Exercise (Instructions): HTTPS and Secure Communication | Coursera

Objectives and Outcomes

In this exercise you will explore the use of the HTTPS server core node module to create and run a secure server. You will also learn about generating your private key and public certificate and use them to configure your Node HTTPS server. At the end of this exercise, you will be able to:

- Configure a secure server in Node using the core HTTPS module
- Generate the private key and public certificate and configure the HTTPS server
- Redirect traffic from the insecure HTTP server to a secure HTTPS server.

Updating the bin/www File

- Open the *www* file in the *bin* directory and update its contents as follows:

```
#!/usr/bin/env node
/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('rest-server:server');
var http = require('http');
var https = require('https');
var fs = require('fs');

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');

app.set('port', port);
app.set('secPort',port+443);

/**
 * Create HTTP server.
 */
```

```
var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port, function() {
   console.log('Server listening on port ',port);
});
server.on('error', onError);
server.on('listening', onListening);

/**
 * Create HTTPS server.
 */ var options = {
  key: fs.readFileSync(__dirname+'/private.key'),
  cert: fs.readFileSync(__dirname+'/certificate.pem')
};

var secureServer = https.createServer(options,app);

/**
 * Listen on provided port, on all network interfaces.
 */

secureServer.listen(app.get('secPort'), function() {
   console.log('Server listening on port ',app.get('secPort'));
});
secureServer.on('error', onError);
secureServer.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);
  if (isNaN(port)) {
    // named pipe
    return val;
```

```
  }
  if (port >= 0) {
    // port number
    return port;
  }
  return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }
  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  // handle specific listen errors with friendly messages
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;

    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;

    default:
      throw error;
  }
}

/**
 * Event listener for HTTP server "listening" event.
 */
```

```
function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}
```

Updating app.js

- Open app.js and add the following code to the file:

```
// Secure traffic only
app.all('*', function(req, res, next){
    console.log('req start: ',req.secure, req.hostname, req.url,
app.get('port'));
  if (req.secure) {
    return next();
  };

 res.redirect('https://'+req.hostname+':'+app.get('secPort')+req.url);
});
```

Generating Private Key and Certificate

- Go to the bin folder and then create the private key and certificate by typing the following at the prompt:

```
openssl genrsa 1024 > private.key
openssl req -new -key private.key -out cert.csr
openssl x509 -req -in cert.csr -signkey private.key -out certificate.pem
```

Note for Windows Users

- If you are using a Windows machine, you may need to install openssl. You can find some openssl binary distributions here. Also, this article gives the steps for generating the certificates in Windows. Another article provides similar instructions. Here's an online service to generate self-signed certificates.
- Run the server and test.

Conclusions

In this exercise, you learnt about configuring a secure server running HTTPS protocol in our Express application.