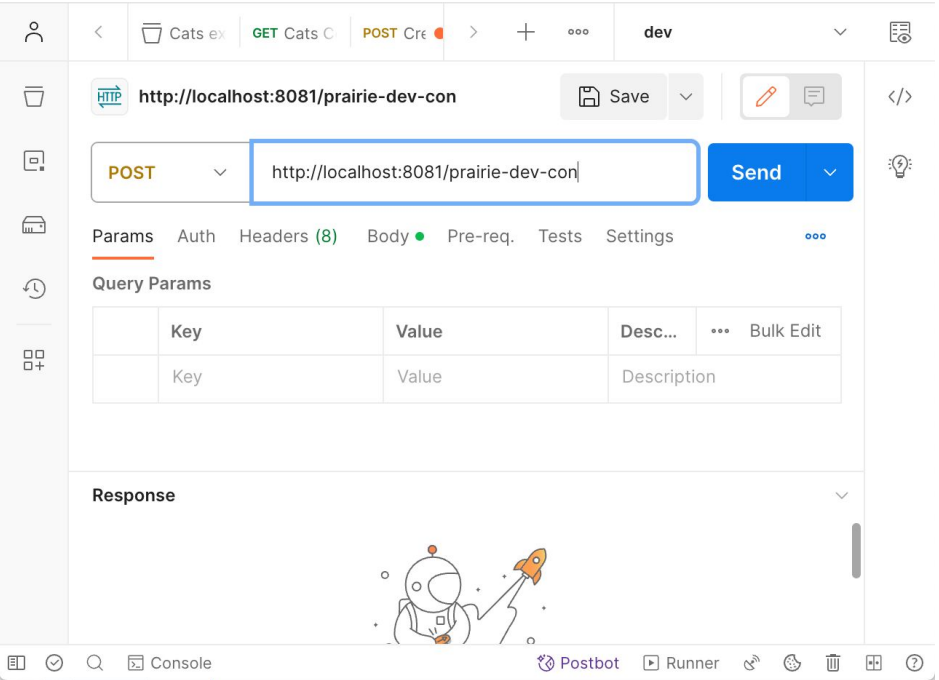
Common Tools: Certificate Authorities

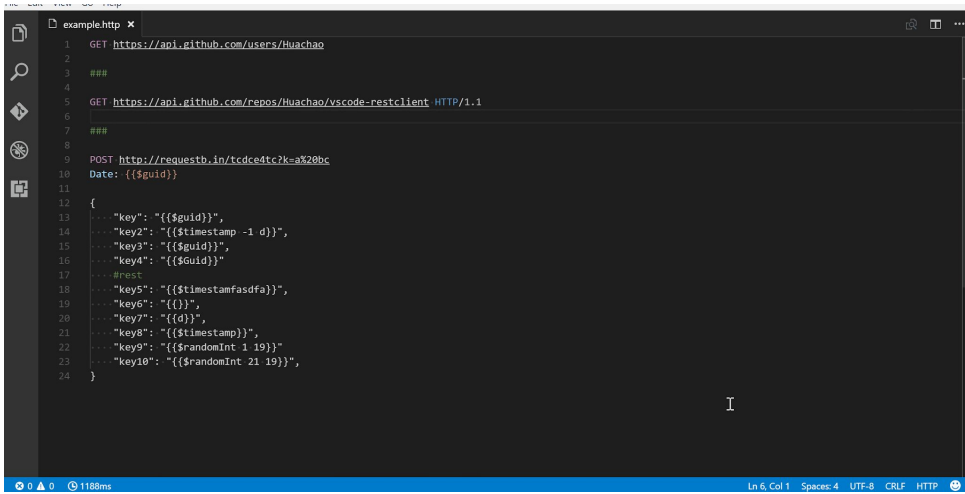
- https://letsencrypt.org/
- https://www.digicert.com/
- https://www.entrust.com/
- https://www.comodo.com/
- Your cloud provider (Azure, AWS, Google Cloud etc)

- Already mentioned the CA purpose
  - Facilitate the digital signature

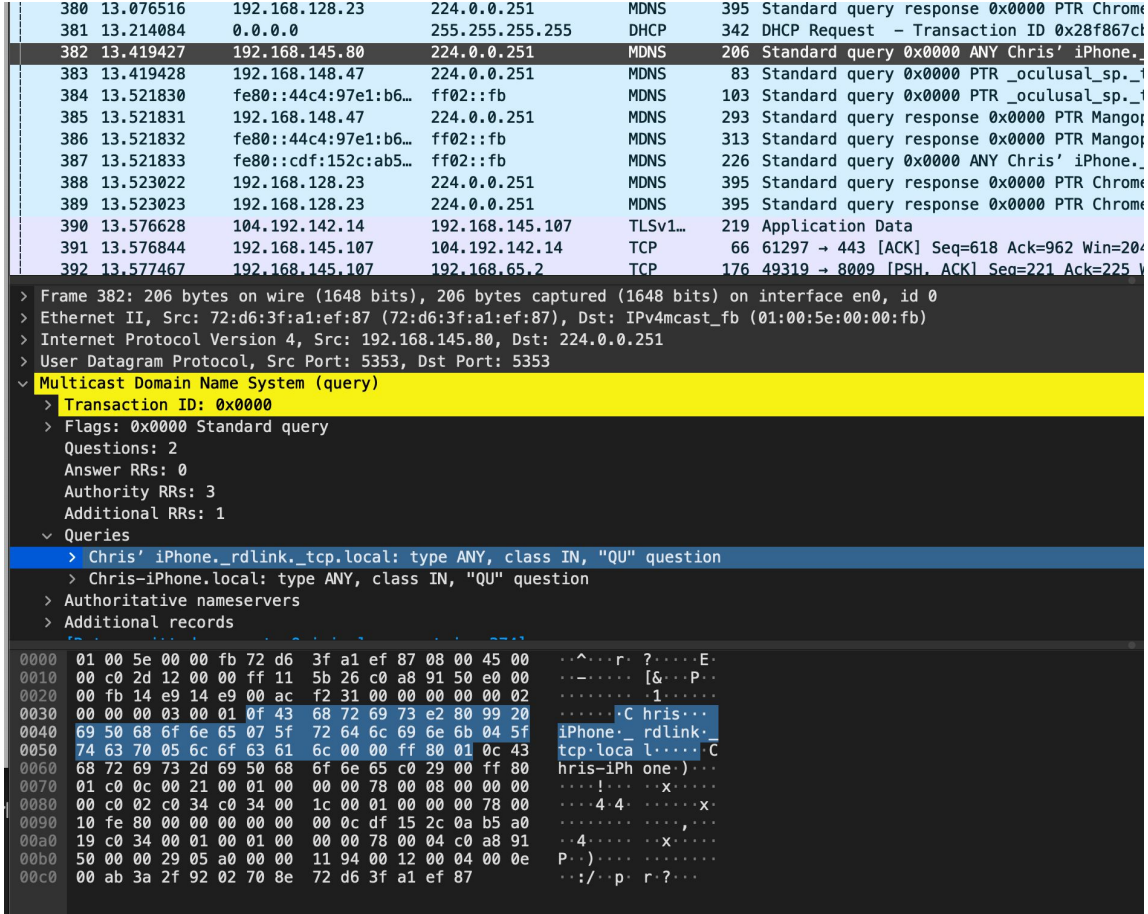
Common Tools: Curl / Postman / Rest Client

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





Common Tools: Wireshark

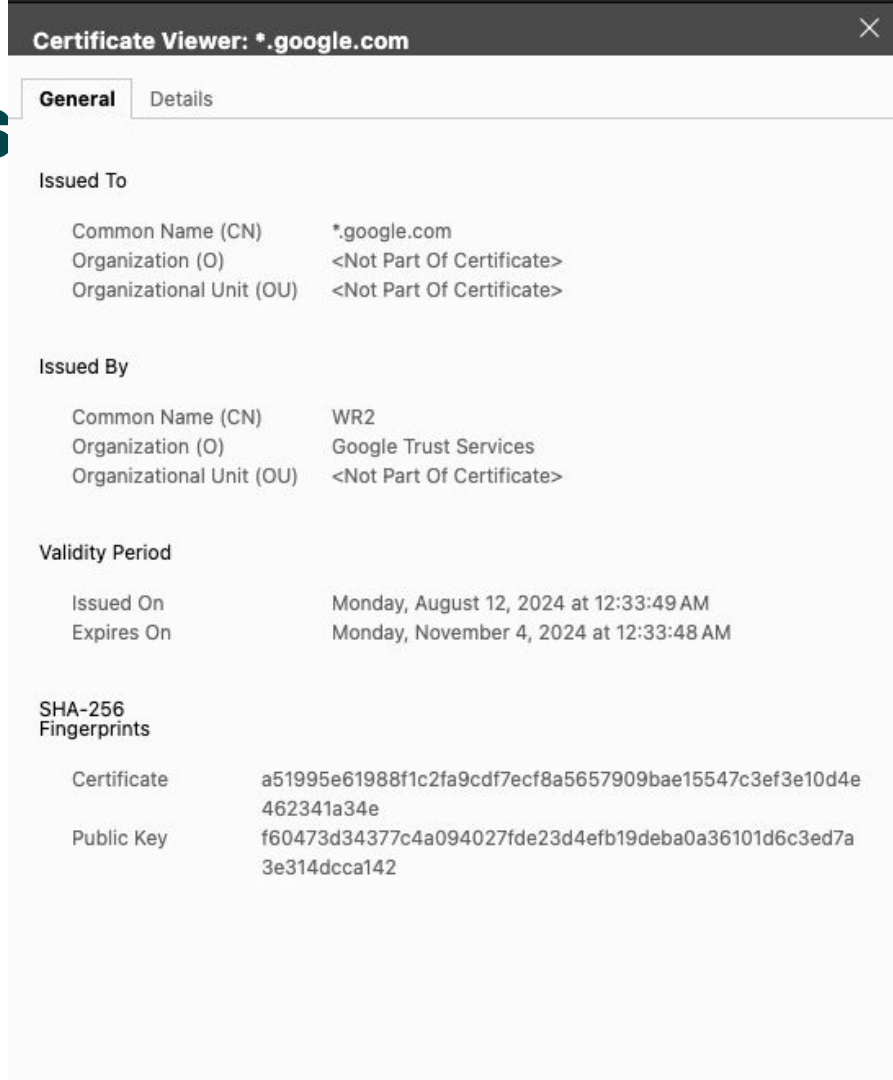


Common Tools: Web Server Configuration

- More on this in the next section

HTTPS Vulnerabilities

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



Practicing HTTPS

Practicing HTTPS

Smallest example

Deploy nodejs app to azure

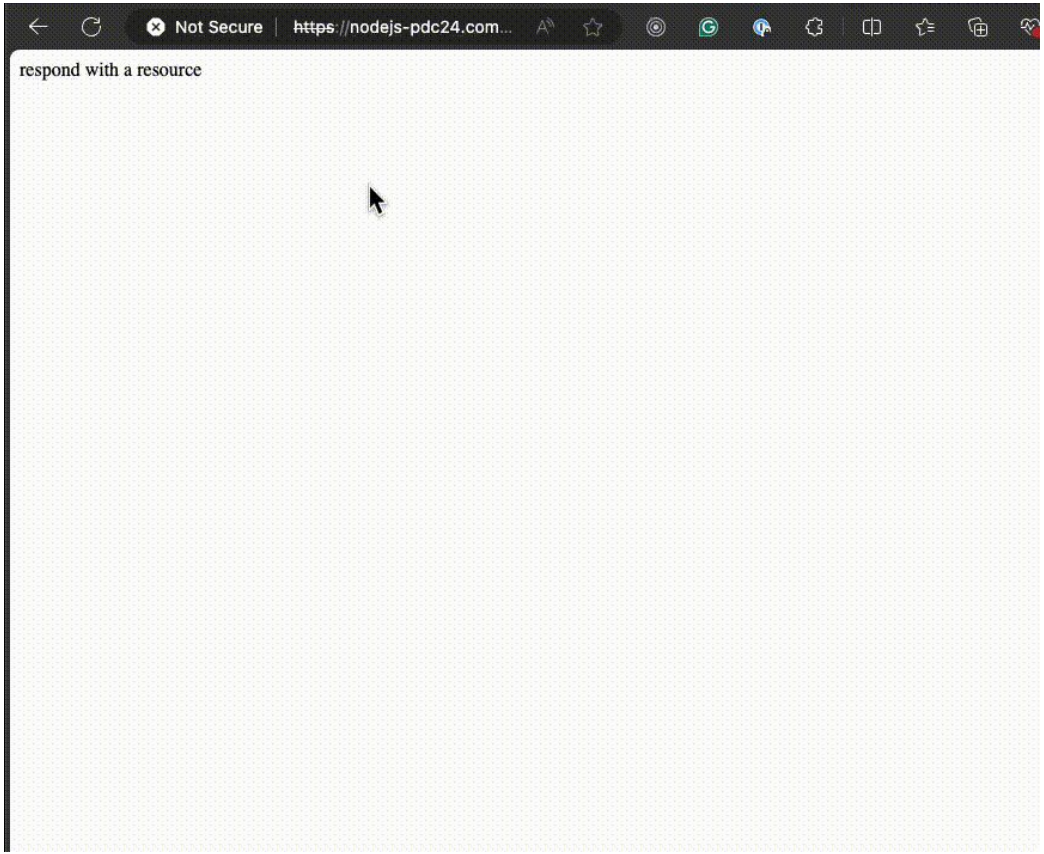
Custom certificates

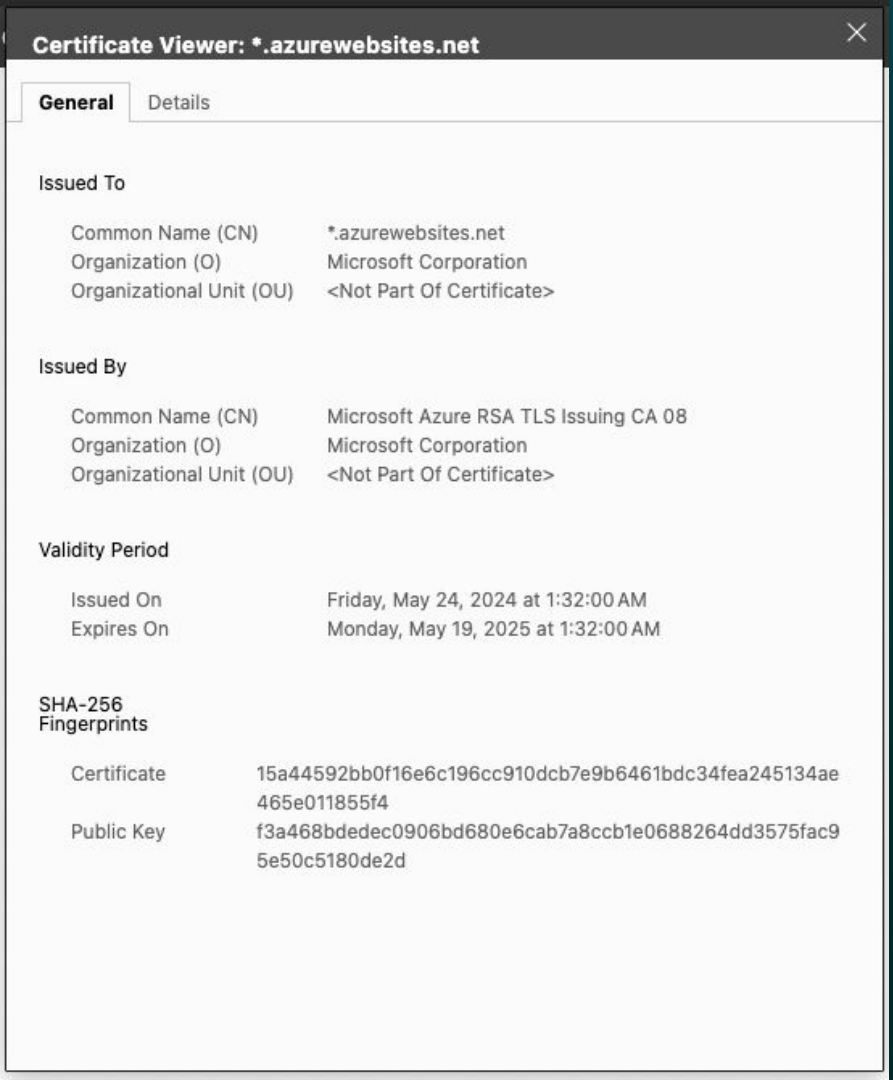
Same as above with custom cert

Consuming custom certificates

Issues and resolutions

How to get a website's certificate

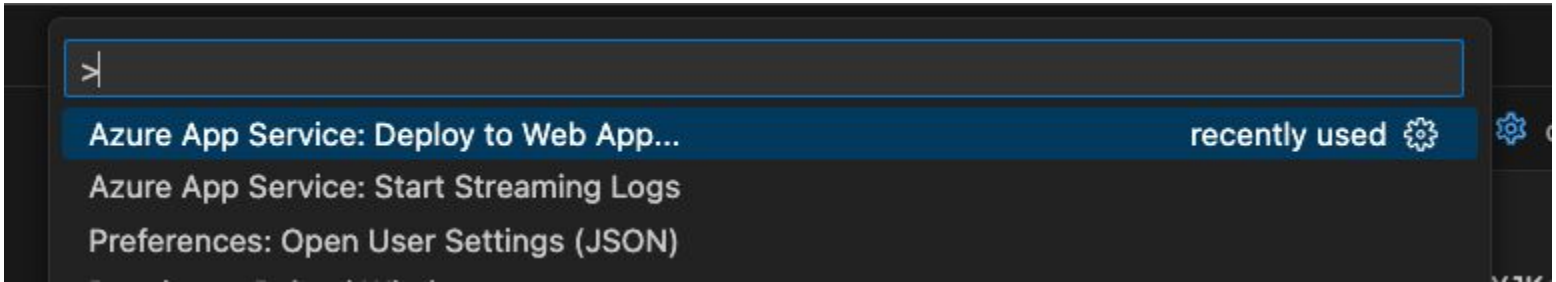


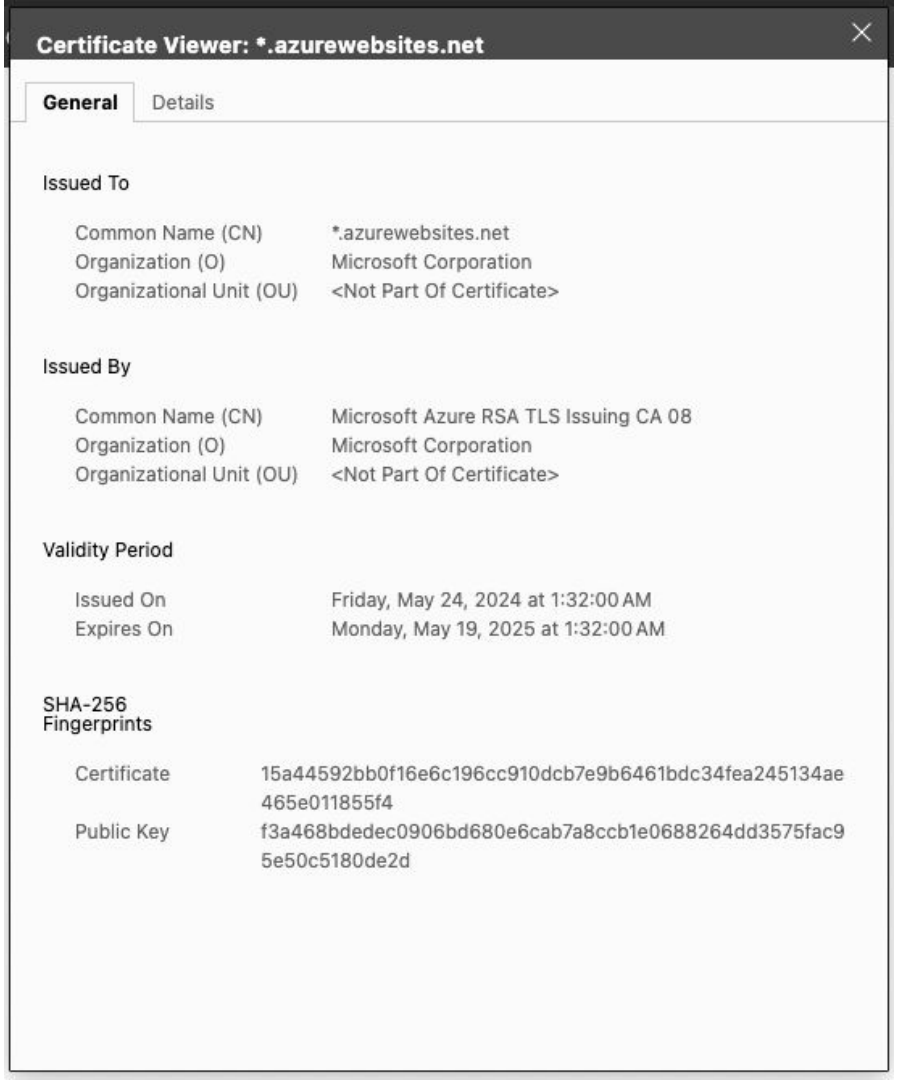


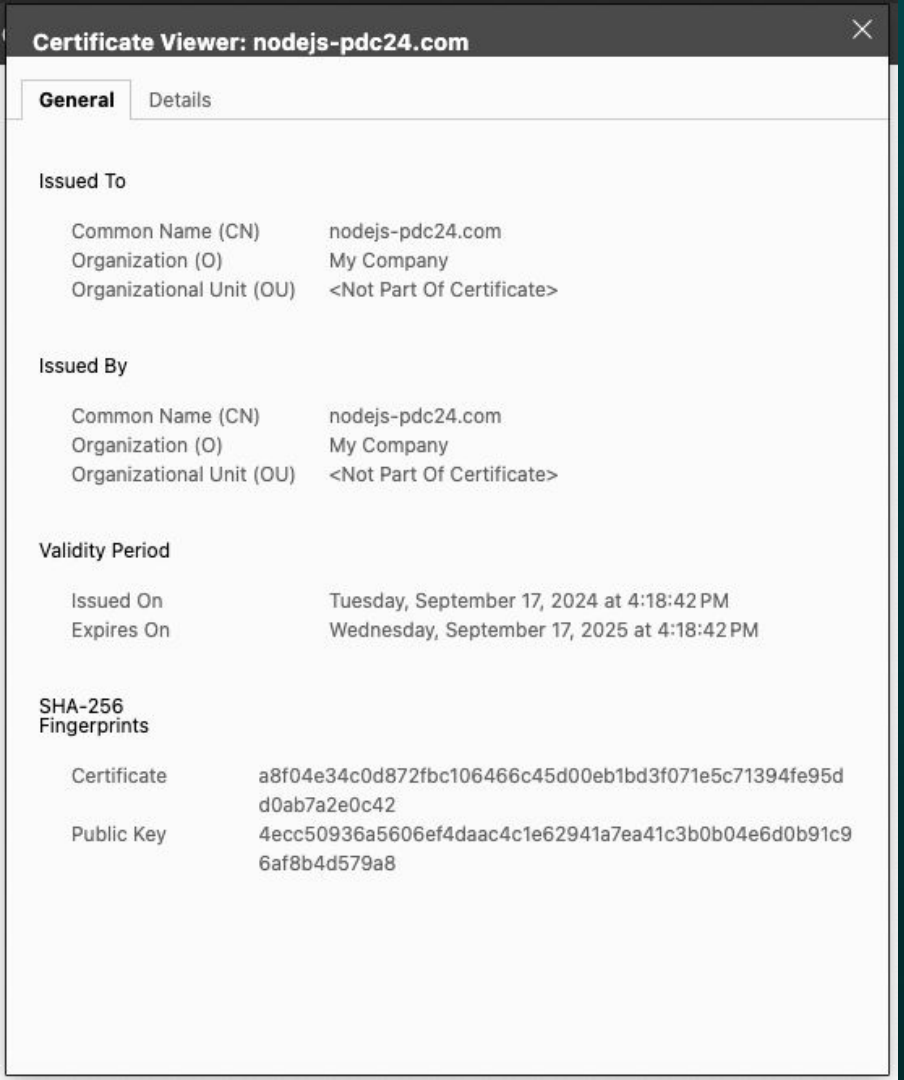
Smallest example

Smallest Example

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




Certificate Viewer: *.azurewebsites.net
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 15a44592bb0f16e6c196cc910dcb7e9b6461bdc34fea245134ae465e011855f4
Public Key f3a468bdedec0906bd680e6cab7a8ccb1e0688264dd3575fac95e50c5180de2d

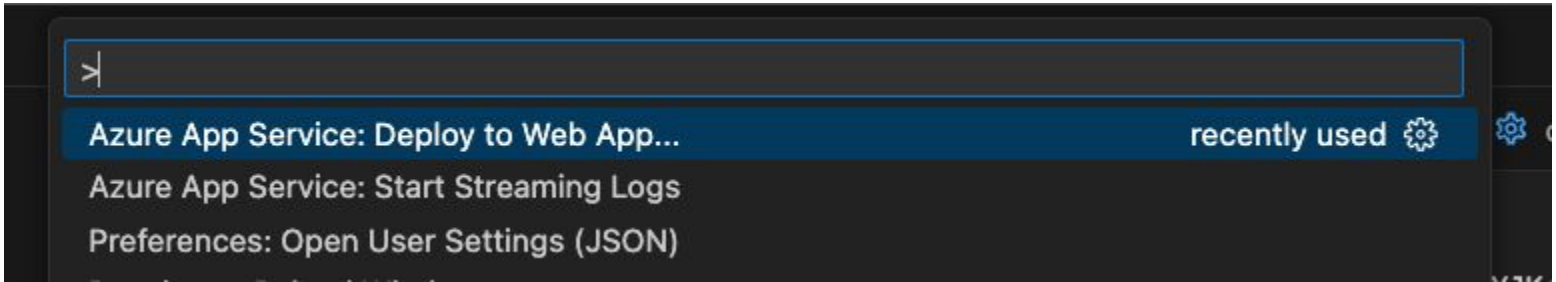


Custom Certificates

How to add your own certificate

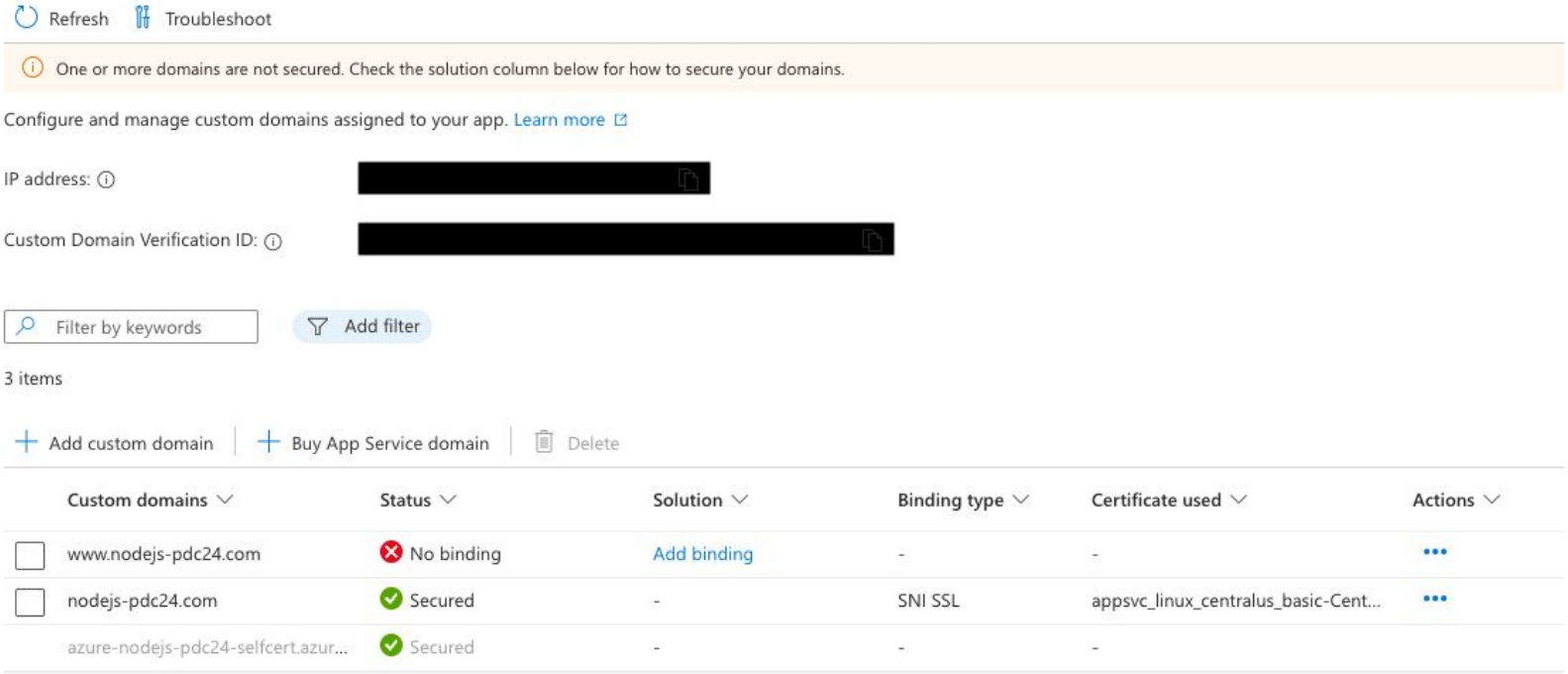
Custom Certificates

- 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

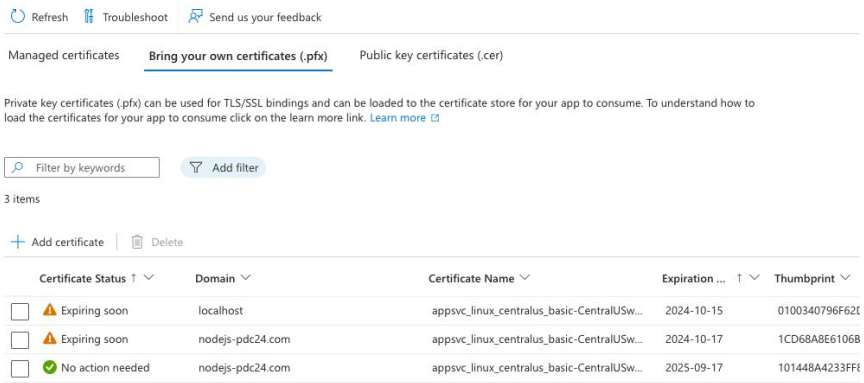
- Get a custom domain



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
```



openssl.cnf

```
[ req ]
default_bits       = 2048
distinguished_name = req_distinguished_name
req_extensions     = req_ext
x509_extensions    = v3_ca # The extensions to add to the self-signed cert

[ req_distinguished_name ]
countryName                 = Country Name (2 letter code)
countryName_default         = US
stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = California
localityName                = Locality Name (eg, city)
localityName_default        = San Francisco
organizationName            = Organization Name (eg, company)
organizationName_default    = My Company
commonName                  = Common Name (e.g. server FQDN or YOUR name)
commonName_default          = nodejs-pdc24.com
commonName_max              = 64

[ req_ext ]
subjectAltName = @alt_names

[ alt_names ]
DNS.1   = nodejs-pdc24.com

[ v3_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer
basicConstraints = CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth
```

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  }
```

It just works, thanks microsoft

Nest to Custom Cert

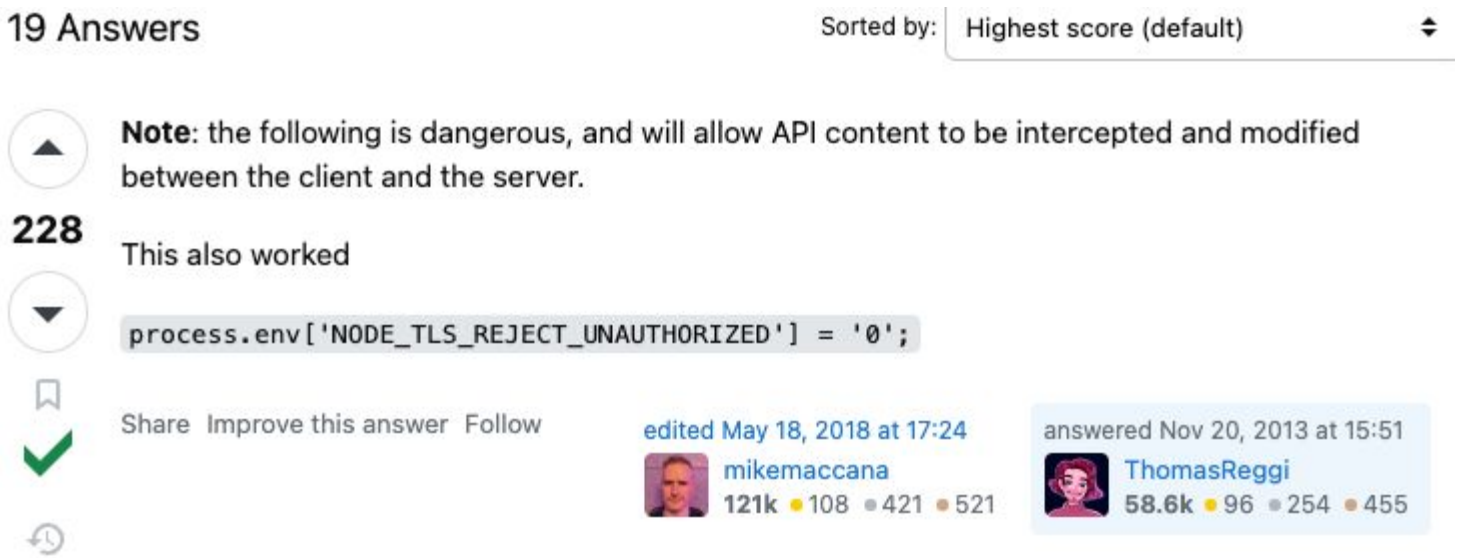
```
29    getCustom(): Observable<string> {
30      return this.httpService.get('https://nodejs-pdc24.com/users').pipe(
31        map(({ data }) => {
32          return data;
33        }),
34        catchError((error: AxiosError) => {
35          console.error(error);
36          throw new Error('Error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



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)

Just use the custom certificate

Conclusion

Securing the web

Theory of Https

What is Https made of

Practice of Https

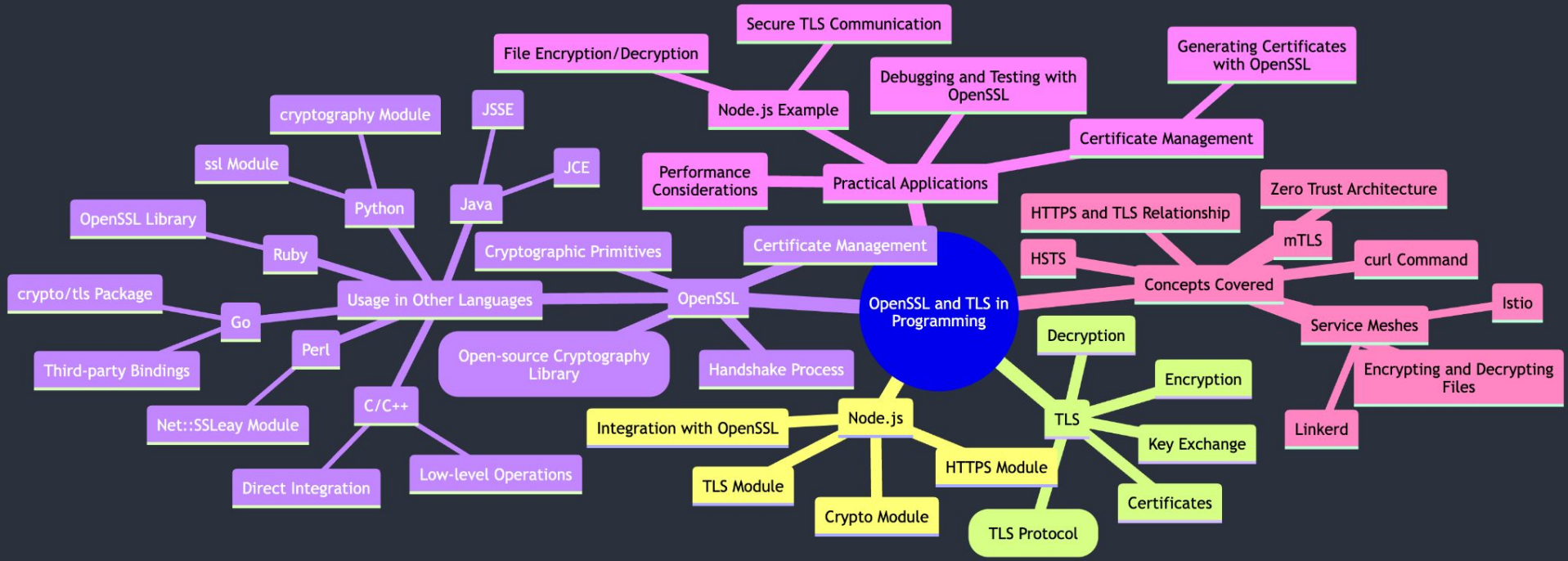
Couple examples of Https in action

OpenSSL

Commands to google every time

Certificates

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


OpenSSL and TLS in Programming
Usage in Other Languages
Python
cryptography Module
ssl Module
Java
JSSE
JCE
Ruby
OpenSSL Library
Go
crypto/tls Package
Third-party Bindings
Perl
Net::SSLeay Module
C/C++
Direct Integration
Low-level Operations
OpenSSL
Cryptographic Primitives
Open-source Cryptography Library
Certificate Management
Handshake Process
Practical Applications
Node.js Example
File Encryption/Decryption
Secure TLS Communication
Performance Considerations
Debugging and Testing with OpenSSL
Certificate Management
Generating Certificates with OpenSSL
Node.js
Integration with OpenSSL
TLS Module
Crypto Module
HTTPS Module
TLS
Decryption
Encryption
Key Exchange
Certificates
TLS Protocol
Concepts Covered
HTTPS and TLS Relationship
HSTS
mTLS
Zero Trust Architecture
curl Command
Service Meshes
Istio
Linkerd
Encrypting and Decrypting Files

Thanks!

Keep in touch

aj@bitovi.com

in /softwarebyaj/

bitovi